Co-manipulation with a library of virtual guides

Gennaro Raiola

NNT : 2017SACLY001

THÈSE DE DOCTORAT
DE
L'UNIVERSITÉ PARIS-SACLAY
PRÉPARÉE À
"L'ENSTA-PARISTECH"

ECOLE DOCTORALE N° (573)
INTERFACE

Spécialité de doctorat (Automatique)

Par

M Gennaro Raiola

Co-manipulation with a library of Virtual Guides

Thèse présentée et soutenue à « Palaiseau », le «02 02 2017» :

Composition du Jury :

M, Geffard, Franck	Ingénieur de recherche, CEA-List	Président
M, Calinon, Sylvain	Chercheur, Idiap Research Institute	Rapporteur
M, Lopes, Manuel	Maître de conférence, Instituto Superior Técnico	Rapporteur
Mme, Tapus, Adriana	Professeur, ENSTA-ParisTech	Directeur de thèse
M, Stulp, Freek	Chef de département, DLR	Co-directeur de thèse
M, Lamy, Xavier	Ingénieur de recherche, CEA-List	Co-directeur de thèse
M, Tliba, Sami	Maître de conférence, Supelec	Invité
M, Rodriguez, Pedro	Professeur, Supelec	Invité

Summary

Robots have a fundamental role in industrial manufacturing. They not only increase the efficiency and the quality of production lines, but also drastically decrease the work load carried out by humans. However, due to the limitations of industrial robots in terms of flexibility, perception and safety, their use is limited to well-known structured environment. Moreover, it is not always cost-effective to use industrial autonomous robots in small factories with low production volumes. This means that human workers are still needed in many assembly lines to carry out specific tasks. Therefore, in recent years, a big impulse has been given to human-robot co-manipulation. By allowing humans and robots to work together, it is possible to combine the advantages of both; abstract task understanding and robust perception typical of human beings with the accuracy and the strength of industrial robots. One successful approach to facilitate human-robot co-manipulation is the use of Virtual Guides which constrain the motion of the robot along certain task-relevant trajectories. The virtual guide acts as a passive tool that improves the performances of the user in terms of task time, mental workload and errors. The innovative aspect of our work is to present a library of virtual guides that allows the user to easily select, generate and modify the guides through an intuitive haptic interaction with the robot. We demonstrated in two industrial tasks that these innovations provide a novel and intuitive interface for joint human-robot completion of tasks.

Résumé

Les robots ont un rôle fondamental dans la fabrication industrielle. Non seulement ils augmentent l'efficacité et la qualité des lignes de production, mais aussi diminuent considérablement la charge de travail des humains. Cependant, en raison des limites des robots industriels en termes de flexibilité, de perception et de sécurité, Leur utilisation est limitée à un environnement structuré bien connu. En outre, il n'est pas toujours rentable d'utiliser des robots autonomes industriels dans de petites usines à faibles volumes de production. Cela signifie que des travailleurs humains sont encore nécessaires dans de nombreuses chaînes d'assemblage pour exécuter des tâches spécifiques. Par conséquent, ces dernières années, une grande impulsion a été donnée à la co-manipulation homme-robot. En permettant aux humains et aux robots de travailler ensemble, il est possible de combiner les avantages des deux; La compréhension des tâches abstraites et la perception robuste typique d'un être humain avec la précision et la force d'un robot industriel. Une approche réussie pour faciliter la co-manipulation homme-robot, est l'approche de guides virtuels qui contraint le mouvement du robot sur seulement certaines trajectoires pertinentes. Le guide virtuel ainsi réalisé agit comme un outil passif qui améliore les performances de l'utilisateur en termes de temps de tâche, de charge de travail mentale et d'erreurs. L'aspect innovant de notre travail est de présenter une bibliothèque de guides virtuels qui permet à l'utilisateur de facilement sélectionner, générer et modifier les guides grâce à une interaction intuitive haptique avec le robot. Nous avons démontré, dans deux tâches industrielles, que ces innovations fournissent une interface novatrice et intuitive pour l'accomplissement des tâches par les humains et les robots.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In co-manipulation, humans and robots solve manipulation tasks together. Virtual guides are important tools for co-manipulation, as they constrain the movement of the robot to avoid undesirable effects, such as collisions with the environment. Defining virtual guides is often a laborious task requiring expert knowledge. This restricts the usefulness of virtual guides in environments where new tasks may need to be solved, or where multiple tasks need to be solved sequentially, but in an unknown order. To this end, we propose a framework for multiple virtual guides. Our approach enables non-expert users to create a library of virtual guides through demonstrations. Also, they may demonstrate new guides, even if already known guides are active. Finally, users are able to intuitively select the appropriate guide from a set of guides through physical interaction with the robot.

This work was funded by DIGITEO, Doctoral Fellowship 2014 on *collaborative robots for production*. It took place from January 2014 to December 2016 at ENSTA-ParisTech and CEA-List. The subject of this work is multidisciplinary, between the area of robotics, controls, machine learning and software engineering.

1.1 Context

Since the last decade, the demand for industrial robots has increased considerably due to the continuous investments in automation and the ongoing improvements of industrial robots (Fig-

ure 1.1). Thanks to the great developments in areas like sensor technology, human-machine in-

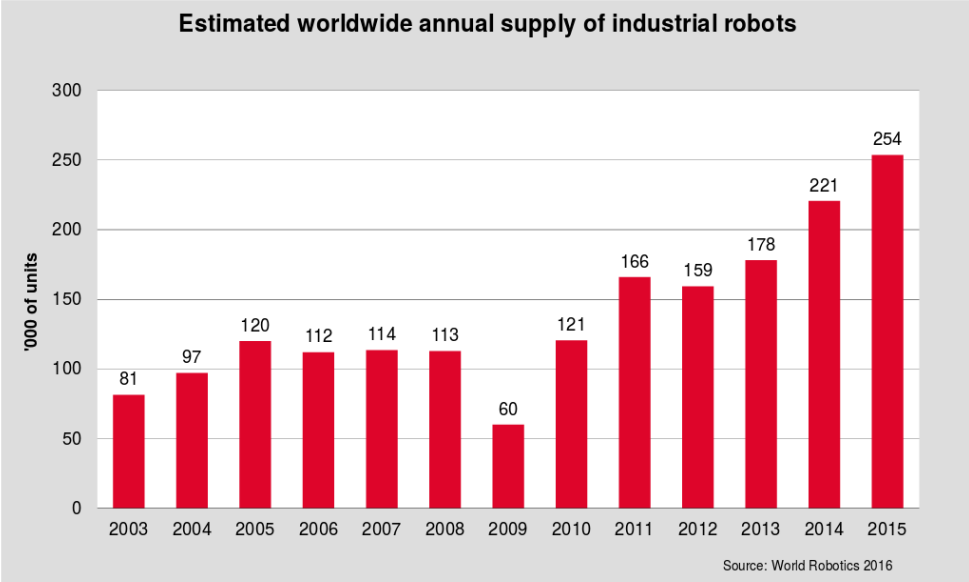


Figure 1.1: In 2015 robot sales increased by 15% to 253,748 units. This is by far the highest level ever recorded for one year (Source taken from [WorldRobotics, 2016])

terfaces and high-level programming (Figure 1.2), robots are now easier to install and program.

These new technologies represent a big opportunity to develop robotics and automation solu-

Figure 1.2: Franka robot. In recent years a lot of effort have been made to improve human-machine interaction. New devices such as smart phones, tablets and augmented reality devices make easier to interact with a robot.

tions for the new industrial revolution that the experts call *Industry 4.0* [Hermann et al., 2016].

The vision of Industry 4.0 is to have assembly lines where humans and robots communicate

and cooperate with each other directly in order to have a strong customization of products and high flexibility in production. This way, the production cycle can easily accommodate customer preferences in any of the production steps, from the product conception to its development. On a larger scale, this will represent for the industry the possibility to be highly responsive to changes in the marketplace while keeping mass-production capability and cost effectiveness. Such flexibility in the production lines requires dexterity and problem-solving skills, something that the industrial robots are not capable of, thus keeping heavily dependent on human workers most of the productions where flexibility and customization is a concern. Unfortunately, creating robots with human-like dexterity and problem-solving skills it is not possible yet. Moreover, programming a modern industrial robot is still a tedious and time-consuming task that requires technical expertise to be accomplished. In general, an industrial robot is programmed to perform a specific task, and thus, unskilled users will not be able to easily re-program the robot to perform a different task. This contrasts with the idea of flexibility at the core of Industry 4.0.

One promising solution to facilitate the interaction between humans and robots reside in the recent development of cobots. Cobot is a neologism from the words *collaborative* and *robot*. The objective of these cobots is to automate a large range of tasks and to perform work in closer collaboration with people without any need of fences or restraining security measures (Figure 1.3). Cobots are intended for productive interaction with humans. Some of their tech-

Figure 1.3: A cobot from Universal Robots working on a motor assembly.

nical characteristics are: lightweight design with rounded and padded surfaces, human-like dimensions, moderate payload with limitations on speed and force, making them both harmless and productive. Most important is their flexibility, which facilitates redeployment for variable tasks in mixed human-robot environments. This enables a variable degree of automation in production, with people and robots working together, focusing on their respective strengths: robots can contribute to ensure the quality, repeatability and endurance in specific steps, while humans contribute with their superior skills in complex manipulation, problem solving and visual perception. Since cobots are intrinsically safe, it is possible to use more explicit and human-friendly ways of instructing them. More specifically, it is possible to program them by demonstration by guiding the robot to perform the desired movements through kinesthetic teaching. Still there are situations where the human presence remains necessary, these occur when the human skills (perception, intuitiveness and abstract thinking) are needed and can not be reproduced by the cobot due to its limitations. To overcome these limitations, one successful approach that combines the advantages of both worlds is the *virtual guiding* approach [Colgate et al., 2003].

1.2 Virtual guides: Advantages and Limitations

A virtual guide constrains the robot movement along task-relevant trajectories. To do an analogy with the real world, a virtual guide can be compared to a ruler which can be used to draw straight lines with minimal effort (Figure 1.4). The ruler enforces a one dimensional constraint on a two dimensional surface (the paper); thanks to the reduced number of degrees of freedom, the precision is higher when drawing straight lines and the user cognitive effort is lower since he/she no longer needs to worry about fulfilling the constraint. Likewise, when using the virtual guides, the robot becomes a tool that improves the human's capabilities and enhances the task efficiency in terms of execution time, mental workload, safety and precision. These advantages make the virtual guides an useful tool for the human-robot comanipulation.

Unfortunately, the current approach is limited to situations where only one guide at a time

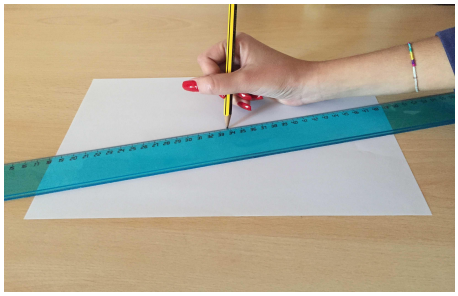
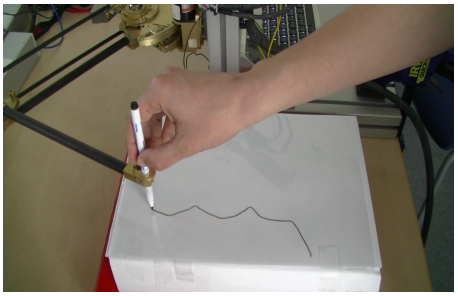



Figure 1.4: A virtual guide can be compared with a ruler. Both help the user to draw lines with minimal effort and high precision.

is used, moreover, creating a virtual guide can be laborious and could require precise modelling of the task.

Many real-life applications requires multiple operations to perform with the necessity to easily generate and adapt *on-line* the guides to different tasks and situations (e.g. transporting an object to one of multiple possible positions, different assembly sub-tasks to perform based on the tools availability, etc...). To this end, and to enhance the flexibility of using virtual guides, we defined the concept of *library* of probabilistic guides thanks to whom the user is able to create, modify, use individual virtual guides, and which provides a method to select the appropriate guide *on-line* through haptic interaction.

1.3 Contributions and Impact

To summarize, the main contributions of our work are the following:

- Extend virtual guides approach to **multiple virtual guides**, where users are able to select the appropriate guide on-line through physical interaction with the robot, and formalize this approach in a novel control scheme (chapter 4).
- Define several interaction modes that allow users to manage a **library of virtual guides** in order to create, modify and exploit the multiple virtual guides (chapter 5)
- Propose **probabilistic virtual guides**, which enable the two contributions above (chapter 3). Thanks to the probabilistic definition of virtual guides the user is able to interact

with the *library of virtual guides*. Moreover, the probabilistic definition gives the possibility to estimate which of the *multiple virtual guides* the user intends to select.

- Prove the **stability** of using multiple probabilistic virtual guides with the proposed control scheme. This ensures the safety of the human-robot interaction when using virtual guides (chapter 4).

- **Empirically evaluate** the usability of multiple virtual guides as well as their effect on the efficiency and ergonomy of executing two different industrial duties (chapter 6).

By sharing autonomy between the human operator and the robot, the robot is able to work in non-structured environments much more robustly. Also, the human supervision and high-level decision making will enable the robot to handle a much more diverse set of parts, necessary for the customization of products. For the human operator, assistive robots improve the ergonomy, safety and efficiency of executing the task. The flexibility and adaptivity of our approach allows rapid acquisition of new virtual guides with intuitive teaching methods. This ease of deployment will make such technology more accessible for mid-size industries, where robots are currently not used as much as in large-scale assembly lines. The impact of this new wave of flexible and supportive robots will be to secure existing assembly jobs, create new jobs for developing such robots and marketing them worldwide. In the long term, we believe that achieving semi-autonomous robots that support humans in industrial environments will be an important, perhaps necessary, step forward to achieve *fully-autonomous* robots.

1.4 Related Works

Virtual guides, also known as virtual fixtures, are used to enforce virtual constraints on the movements of robots, in order to assist the user during a collaborative task. The type of assistance offered by these constraints can vary among the different virtual guide 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. *Virtual fixtures* have been first introduced by [Rosenberg,

1993]. In this work, 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 have 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.

To proceed with the related works we will cover and classify some relevant works based on two main questions:

- How do we *define* a virtual guide?
- How do we *create* a virtual guide?

For each question, we will compare these works with our implementation of the virtual guides, keeping the exhaustive classification given by [Bowyer et al., 2014] in mind.

1.4.1 How to define virtual guides

Teleoperation and Hands on Device approaches

A first classification on the virtual guides definition can be done in respect of the user interaction with the robot. In a *teleoperated* system 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] (Figure 1.5). Teleoperation 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. However, a key problem with teleoperation

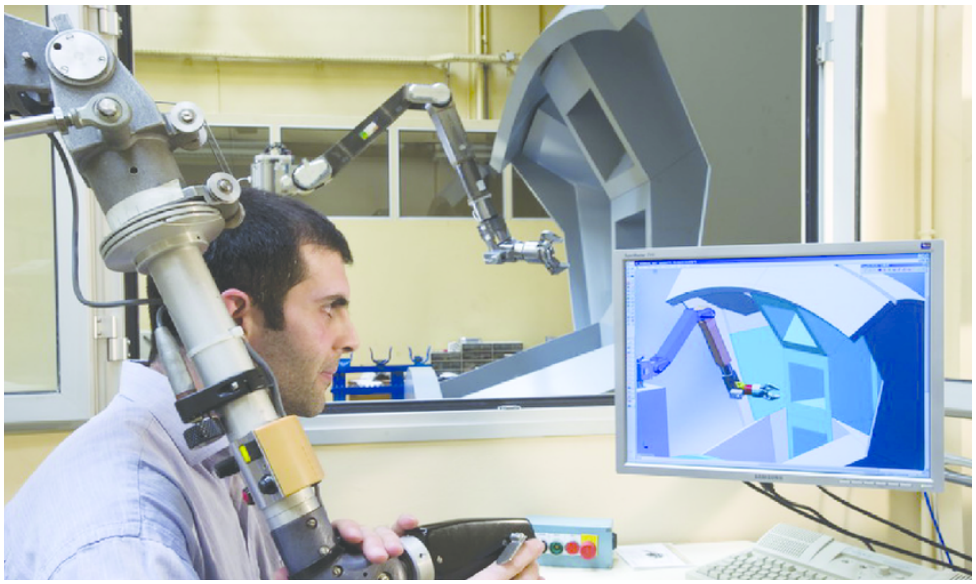


Figure 1.5: In [David et al., 2014] virtual guides are created on the fly into a physical engine, using linear interpolations. Since the guides are not programmed into the real robot controller, slight variations in their respective positions are possible. Also, in the context of co-manipulation tasks 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.

is that any information about the environment that is required by the user has to be interpreted at the slave robot, then transmitted to the master and finally presented to the user in some way.

With Hands-on interaction, the user directly interacts with the robot through physical contact [Sanchez Restrepo et al., 2017, Becker et al., 2013, Marayong et al., 2003, Pezzementi et al., 2007] (Figure 1.6). This allows for 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 workspace through a teleoperated robot. Another example of hands-on interaction is presented by [Dumora, 2014] in the context of big objects co-manipulation. A library of virtual guides was hard programmed on the robot during co-manipulation, 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.

Our work fall in the latter category [Raiola et al., 2015a], whereby the user directly manipulates the tool while the robot enforces a constraint on the possible tool movements.

Figure 1.6: RB3D collaborative robot. This image shows an example of *hands on* interaction since the user directly manipulates the tool through the robot.

Impedance and admittance constraint

While the teleoperation and hands on device differs on the way the user interacts with the system, the main distinction on the virtual guides definition can be done between guides defined as Impedance or Admittance constraints.

Although the desired effect of both is the same, an impedance constraint conceptually acts very differently to an admittance constraint. In an impedance constraint, the controller seeks to apply a force to the robot which nullifies, to some degree, the motion which violates a constraint. On the other hand, an admittance constraint acts like a filter on the robot's motion, meaning that only the components that does not violate the constraints are passed to the robot's actuators.

An implementation of impedance constraints is presented in [Joly and Andriot, 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 et al., 2007] where they are called "proxies". Implementations of admittance constraints are presented in [Marayong et al., 2003, Bettini et al., 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 with the impedance constraint. In our work we use the same impedance constraint defined in [Joly and Andriot, 1995], but we expand the concept to be able to handle multiple constraints in parallel.

Regional and guidance fixtures

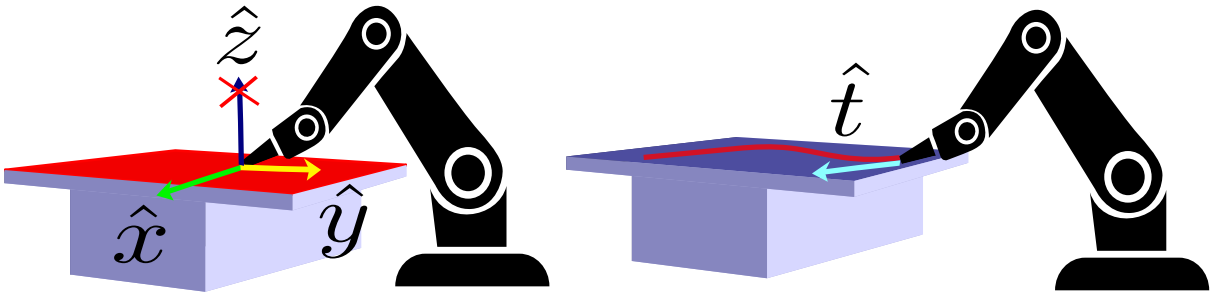


Figure 1.7: Left: Virtual fixture defined as regional constraint, in this example the table surface is the constraint which means that the robot can not move on the table's normal direction represented by $\hat{z}$. Right: Virtual fixture defined as a guidance constraint, in this case the robot can move along the curve (with tangent versor $\hat{t}$) but not away from it.

Most of the virtual fixtures have one of two purposes. "Regional" constraints (commonly called "forbidden region virtual fixtures" [Abbott and Okamura, 2003]) in which case they bound a manipulator's tool to certain regions within its task or joint space. "Guidance" constraints (commonly called "guidance virtual fixtures" [Marayong et al., 2003] or simply guides) allow the user to move the tool along a specific pathway or toward a specific target [Burghart et al., 1999, Bettini et al., 2004]. By their nature, guidance constraints are more intrusive upon the user than regional constraints; however, the ability to constrain positions precisely is important in many applications. Examples of these two constraint types are shown in Figure 1.7. While the definition of virtual fixtures given by [Joly and Andriot, 1995] is general and can be expanded to both the kind of constraints (regional and guidance), in our work we are interested in generating trajectories for the robot that can be used as *virtual guides*, thus our work fall in the latter category.

	Definition of virtual fixture in our work
Teleoperation vs Hands On	Hands On
Impedance vs Admittance	Impedance
Regional vs Guidance	Guidance

Table 1.1: Summary for our definition of virtual fixture.

1.4.2 How to create virtual guides

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 pre-defined 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.

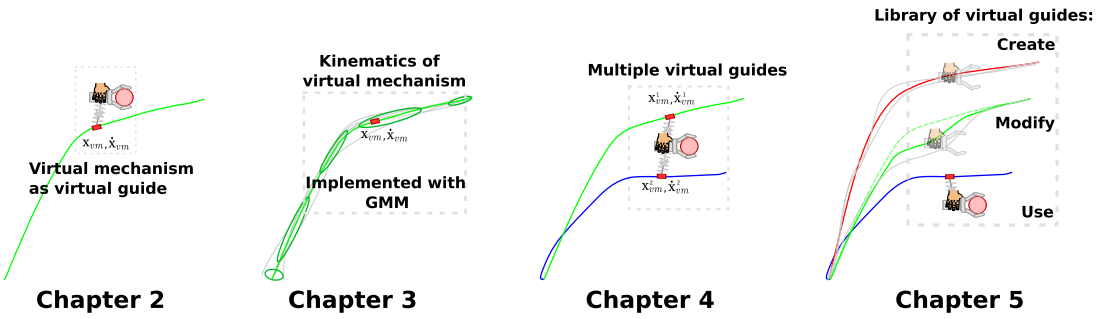
In our work, we use the demonstrations of the user to train Gaussian Mixture Models (GMM) as in [Calinon et al., 2007], which ensures smooth movements and explicitly models the variance in user demonstrations. Moreover this allows us to define one of the novel aspect of our work, i.e. the *probabilistic* 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 [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]. The use of GMM allows us to define virtual fixtures in a probabilistic way, enabling the possibility to activate or deactivate virtual fixtures based on the probability to be selected. So far, probabilistic methods such as HMM (Hidden Markov Models) and SVM (Support Vector Machine) has been used to switch virtual fixtures on/off as in [Li and Okamura, 2003, Aarno et al., 2005, Yu et al., 2005]; this is done princi-

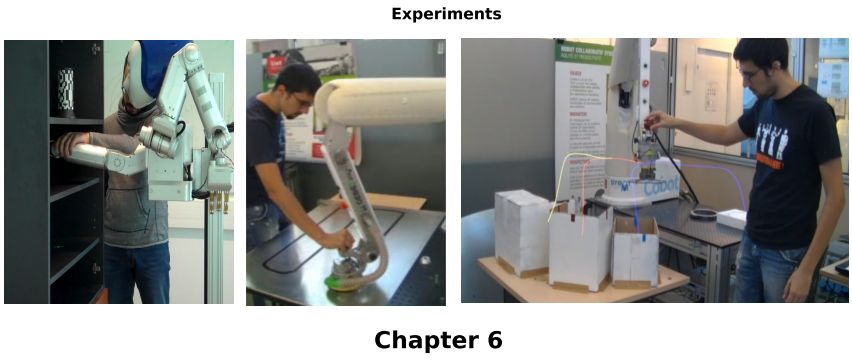
pally to avoid obstacles, while in [Nolin et al., 2003], a distance threshold is used to disable the guide. A second advantage is that the probabilistic approach allows us to simultaneously activate and recognize several guides, by assigning probabilities to each guide based on user behavior. Thus, our method enables the use of a *library of guides*, with one guide for each distinct tasks. 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 a HMM to probabilistically choose a guide in a sequence of linear guides to accomplish a pick and place task. Finally, unlike many of previous works, we use virtual guides in a co-manipulation framework instead of teleoperation; in this aspect our work can be compared to [Ben Amor et al., 2014, Hernández et al., 2012, Rozo et al., 2016] where the user and the robot 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, Sanchez Restrepo et al., 2017] where it is exploited the concept of *Task refinement*, as we will see in chapter 3 this is possible thanks to the incremental training of GMM [Calinon, 2007].

1.5 Outline

The thesis is structured as follows:

- In chapter 2 we introduce a possible definition of virtual guides by using virtual mechanisms as proposed in [Joly and Andriot, 1995], which forms the basis of our work.
- We explain how to create the virtual mechanisms kinematics using Gaussian Mixture Models (GMM) in chapter 3.
- In chapter 4 we describe the control framework for multiple virtual guides and we prove the stability of the proposed controller.
- In chapter 5 we define the concept of library of virtual guides, which allow the user to create, modify and use multiple guides.
- We evaluate our approach with three different experiments in chapter 6.
- Final conclusions and perspective are presented in chapter 7.





Chapter 2

Virtual Mechanism as Virtual Guide

This chapter details the theory and implementation of a Virtual Guide through the use of a Virtual Mechanism. In section 2.1 the general definition of Virtual Mechanism and the force interactions that it exchanges with the robot are presented. In section 2.2 the passivity of the Virtual Mechanism is proven by analyzing the energy dissipated on it. The section 2.3 discusses the problem related to the kinematics singularities of the Virtual Mechanism caused by the unpredictability of the user demonstrations and two possible solutions based on dynamic damping and Jacobian normalization.

2.1 Definition of Virtual Mechanism

In this work, a Virtual Guide is realized as a connection between the end-effector of the robot and a simulated virtual robot called Virtual Mechanism (VM) [Joly and Andriot, 1995]. In general the virtual robot has less degrees of freedom than the real one thus the movements of the real robot are constrained by the possible movements of the virtual robot (Figure 2.1). Originally, virtual mechanisms have been introduced at CEA[1] to safely teleoperate robots for the maintenance of nuclear sites. A common application in a nuclear site is the remote inspection of pipes. In this kind of task, it is necessary to avoid collisions between a robotic camera and the pipe, this can be ensured by constraining the camera movements along the pipe by

[1] Commissariat à l'énergie atomique et aux énergies alternatives

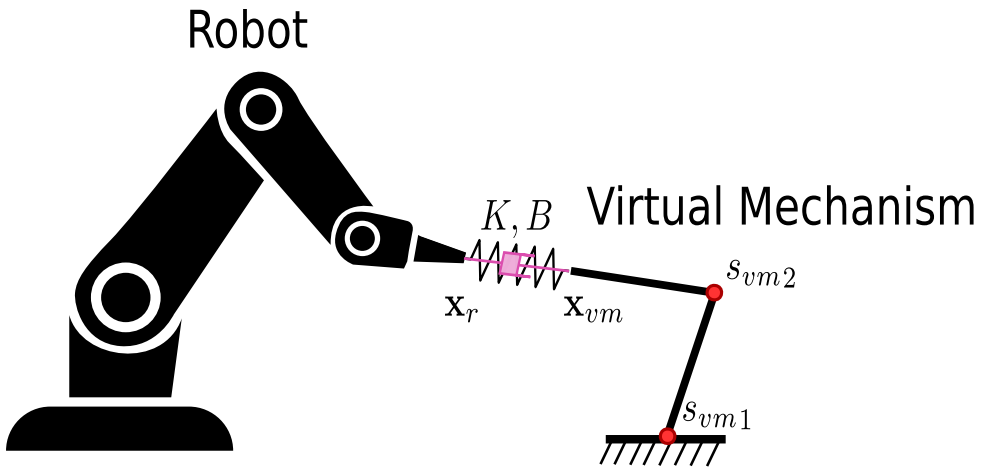


Figure 2.1: Virtual connection between the robot and the virtual mechanism. $\mathbf{x}_{\text{vm}}$ and $\mathbf{x}_{\text{r}}$ represent respectively the end-effector position in Cartesian space of the virtual mechanism and the robot. The s_{vmi} represent the degrees of freedom of the virtual mechanism. By connecting the robot to the virtual mechanism through the spring-damper system, the robot's movements are limited by the mechanism's movements. For example, with a 2 degrees of freedom mechanism, the robot can only move in a plane.

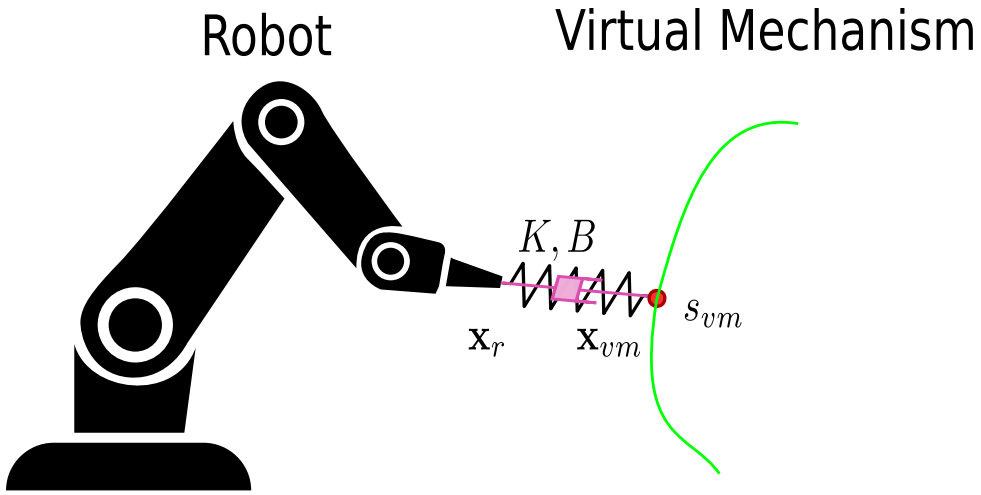


Figure 2.2: In our work, the virtual mechanism has only one degree of freedom represented by s_{vm}. The green trajectory represents the possible configurations of the virtual mechanism in the Cartesian space $\mathbf{x}_{\text{vm}}$, and because of the connection with the spring-damper system, it represents the only allowed configurations of the robot tool-tip $\mathbf{x}_{\text{r}}$.

modelling the surface through a cylindrical virtual mechanism [Joly and Andriot, 1995]. In our work instead, we use the virtual mechanisms in a co-manipulation framework where the user is directly in contact with the robot (Figure 2.3). Furthermore, we are interested in one dimensional constraints; i.e. the VM can be thought as a cart moving along a rail, with the rail acting as the constraint (Figure 2.2). The robot end-effector and the virtual *cart* mechanism

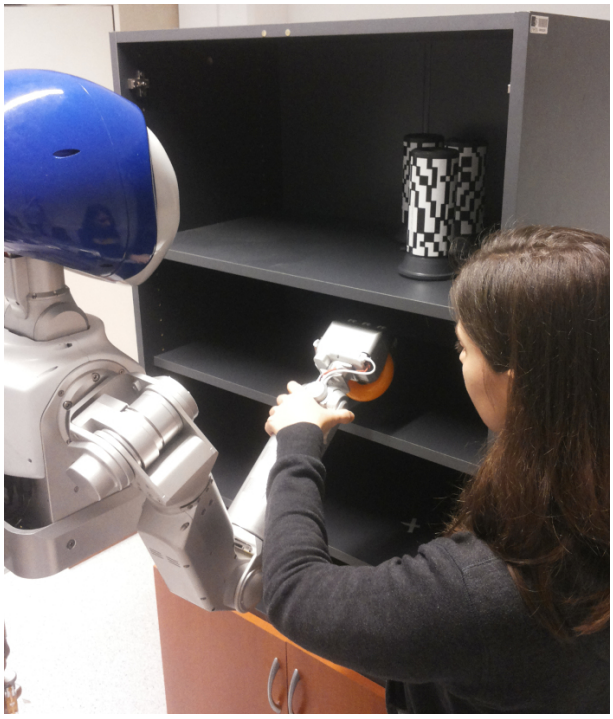


Figure 2.3: Image taken from the experiment conducted with the Meka robot at ENSTA-ParisTech (section 6.1). The task consisted in using virtual guides assistance to facilitate the placement of objects in a cupboard with shelves while avoiding collisions with them. Within the virtual guide formalization, it is important to distinguish between three participators: 1) the human operator, who exerts forces on the robot end-effector 2) the robot 3) the virtual mechanism, which constrains the movement of the robot.

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 (Figure 2.4).

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 Figure 2.5. The kinematics of the virtual mechanism is described by:

$$\mathbf{x}_{\text{vm}} = f(s_{\text{vm}}) \tag{2.1}$$

$$\dot{\mathbf{x}}_{\text{vm}} = \mathbf{J}_{\text{vm}}(s_{\text{vm}})\dot{s}_{\text{vm}} \tag{2.2}$$

Figure 2.4: The overall virtual guide scheme is reminiscent of Victorian waterway transport, when horses pulled boats along canals with ropes. The rider (human) leads the horse (the robot end-effector) to pull the boat (the virtual mechanism) along the canal (the guide) with the rope (the spring-damper system). The boat and rope constrain the horse so that it cannot walk away from the canal.

In chapter 3 we will describe how to implement the functions $f(s_{\text{vm}})$ and $\mathbf{J}_{\text{vm}}(s_{\text{vm}})$ through user's demonstrations.

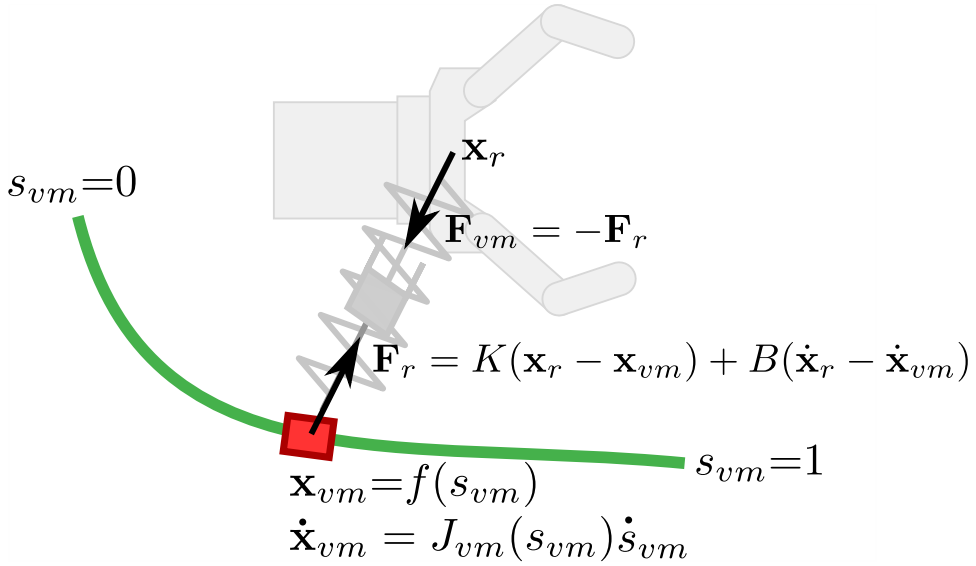


Figure 2.5: The main variables and equations of the virtual mechanism.

2.1.1 Force on the virtual mechanism

The virtual mechanism is connected to the robot end-effector with a virtual spring-damper system (PD Controller). The force applied to the virtual mechanism by the robot is:

$$\mathbf{F}_{\mathrm{r}} = K(\mathbf{x}_{\mathrm{r}} - \mathbf{x}_{\mathrm{vm}}) + B(\dot{\mathbf{x}}_{\mathrm{r}} - \dot{\mathbf{x}}_{\mathrm{vm}}). \tag{2.3}$$

The virtual mechanism is ideal, so the efforts applied on it are null

$$\mathbf{J}_{\mathrm{vm}}{}^{\mathsf{T}}\mathbf{F}_{\mathrm{r}} = 0, \tag{2.4}$$

which leads to

$$\mathbf{J}_{\mathrm{vm}}{}^{\mathsf{T}}(K(\mathbf{x}_{\mathrm{r}} - \mathbf{x}_{\mathrm{vm}}) + B(\dot{\mathbf{x}}_{\mathrm{r}} - \mathbf{J}_{\mathrm{vm}}\dot{s}_{\mathrm{vm}})) = 0. \tag{2.5}$$

By solving (2.5) 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:

$$\dot{s}_{\mathrm{vm}} = (\mathbf{J}_{\mathrm{vm}}{}^{\mathsf{T}}B\mathbf{J}_{\mathrm{vm}})^{-1}\mathbf{J}_{\mathrm{vm}}{}^{\mathsf{T}}(K(\mathbf{x}_{\mathrm{r}} - \mathbf{x}_{\mathrm{vm}}) + B\dot{\mathbf{x}}_{\mathrm{r}}). \tag{2.6}$$

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 (2.6).

2.1.2 Force on the robot end-effector

Since 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.

$$\mathbf{F}_{\mathrm{vm}} = -\mathbf{F}_{\mathrm{r}} = K(\mathbf{x}_{\mathrm{vm}} - \mathbf{x}_{\mathrm{r}}) + B(\dot{\mathbf{x}}_{\mathrm{vm}} - \dot{\mathbf{x}}_{\mathrm{r}}). \tag{2.7}$$

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 $\mathbf{J}_r^\intercal$ to convert the forces into torque references for the motor controllers:

$$\tau_r = \mathbf{J}_r^\intercal \mathbf{F}_{vm}. \tag{2.8}$$

In Figure 2.6 the signals between the robot and the virtual mechanism are illustrated.

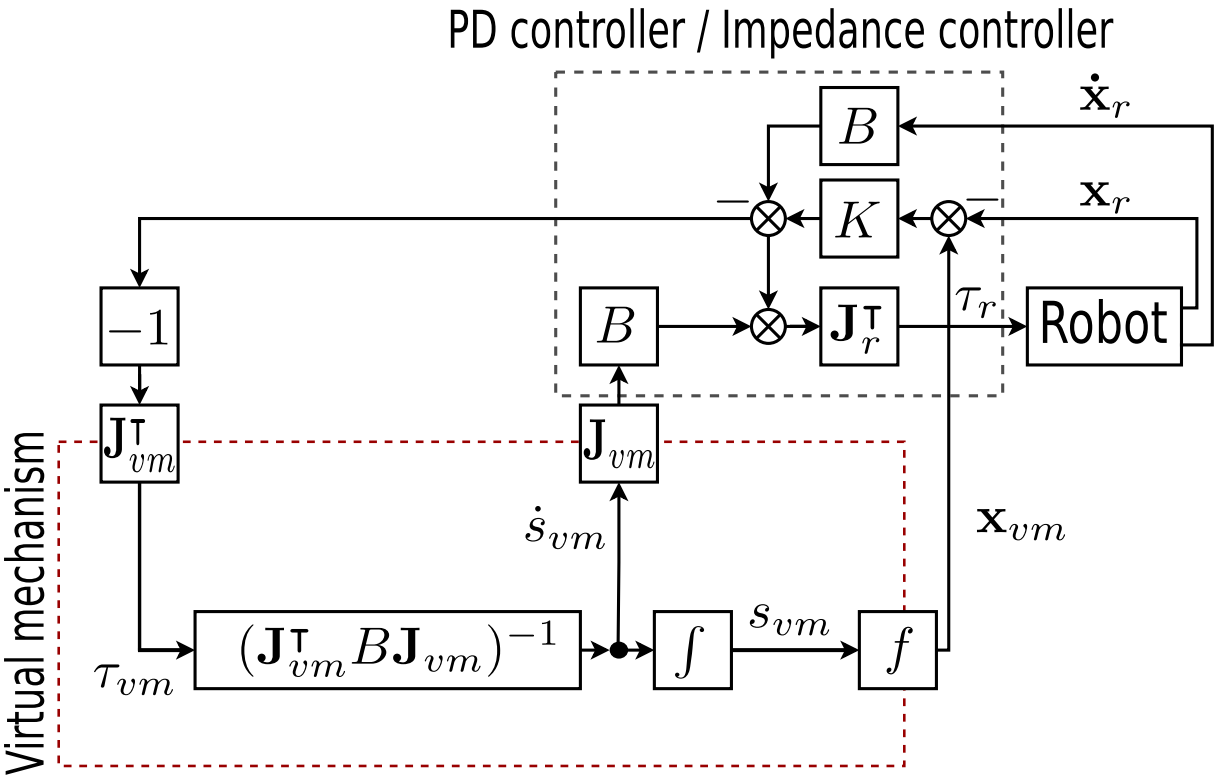


Figure 2.6: Control scheme for the virtual mechanism.

2.2 Passivity

In situations where robots share their workspace with humans, safety is of supreme importance. Most industrial robots in use today operate behind fences to keep people outside of the robot workspace. This is not possible in applications that involve cooperation or physical contacts between robots and humans.

The *safety* of a robotic system is mainly enforced through three different factors [Van Damme et al., 2010]:

- *Safe element design* in the form of padded and rounded link shapes and low weight components. The low weight helps to reduce eventual collision forces while the shape of the robot helps to avoid injuries caused by accidental contacts with the robot.

- *Passive compliance* devices such as torque/velocity limitation devices, elastic actuators [Pratt and Williamson, 1995] or variable stiffness actuators [Tonietti et al., 2005]. They allow the robot to be more compliant (i.e. less stiff) and consequently more safe because in case of contacts with the environment or a person, part of the energy is absorbed by the robot.

- *Active compliance*. The compliance can be ensured by the way the robot is controlled. There are two main kind of *compliant* controllers: Force and Impedance controllers [Siciliano et al., 2009].

In our formulation we control the robot through a spring-damper system (2.7) meaning that the virtual guide acts as an impedance controller for the robot. This controller generates forces for the robot end-effector based on the mechanism position and velocity. In this section, we are interested in studying the passivity of this controller to assure that it does not lead to the instability of the robotic system [Khalil and Grizzle, 1996]. This represents an important step to prove that the co-manipulation with virtual guides is *safe*.

In [Joly and Andriot, 1995] the passivity for the controller is proven by studying its dissipated energy. A system is considered to be passive when it does not provide more energy than it has received. Intuitively our controller is passive because it is realized with passive mechanical components such as springs and dampers.

Considering the system in Figure 2.6, the supplied power of the controller is:

$$P = \tau_{\text{vm}}^{\intercal} \dot{s}_{\text{vm}} - \tau_{\text{r}}^{\intercal} \dot{q}_{\text{r}}, \tag{2.9}$$

$$= \mathbf{F}_{\text{vm}}^{\intercal} \dot{\mathbf{x}}_{\text{vm}} - \mathbf{F}_{\text{vm}}^{\intercal} \dot{\mathbf{x}}_{\text{r}}. \tag{2.10}$$

Where $\dot{q}_{\text{r}}$ represents the joint velocities of the robot.

$$P = \mathbf{F}_{\text{vm}}^{\intercal} (\dot{\mathbf{x}}_{\text{vm}} - \dot{\mathbf{x}}_{\text{r}}), \tag{2.11}$$

$$= (K(\mathbf{x}_{\text{vm}} - \mathbf{x}_{\text{r}}))^{\intercal} (\dot{\mathbf{x}}_{\text{vm}} - \dot{\mathbf{x}}_{\text{r}}) + (\dot{\mathbf{x}}_{\text{vm}} - \dot{\mathbf{x}}_{\text{r}})^{\intercal} B (\dot{\mathbf{x}}_{\text{vm}} - \dot{\mathbf{x}}_{\text{r}}). \tag{2.12}$$

By integrating both the side of the equation we obtain the energy balance:

$$\underbrace{\int_0^t P\,dt}_{\text{supplied energy}} = \underbrace{E_{pot}(t) - E_{pot}(0)}_{\text{stored spring energy}} + \underbrace{\int_0^t (\dot{\mathbf{x}}_{\text{vm}} - \dot{\mathbf{x}}_{\text{r}})^{\intercal} B (\dot{\mathbf{x}}_{\text{vm}} - \dot{\mathbf{x}}_{\text{r}})\,dt}_{\text{dissipated energy}}. \tag{2.13}$$

In this equation E_{pot} represents the potential energy associated to the spring K. The controller is passive because E_{pot} and the dissipated energy are positive (B is positive definite):

$$\int_0^t P\,dt >= -E_{pot}(0). \tag{2.14}$$

This equation shows that the controller can not supply more energy than the initial one. Moreover since $\mathbf{J}_{\text{vm}}{}^{\intercal} \mathbf{F}_{\text{r}} = 0$ for (2.4):

$$\int_0^t \tau_{\text{r}}^{\intercal} (-\dot{q}_{\text{r}})\,dt >= -E_{pot}(0). \tag{2.15}$$

This proves the passivity of the controller. Regarding the mechanism's gains K and B, 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. Though, the gains specification for the virtual mechanism is independent from the stability of the robot.

2.3 Kinematic Singularities

By looking at the dynamical system (2.6), it is clear that the term $(\mathbf{J}_{\text{vm}}{}^{\intercal} B \mathbf{J}_{\text{vm}})$ has to be invertible in order to avoid singularities on the mechanism evolution. This translates in ensuring that the Jacobian $\mathbf{J}_{\text{vm}}(s_{\text{vm}})$ doesn't nullify for any value of the phase s_{vm}. This is not easy to guarantee because as we will see in chapter 3 the Jacobian $\mathbf{J}_{\text{vm}}$ is created through user demonstrations. To overcome this problem, we propose two solutions:

- To use an adaptive *damping* on the virtual mechanism.
- To *normalize* the kinematics of the virtual mechanism.

2.3.1 Damping

This first solution uses an extra damping on the mechanism defined in (2.6):

$$\dot{s}_{\text{vm}} = (\mathbf{J}_{\text{vm}}{}^{\intercal} B \mathbf{J}_{\text{vm}} + B_d)^{-1} \mathbf{J}_{\text{vm}}{}^{\intercal} (K(\mathbf{x}_{\text{r}} - \mathbf{x}_{\text{vm}}) + B\dot{\mathbf{x}}_{\text{r}}). \tag{2.16}$$

Where B_d represents a damping diagonal matrix with positive values. To have a constant damping effect on the virtual mechanism, we can define B_d as a constant matrix, otherwise to have the damping affecting the mechanism only when close to a singularity, we can scale its values based on the proximity to the singular configurations. To do so, we can define a relationship between the damping and the determinant of the norm of the Jacobian $\det(\mathbf{J}_{\text{vm}}{}^{\intercal} \mathbf{J}_{\text{vm}})$. In this case, the damping matrix becomes:

$$B_d = B_{d_{\max}} \exp\left(-\frac{4}{\gamma} \det(\mathbf{J}_{\text{vm}}{}^{\intercal} \mathbf{J}_{\text{vm}}) \right), \tag{2.17}$$

Where $B_{d_{\max}}$ is a diagonal matrix which defines the maximum damping when the determinant of $\mathbf{J}_{\text{vm}}{}^{\intercal}\mathbf{J}_{\text{vm}}$ is zero, γ defines the maximum value of $\det(\mathbf{J}_{\text{vm}}{}^{\intercal}\mathbf{J}_{\text{vm}})$ above of what the damping is close to zero (in this case the mechanism is away from a singular configuration) [2]. This first solution solves the problem of the singularities at the price of introducing an extra damping on the guide. Another downside of that solution, is that Jacobian is not normalized, meaning that its norm is not constant. This translates in having a virtual guide where for certain values of s_{vm} the velocity module of the virtual mechanism is not determined only by the force applied on it but also by the velocities shown during the demonstrations. These variations could affect the user interaction with the virtual mechanism.

2.3.2 Normalization

Due to the limits of the solution proposed before, we can parametrize the kinematics equations of the virtual mechanism as functions of the arc-length l_{vm} instead of the phase s_{vm} which could depend on the sampling time of the demonstrations. The definition of l_{vm} is:

$$l_{\text{vm}}(s_{\text{vm}}) = \int_0^1 ||\frac{\partial \mathbf{x}_{\text{vm}}(s_{\text{vm}})}{\partial s_{\text{vm}}}||\mathrm{d}s_{\text{vm}},$$
$$\text{with } \frac{\partial \mathbf{x}_{\text{vm}}(s_{\text{vm}})}{\partial s_{\text{vm}}} = \mathbf{J}_{\text{vm}}. \tag{2.18}$$

Since it could be complicated to find a closed form solution for the integral above (since $\mathbf{J}_{\text{vm}}$ is learnt by demonstrations), we can approximate the arc-length values with a numeric method:

$$l_{\text{vm}k} = \sum_{i=0}^{k-1} ||\mathbf{x}_{\text{vm}i+1} - \mathbf{x}_{\text{vm}i}||. \tag{2.19}$$

Where $\{\mathbf{x}_{\text{vm}k}\}_{k=0}^{M}$ represents the demonstrated Cartesian positions. In order to normalize the arc-length we can divide each value $l_{\text{vm}k}$ with the total length of the curve, i.e. $l_{\text{vm}k}/l_{\text{vm}N}\ \forall\ k = 1..M$ [3]. The demonstration is now described by a vector containing the following $\{\mathbf{x}_{\text{vm}k}, l_{\text{vm}k}, s_{\text{vm}k}\}_{k=0}^{M}$. Finally we can interchange the phase s_{vm} with the arc-length l_{vm}

[2] Note that the value $\det(\mathbf{J}_{\text{vm}}{}^{\intercal}\mathbf{J}_{\text{vm}})$ is always positive.

[3] A larger number of points M ensures a better approximation in (2.19)

in the system (2.6). By doing so the kinematics of the Virtual Mechanism can be re-written as:

$$\mathbf{x}_{\text{vm}} = f(l_{\text{vm}}), \tag{2.20}$$

$$\dot{\mathbf{x}}_{\text{vm}} = \mathbf{J}_{\text{vm}}(l_{\text{vm}})\dot{l}_{\text{vm}}, \tag{2.21}$$

$$\text{with} \quad ||\mathbf{J}_{\text{vm}}(l_{\text{vm}})|| = \sqrt{\mathbf{J}_{\text{vm}}(l_{\text{vm}})^{\intercal}\mathbf{J}_{\text{vm}}(l_{\text{vm}})} = 1. \tag{2.22}$$

Another way to normalize the kinematics of the mechanism is through the use of an interpolation function between the phase s_{vm} and the arc-length l_{vm}. If we define this interpolation function as:

$$s_{\text{vm}} = s(l_{\text{vm}}) \tag{2.23}$$

$$\dot{s}_{\text{vm}} = \frac{\partial s_{\text{vm}}}{\partial l_{\text{vm}}}\dot{l}_{\text{vm}} \tag{2.24}$$

We can normalize the mechanism's Jacobian (2.2):

$$\dot{\mathbf{x}}_{\text{vm}} = \frac{\partial \mathbf{x}_{\text{vm}}}{\partial s_{\text{vm}}}\frac{\partial s_{\text{vm}}}{\partial l_{\text{vm}}}\dot{l}_{\text{vm}} = \frac{\partial \mathbf{x}_{\text{vm}}}{\partial l_{\text{vm}}}\dot{l}_{\text{vm}} = \mathbf{J}_{\text{vm}}(l_{\text{vm}})\dot{l}_{\text{vm}}, \tag{2.25}$$

$$\text{with} \quad ||\mathbf{J}_{\text{vm}}(l_{\text{vm}})|| = 1. \tag{2.26}$$

In our work we defined s as a cubic spline since it is fast to compute. Moreover its derivate (2.24) can be easily computed by decreasing the spline order from cubic to quadratic. The only downside of this approach is that in order to obtain a "good" normalization with the spline, it is necessary to choose a high number of interpolation points (i.e. knots), for example in our experiments we used approximatively 1000 points. An advantage of normalizing the kinematics through interpolation resides in the fact that this method is general; with this method we can normalize the mechanism's Jacobian independently from the learning algorithm used to define the kinematics in (2.1) and (2.2). This will be useful when we will use a probabilistic model to define the kinematics of the virtual mechanism (as we will see in section 3.3).

2.4 Conclusions

In this chapter, we presented how virtual guides can be created using the definition of virtual mechanism proposed by [Joly and Andriot, 1995]. This virtual mechanism works as an impedance controller for the robot, allowing the movements of the robots along the preferred directions and prohibiting the movements along the restricted directions. The passivity of the system is proven by studying the dissipated energy of the system. We introduced the problem related to the singularities of the mechanism's kinematics when these are defined by demonstrations and two possible solutions based on adaptive damping and Jacobian normalization. In the following chapter we will see how the Jacobian normalization can be applied when using a regression method such as Gaussian Mixture Regression (GMR) to describe the mechanism's kinematics.

Chapter 3

Kinematics of Virtual Mechanism

In order to create a Virtual Guide, we have to implement the kinematics equations of the virtual mechanism described by (2.1) and (2.2). There are two main ways to implement these equations:

- Through geometric modelling.
- Through demonstrations.

The first method was used by [Joly and Andriot, 1995]. This method can lead to a precise definition of the virtual guide's shape but requires the user to have a good understanding of the task in order to implement the kinematics of the virtual mechanism. Moreover the geometric model has to be defined in the operational space of the robot; this operation is not always trivial since it could require a transformation from the task reference to the robot reference. An alternative approach would be to generate the kinematics with a set of demonstrations which could be provided by the user through kinesthetic teaching directly in the robot's workspace. This approach has been extensively used in robotics in the past years [Calinon et al., 2007, Tykal et al., 2016]. The kinesthetic teaching is an approach whereby a user physically guides the robot to perform the desired movements. These movements can be recorded in the Cartesian space and can be used to define the shape of the virtual guide. The advantage of this approach is that a non-expert user may specify new virtual guides through lead-through programming without any prior knowledge about the task or about the task's geometric model as in [Stulp et al.,

2014, Stulp et al., 2013]. In order to create the virtual guides from demonstrations, we used a probabilistic model to define the kinematics of the virtual mechanism (2.1) and (2.2). Because of its *probabilistic* nature, we call the so created system "Probabilistic Virtual Mechanism". The model used in our work to generate the probabilistic virtual mechanism is the Gaussian Mixture Model (GMM). A Gaussian Mixture Model encodes the demonstrated data in a mixture of Gaussian functions defined by their mean and covariance, thanks to them we can estimate the position and the velocity of the virtual mechanism by conditioning on the mechanism's phase s_{vm} through a Gaussian Mixture Regression (GMR).

In section 3.1 we will give a formal definition of *probabilistic virtual mechanism*, afterwards in section 3.2 we will define the Gaussian Mixture Model (GMM) and explain how to *train* it to encode the desired guide. Furthermore two different training methods will be discussed: A *batch* and an *incremental* training method (presented respectively in subsection 3.2.1 and subsection 3.2.2). Finally we will show in section 3.3 how to extract from the GMM the position (2.1) and the velocity of the virtual mechanism (2.2) by using the Gaussian Mixture Regression (GMR).

3.1 Probabilistic Virtual Mechanisms

We define a "Probabilistic Virtual Mechanism" to be a VM which kinematics are defined through a probabilistic model. Every time the user moves the robot, the force applied by the robot on the mechanism is translated in a new value of the phase s_{vm} by integrating the dynamical system defined in (2.6). To determine the new Cartesian position of the virtual mechanism $\mathbf{x}_{\text{vm}}$, we can use the probabilistic model to estimate the *expected* position for a given value of the phase $s_{\text{vm}} = s$. Moreover, since the probabilistic model is created by demonstrations, we can estimate what is the *reliability* of the mechanism's position. In probability theory this translates in computing the following:

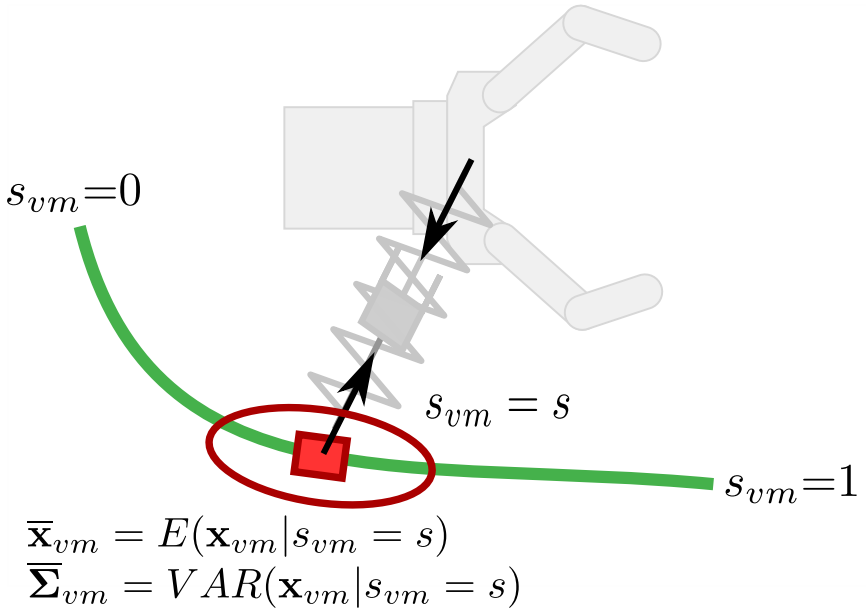


Figure 3.1: The main variables and equations of the probabilistic virtual mechanism. We can model the current state of the virtual mechanism as a multi-variate Gaussian distribution $\mathcal{N}(\overline{\mathbf{x}}_{\text{vm}}, \Sigma_{\text{vm}})$. This representation of the virtual mechanism is particularly useful when multiple probabilistic mechanisms are used in parallel.

$$\overline{\mathbf{x}}_{\text{vm}} = E(\mathbf{x}_{\text{vm}}|s_{\text{vm}} = s), \tag{3.1}$$

$$\Sigma_{\text{vm}} = VAR(\mathbf{x}_{\text{vm}}|s_{\text{vm}} = s), \tag{3.2}$$

where $\overline{\mathbf{x}}_{\text{vm}}$ and Σ_{vm} represent respectively the estimated mechanism position and the covariance matrix which encodes the uncertainty related to the estimation.

We can use the estimated position $\overline{\mathbf{x}}_{\text{vm}}$ as the current position for the virtual mechanism i.e. $\mathbf{x}_{\text{vm}} = \overline{\mathbf{x}}_{\text{vm}} = f(s_{\text{vm}})$ where $f(s_{\text{vm}})$ represents the kinematics for the virtual mechanism defined previously in (2.1). As we will see in chapter 4, equation (3.2) will be useful to determine which mechanism is responsible for the task execution when multiple mechanisms are active in parallel. Furthermore, since this information encodes the variability in the user demonstrations, we could think to use that uncertainty to scale the force exerted by the mechanism on the robot (subsection 7.1.1).

In the following sections of this chapter, we will describe how to create the probabilistic

model through GMM by exploiting two different training methods (batch and incremental). After the GMM has been created using one of these two methods, the GMR is used to obtain the conditioned mean that specifies the mechanism position (3.1) and the related uncertainty (3.2).

3.2 Gaussian Mixture Models

With GMM, the demonstrated data is modelled by a mixture of K components defined by a probability density function:

$$p(\zeta_m) = \sum_{k=1}^{K} p(k)p(\zeta_m|k), \tag{3.3}$$

where $\{\zeta_m\}_{m=1}^{M}$ represents the demonstrated set of Cartesian points of dimension D, $p(k)$ is the prior and $p(\zeta_m|k)$ is the conditional probability given by the Gaussian distribution (3.5). 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[1] in ζ. For a mixture of K components of dimensionality D the parameters in (3.3) are defined as:

$$\begin{aligned} p(k) &= \pi_k, \\ p(\zeta_m|k) &= \mathcal{N}(\zeta_m; \mu_k, \Sigma_k), \qquad (3.4) \\ &= \frac{e^{\left(-\frac{1}{2}(\zeta_m-\mu_k)^{\mathsf{T}}\Sigma_k{}^{-1}(\zeta_m-\mu_k)\right)}}{\sqrt{(2\pi)^D|\Sigma_k|}}. \qquad (3.5) \end{aligned}$$

The log-likelihood of the model described by θ, given a set of M datapoints $\{\zeta_m\}_{m=1}^{M}$ is:

$$\mathscr{L}(\theta) = \frac{1}{M}\sum_{m=1}^{M} \ln(p(\zeta_j)), \tag{3.6}$$

[1] Note that M is not strictly necessary to describe the model but it will be useful for the incremental training.

where $p(\zeta_m)$ is the probability that ζ_m has been generated by the model, which is computed using (3.3). Training of GMM is performed using the Expectation-Maximization algorithm (EM) which adjusts the priors π_k and the parameters μ_k and Σ_k of the Gaussian functions until a stop criterion is met. This algorithm guarantees monotone increase of the likelihood of the training set during optimization.

Next we will introduce two methods to train the GMM from demonstrations, a batch and an incremental method. These two methods rely on two different definitions of the EM algorithm. The first definition of EM uses all the data at once to create a GMM, no further updates of the model are possible. The second definition instead, allows to incrementally update the GMM every time new data is available.

3.2.1 Batch Training Method

With the Batch method, the user provides several examples of the tasks as demonstrations. Each task is performed multiple times in order to reduce the error residing in the human gestures.

In order to realize the virtual guides with the batch method, the following steps are performed:

1. *Gather* Cartesian trajectories by user demonstrations.

2. Perform a *clustering* on the gathered trajectories using Dynamic Time Warping (DTW) as metric function.

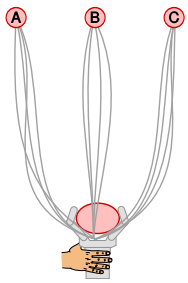
3. Use DTW to *align* the clustered trajectories to reduce their covariance.

4. *Fit* a Gaussian Mixture Model (GMM) on each cluster.

An experiment with the batch method and the Meka robot is reported in section 6.1. In Figure 3.7 are shown the steps performed with the batch method.

1 - Gather the trajectories

Each task is performed multiple times by the user through kinesthetic teaching, i.e. the user holds the end-effector of the robot, and demonstrates by guiding it along the desired trajectory. The demonstrations are gathered without a specific order, in this way the number of tasks is not known a priori.



Each sample $[x(t_m), y(t_m), z(t_m)]_{m=1:M}$ in a trajectory is associated with a phase value $s(t_m) = (t_m - t_1)/(t_M - t_1)$, i.e. $s(t_1) = 0$ at the beginning of the trajectory, and $s(t_M) = 1$ at the end. This phase represents a normalized version of the recording time. As introduced in subsection 2.3.2 we could alternatively use the arc-length instead of the phase s_{vm}. Both are valid alternatives to train a GMM. The reason why we introduced the arc-length l_{vm} is to guarantee a normalized virtual mechanism's Jacobian, but unfortunately we can not guarantee that during the GMM training, and a supplementary step is required when using GMR, see subsection 3.3.1.

As example, in Figure 3.2 (Left) are shown the trajectories gathered to reach the shelves of a cupboard with the Meka robot.

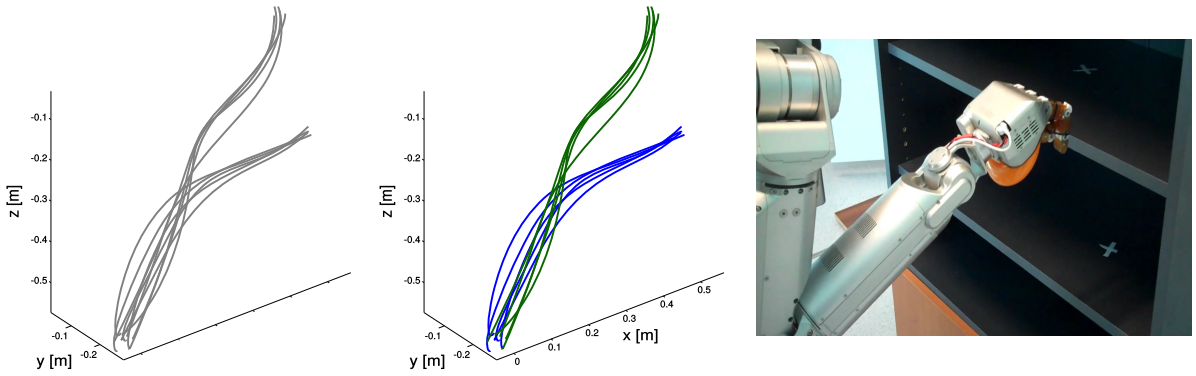
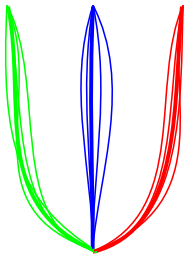


Figure 3.2: Left: Trajectories gathered to perform the experiment with Meka, these trajectories represent the movements needed to reach the two shelves in the cupboard. Center: Clustered trajectories. Right: Meka's arm and shelves.

2 - Clustering

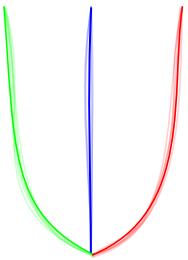
In this step a clustering is performed on the gathered trajectories to retrain the different tasks, see the example in Figure 3.2 (Center).



Due to different time length of each demonstration (even inside a single task) the trajectories can not be compared with the standard Euclidean distance. To avoid this problem, the trajectories are compared using the Dynamic Time Warping algorithm, which provides a distance between two trajectories that is not affected by time. After that, these distances are used to perform a hierarchical clustering [Maimon and Rokach, 2005].

3 - Alignment

The demonstrated trajectories can present a very different time length due the difficulty for the user to keep repeating the same motion over different demonstrations. This means that the time length of the demonstrations can change over the several repetitions; this generates undesired variance in the data.



To this end, we use DTW to reshape the trajectories in respect of time [Vakanski et al., 2012].

DTW algorithm

Dynamic time warping (DTW) is an algorithm developed originally for speech recognition [Sakoe and Chiba, 1978] that aims at aligning two sequences by warping their time indices iteratively until an optimal match between the two is found. Given two sequences expressed as $A = a_1, a_2, ..., a_n$ and $B = b_1, b_2, ...b_m$, they can be arranged on the sides of a grid as shown in Figure 3.3.

Each cell contains the distance between the corresponding elements of the two sequences[2].

[2]This can be easily generalized to multi dimensional sequences by computing the euclidean distance between the points.

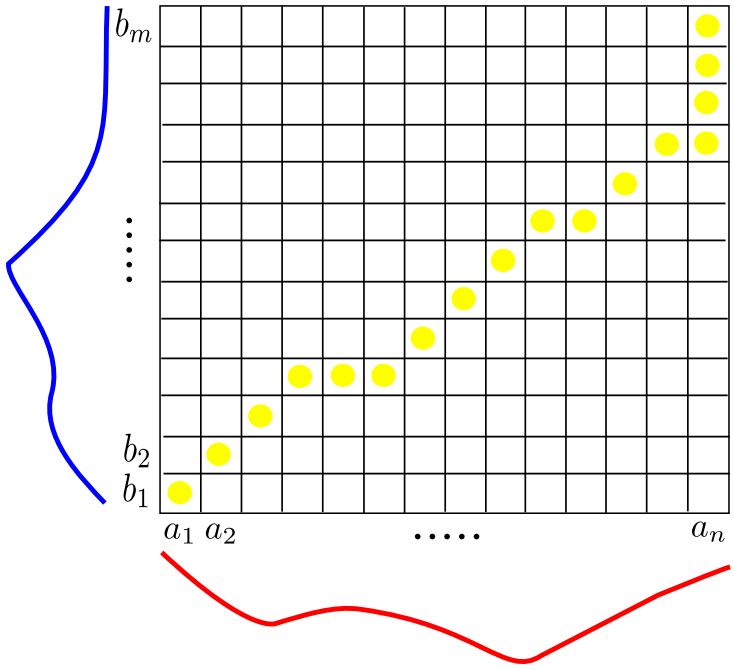


Figure 3.3: The yellow dots represents the path minimizing the distance between the two sequences (represented by the red and blue lines).

By using the dynamic programming optimization algorithm above the elements of the grid, it is possible to find a path on the grid that minimizes the total distance d between the two sequences. This corresponds to associate to each sample of a sequence one or more samples of the other.

We use DTW in a multivariate manner by taking in account all the trajectories' dimensions in order to perform a pairwise comparison among the demonstrated trajectories in each cluster i.e. every trajectory is compared against all the others trajectories belonging to the same cluster. Supposing we have N trajectories inside a cluster, we have $K = \frac{N!}{2(N-2)!}$ comparisons. As pointed out before, DTW can be used to find the total distance d between two trajectories. Thus for each trajectory we obtain a vector containing the respective distances from the others $[d_1, d_2, ..., d_K]$. We can define the distance of a trajectory from all the others as:

$$D = \sum_{i=0}^{K} d_i. \tag{3.7}$$

The trajectory with the lowest distance D is what we call the master trajectory. As explained before, with DTW the total distance d between two sequences is computed by summing the

distances on the optimal path. This path provides the optimal match between the samples of the sequences. We can use the indices of the optimal path to reorder the phase s_{vm} of each trajectory in respect of the master trajectory. In this way, the resulting trajectories will be closer to the master trajectory, reducing the entropy in the cluster see Figure 3.4 and Figure 3.5.

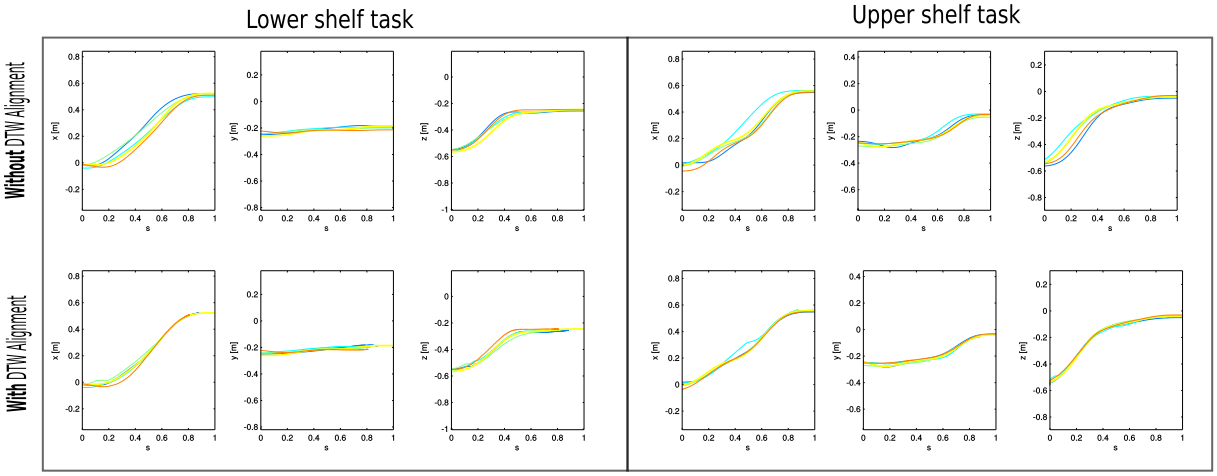


Figure 3.4: Meka experiment trajectories. Left: Trajectories for the lower shelf. Right: Trajectories for the upper shelf. We can see that the trajectories are spatially "squeezed" after the alignment.

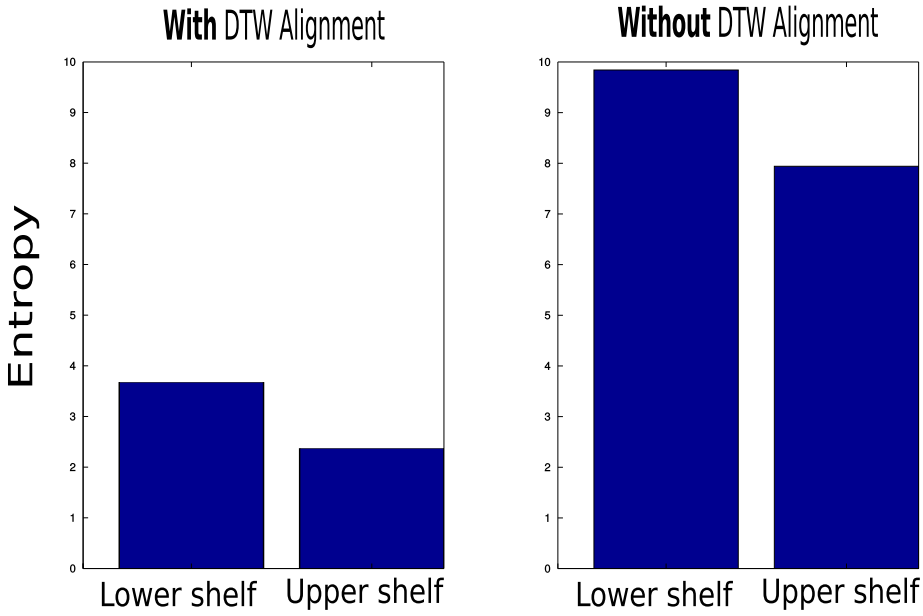


Figure 3.5: Total entropy computed for the Meka experiment. The two plots show the entropy before and after the DTW alignment. We compute the entropy for a GMM as: $H = \sum_{k=1}^{K} \frac{1}{2} ln(2\pi e)^d |\Sigma_{k,X}|$ where $\Sigma_{k,X}$ represent the sub-matrix of dimension dxd related to the spatial components (3.25). As we can see, with the DTW alignment the entropy is reduced, this indicates that alignment step is useful to reduce the spatial covariance of the model.

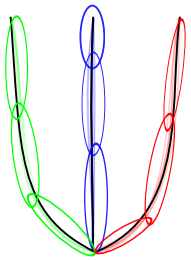
4 - Fit

The last step is to fit a Gaussian mixture model (GMM) to all the D-dimensional data points (Cartesian position *and* phase) in all the trajectories in each of the clusters. In a GMM, the data is represented by a weighted set of K multivariate Gaussians:

$$p(\zeta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\zeta; \mu_k, \Sigma_k), \text{ with } \sum_{k=1}^{K} \pi_k = 1. \tag{3.8}$$

Where π_k with $k = 1..K$ represent the prior probabilities associated to each Gaussian distribution. The most common way of fitting a GMM is using the Expectation-Maximization (EM) algorithm [Dempster et al., 1977].

Starting from an estimate of model parameters, soft membership of data is computed (the Expectation step) which is then used to update the parameters in the maximum likelihood (ML) manner (the Maximization step). This is repeated until convergence, which is theoretically guaranteed [Xu and Jordan, 1995].



In practice, initialization is frequently performed using the K-means clustering algorithm with the centroids corresponding to the means of the Gaussians, and the covariances estimated within each cluster by computing the covariance for all points belonging to it. Moreover the priors could be computed as a fraction of the data points allocated to each cluster.

EM algorithm

Starting from an initial estimation of the GMM with parameters $\{\pi_k^0, \mu_k^0, \Sigma_k^0\}_{k=1}^K$, at each step the following two steps are performed until a stop criterion is met:

E-step:

$$p_{k,m}^{t+1} = \pi_k^t \mathcal{N}(\zeta_m; \mu_k^t, \Sigma_k^t), \tag{3.9}$$

$$E_k^{t+1} = \sum_{m=1}^{M} p_{k,m}^{t+1}, \tag{3.10}$$

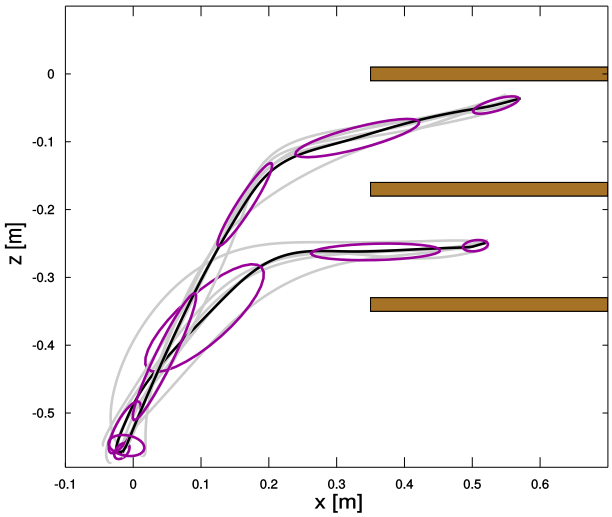


Figure 3.6: Gaussian mixture models for the clustered data from the Meka experiment. In our experiment we used 5 Gaussians. Training trajectories are light gray, the mean of the GMM is black. For visualization purposes, the GMM is projected on the xz-plane.

with E_k called cumulated posterior probability.

M-step:

$$\pi_k^{t+1} = \frac{E_k^{t+1}}{M}, \tag{3.11}$$

$$\mu_k^{t+1} = \frac{\sum_{m=1}^{M} p_{k,m}^{t+1} \zeta_m}{E_k^{t+1}}, \tag{3.12}$$

$$\Sigma_k^{t+1} = \frac{\sum_{m=1}^{M} p_{k,m}^{t+1} (\zeta_m - \mu_k^{t+1})(\zeta_m - \mu_k^{t+1})^\mathsf{T}}{E_k^{t+1}}. \tag{3.13}$$

The EM algorithm stops when $\frac{\mathscr{L}^{t+1}}{\mathscr{L}^t} - 1 < \mathscr{C}$ with the $\mathscr{L}$ defined in (3.6). The threshold $\mathscr{C} = 0.01$ is used in our case.

There are better methods to define the number of Gaussians using some criterions such as BIC (Bayesian Information Criterion) and AIK (Akaike's Information Criterion). In our work we preferred to keep the GMM training procedure simple by choosing manually the number of Gaussians to use, as for example in the experiment conducted with the Meka robot, see Figure 3.6. After the GMM has been created, a further step using Gaussian Mixture Regression (GMR) is needed to extract the GMM's mean which defines the estimated mechanism's position (section 3.3).

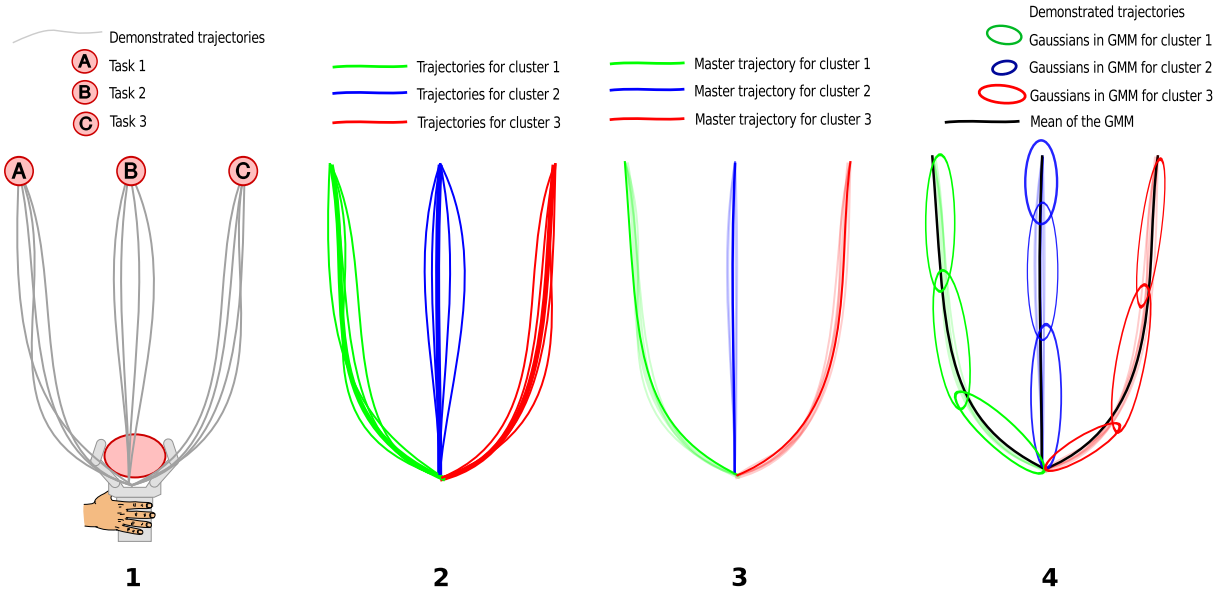


Figure 3.7: Summary for the batch method: 1 - Gather the trajectories: The user guides the robot to one of the different locations (A,B,C) multiple times. Each location is associated to a different task. The robot records the Cartesian positions of the demonstrations. 2 - Clustering: After demonstrating the tasks several times, the robot separates the resulting trajectories into distinct clusters. Each cluster represents a different task. 3 - Alignment: Trajectories after the alignment with DTW. The trajectories are warped in phase in order to maximize the match with the master trajectories represented with the thick lines. 4 - Fit: GMM created on the demonstrated trajectories.

3.2.2 Incremental Training Method

The batch training procedure is an established method in robotics for the GMM training [Calinon et al., 2007]. This procedure requires the user to demonstrate several times different trajectories associated to the desired tasks, meaning that the user has to move the robot repeatedly. Even with a robot with a good gravity compensation this is a tedious work due the presence of inertia and uncoordinated motions of individual joints. These problems decrease the intuitiveness and naturalness of the kinesthetic training and can affect the quality of the learned task. To this end we propose to incrementally train the GMM. After a first demonstration, a first GMM is created, this gives to the user the possibility to be assisted by the newly created virtual guide. Therefore, the user has the opportunity to progressively refine the virtual guide while the guide assists the user actions. As we will see, this is possible thanks to the *soft* interaction mode, which allows the user to escape an active guide in order to provide a new demonstration

subsection 5.2.2. Two main questions arise:

- How do we modify the EM algorithm to allow an incremental training of GMM?
- How do we determine if a new demonstrated trajectory belongs to an already existing model or represents a new one?

The first question does not have a trivial answer. By default, the EM algorithm does not allow an incremental training of GMM, since it needs the entire set of demonstrations to optimize the Gaussian Mixture. Solutions to incrementally train a GMM have been given by [Song and Wang, 2005] and [Arandjelovic and Cipolla, 2005] for online data stream clustering. In [Song and Wang, 2005] the authors suggested to create a new GMM by evaluating multiple GMMs and using the Bayesian Information Criterion (BIC) to select the optimal GMM. Thus given the new incoming data, they create a mixed model by merging the similar components of the old GMM with the new GMM. This algorithm is computationally expensive and tends to produce more components than the standard EM algorithm. In [Arandjelovic and Cipolla, 2005], the authors suggested to update the GMM components for the newly observed data with an *incremental EM* under the constraint of fixed complexity (i.e. the number of components of the GMM does not change) and smoothly varying data. Moreover, they select the number of Gaussian components with splitting and merging operations when the current number of Gaussian components does not represent well the new data. This second approach has been successfully used in [Calinon, 2007] to incrementally teach by imitation a set of basketball communication gestures to a humanoid robot. Moreover, in [Calinon, 2007], a statistical comparison between the classic and incremental EM algorithm is presented and detailed. Given the simplicity of the algorithm explained in [Arandjelovic and Cipolla, 2005] (compared to the method proposed by [Song and Wang, 2005]), we decided to use the same procedure to incrementally update the virtual guides. In a typical scenario, the user demonstrates a single trajectory, afterwards if no guide is available, a first GMM representing a new guide is created; successively the user can demonstrate new trajectories in order to update or create a new guide. This scenario opens the field for the second question: How do we determine if the new demonstrated trajectory belongs

to an already existing model or represents a new one? To answer this question, we propose to use the *relative likelihood* of the GMMs to discriminate between the "update guide" and the "new guide" case. Following we will present the incremental version of EM which forms the core for the incremental GMM estimation. Afterwards we will present an incremental clustering approach based on the relative likelihood.

Incremental GMM Estimation

The idea is to adapt the EM algorithm presented in section 3.2.1 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 paragraph). Thus, the model is first created using the EM algorithm presented in section 3.2.1. The EM algorithm converges after a certain number of iterations T and the GMM is completely defined by the set of parameters $\theta = \{\pi_k^T, \mu_k^T, \Sigma_k^T, M\}_{k=1}^{K}$. 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 $\{\tilde{\pi}_k^0, \tilde{\mu}_k^0, \tilde{\Sigma}_k^0, \tilde{E}_k^0\}_{k=1}^{K} = \{\pi_k^T, \mu_k^T, \Sigma_k^T, E_k^T\}_{k=1}^{K}$ with $\tilde{E}_k^0 = \tilde{\pi}_k^0 M$.

The EM algorithm can be rewritten as:

E-step:

$$\tilde{p}_{k,m}^{t+1} = \tilde{\pi}_k^t \mathcal{N}(\tilde{\zeta}_m; \tilde{\mu}_k^t, \tilde{\Sigma}_k^t), \tag{3.14}$$

$$\tilde{E}_k^{t+1} = \sum_{m=1}^{\tilde{M}} \tilde{p}_{k,m}^{t+1}. \tag{3.15}$$

M-step:

$$\tilde{\pi}_k^{t+1} = \frac{\tilde{E}_k^0 + \tilde{E}_k^{t+1}}{M + \tilde{M}}, \tag{3.16}$$

$$\tilde{\mu}_k^{t+1} = \frac{\tilde{E}_k^0 \tilde{\mu}_k^0 + \sum_{m=1}^{\tilde{M}} \tilde{p}_{k,m}^{t+1} \tilde{\zeta}_m}{\tilde{E}_k^0 + \tilde{E}_k^{t+1}}, \tag{3.17}$$

$$\tilde{\Sigma}_k^{t+1} = \frac{\tilde{E}_k^0 (\tilde{\Sigma}_k^0 + (\tilde{\mu}_k^0 - \tilde{\mu}_k^{t+1})(\tilde{\mu}_k^0 - \tilde{\mu}_k^{t+1})^\intercal)}{\tilde{E}_k^0 + \tilde{E}_k^{t+1}}, \tag{3.18}$$

$$+ \frac{\sum_{m=1}^{\tilde{M}} \tilde{p}_{k,m}^{t+1} (\tilde{\zeta}_m - \tilde{\mu}_k^{t+1})(\tilde{\zeta}_m - \tilde{\mu}_k^{t+1})^\intercal}{\tilde{E}_k^0 + \tilde{E}_k^{t+1}}. \tag{3.19}$$

The stop criterion is based on the log-likelihood $\mathscr{L}$. The iteration stops when $\frac{\mathscr{L}^{t+1}}{\mathscr{L}^t} - 1 < \mathscr{C}$ with the $\mathscr{L}$ defined as (3.6). An example showing the incremental training of GMM with simulated data[3] is presented in Figure 3.8. In Figure 3.9 is shown a comparison between the incremental and the batch estimation of GMM.

Incremental clustering

When the user demonstrates a new trajectory, we have to automatically detect if the new data belongs to one of the guides 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:

$$\hat{L}(\theta_n|\tilde{\zeta}) = \frac{L(\theta_n|\tilde{\zeta})}{L(\theta_{ML}|\tilde{\zeta})}, \ \forall\, n = 1..N, \tag{3.20}$$

[3]To gather the demonstrations we used a MATLAB GUI available at http://www.idiap.ch/software/pbdlib/

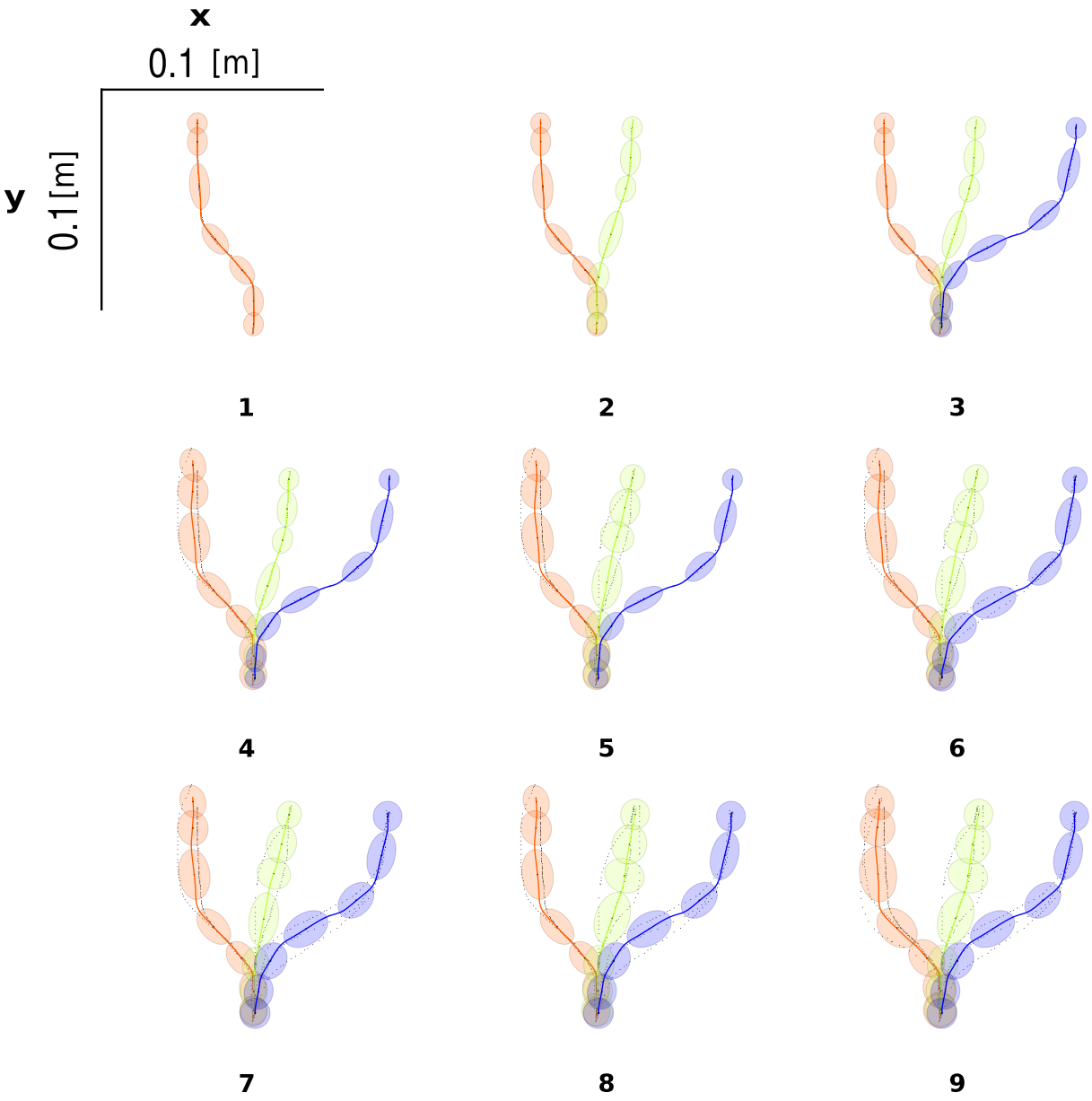


Figure 3.8: Incremental training with GMM. Three different models are incrementally trained with 3 different demonstrations. The last row shows the final models. For the incremental clustering the threshold value is set to $c = 0.3$

where $L(\theta_n|\tilde{\xi})$ represents the likelihood of an existing model r given the new demonstrated data $\tilde{\xi}$. 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 $\tilde{\xi}$ since is the same in each comparison. We can compute the relative likelihood $\hat{L}(\theta_n)$ for each existing

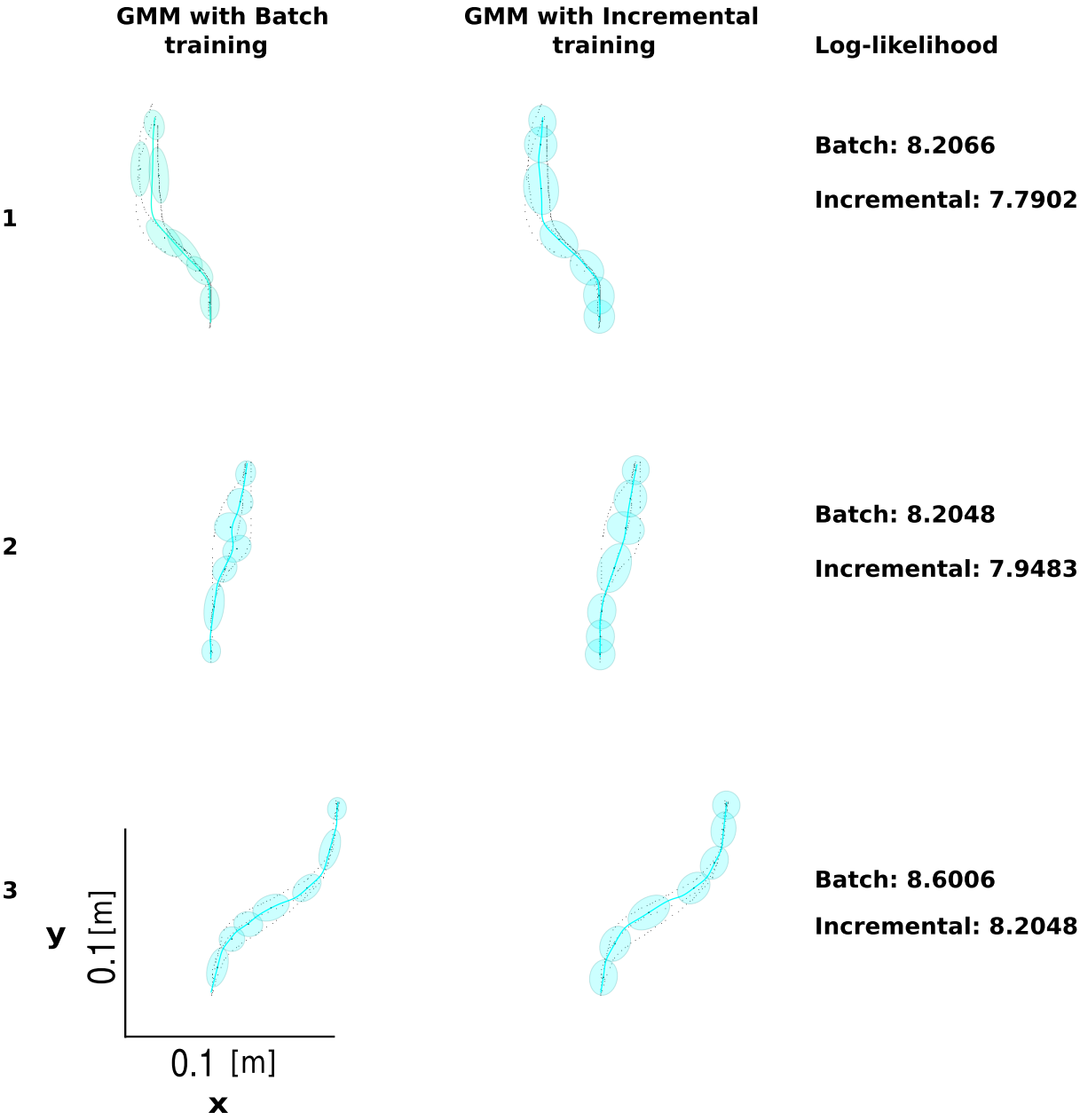


Figure 3.9: Comparison between the batch and incremental training method for GMM using the same demonstrations as in Figure 3.8. The image shows that the log-likelihoods for the batch method are only a little bit higher than for the incremental method. The resulting GMM representations are almost equivalent using the two methods. The loss of performance are negligible compared to the benefit induced by the incremental training of GMM.

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:

$$1 \geq \hat{L}(\theta_n) > c. \tag{3.21}$$

The threshold c can be selected arbitrarily. For example we could categorize the likelihood as:

$$1 \geq \hat{L}(\theta_n) > \frac{1}{3} \quad \theta_r \text{ very plausible,} \tag{3.22}$$

$$\frac{1}{3} \geq \hat{L}(\theta_n) > \frac{1}{10} \quad \theta_n \text{ plausible,} \tag{3.23}$$

$$\frac{1}{10} \geq \hat{L}(\theta_n) \geq 0 \quad \theta_n \text{ not plausible.} \tag{3.24}$$

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 (3.22). 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 (3.22), 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 thanks 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. To conclude, before a demonstration is used to incrementally update the selected GMM, an alignment step using DTW can be performed. This step is the same presented in section 3.2.1 with the difference that the master trajectory is given by the mean of the GMM selected to be updated.

3.3 Gaussian Mixture Regression

Virtual mechanisms require implementations of the kinematics equations $\mathbf{x}_{\text{vm}} = f(s_{\text{vm}})$ (2.1) and $\dot{\mathbf{x}}_{\text{vm}} = \mathbf{J}_{\text{vm}}(s_{\text{vm}})\dot{s}_{\text{vm}}$ (2.2). As anticipated in section 3.1 we can estimate the mechanism's position and its covariance matrix with (3.1) and (3.2). Both these functions can be computed through Gaussian mixture regression (GMR), based on the Gaussian mixture model (GMM) trained in the previous sections (subsection 3.2.1 and subsection 3.2.2).

In the context of a virtual mechanism, the input space is S, and the output space is X, corresponding to the phase s_{vm} and virtual mechanism position $\mathbf{x}_{\text{vm}}$ respectively. Given this partition, the mean and covariance matrix[4] are decomposed as

$$\mu_k = [\mu_{k,S}^\intercal, \mu_{k,X}^\intercal]^\intercal \text{ and } \Sigma_k = \begin{bmatrix} \Sigma_{k,S} & \Sigma_{k,SX} \\ \Sigma_{k,XS} & \Sigma_{k,X} \end{bmatrix}, \tag{3.25}$$

The implementation of $\mathbf{x}_{\text{vm}} = f(s_{\text{vm}})$ in (2.1) corresponds to computing $\bar{\mathbf{x}}_{\text{vm}} = E(\mathbf{x}_{\text{vm}}|s_{\text{vm}})$, i.e. the expectation of $\mathbf{x}_{\text{vm}}$ given the input s_{vm}:

$$\bar{\mathbf{x}}_{\text{vm}} = \sum_{k=1}^{K} \beta_k(s_{\text{vm}})(\mu_{k,X} + \Sigma_{k,XS}\Sigma_{k,S}^{-1}(s_{\text{vm}} - \mu_{k,S})), \tag{3.26}$$

with:

$$\beta_k(s_{\text{vm}}) = \frac{\pi_k g(\mathbf{x}; \mu_{k,S}, \Sigma_{k,S})}{\sum_{l=1}^{K} \pi_l g(\mathbf{x}; \mu_{l,S}, \Sigma_{l,S})} = \frac{\pi_k g(\mathbf{x}; s_{\text{vm}}^k)}{\sum_{l=1}^{K} \pi_l g(\mathbf{x}; s_{\text{vm}}^l)}. \tag{3.27}$$

The function g represents a Gaussian distribution defined as:

$$g(\mathbf{x}; \mu, \Sigma) = \frac{e^{\left(-\frac{1}{2}(\mathbf{x}-\mu)^\intercal \Sigma^{-1}(\mathbf{x}-\mu)\right)}}{\sqrt{(2\pi)^D |\Sigma|}}. \tag{3.28}$$

The function $\mathbf{J}_{\text{vm}}(s_{\text{vm}})$ in (2.2) is implemented with the analytical derivative of (3.26) in respect of s_{vm}:

$$\dot{\bar{\mathbf{x}}}_{\text{vm}} = \sum_{k=1}^{K} \dot{\beta}_k(s_{\text{vm}})(\mu_{k,X} + \Sigma_{k,XS}\Sigma_{k,S}^{-1}(s_{\text{vm}} - \mu_{k,S})) + \beta_k(s_{\text{vm}})(\Sigma_{k,XS}\Sigma_{k,S}^{-1}), \tag{3.29}$$

with:

[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.

$$\dot{\beta}_k(s_{\text{vm}}) = \frac{\pi_k \dot{g}(\mathbf{x}; s^k_{\text{vm}}) \sum_{l=1}^{K} \pi_l g(\mathbf{x}; s^l_{\text{vm}}) - \pi_k g(\mathbf{x}; s^k_{\text{vm}}) \sum_{l=1}^{K} \pi_l \dot{g}(\mathbf{x}; s^l_{\text{vm}})}{(\sum_{l=1}^{K} \pi_l g(\mathbf{x}; s^l_{\text{vm}}))^2}. \tag{3.30}$$

$\dot{g}$ can be easily computed thanks to the fact that s_{vm} is 1-dimensional:

$$\dot{g}(\mathbf{x}; s^k_{\text{vm}}) = -g(\mathbf{x}; s^k_{\text{vm}}) \Sigma^{-1}_{k,S}(s_{\text{vm}} - \mu_{k,S}). \tag{3.31}$$

Finally, Σ_{vm} is implemented by computing the conditional variance $VAR(\mathbf{x}_{\text{vm}}|s_{\text{vm}})$:

$$\Sigma_{\text{vm}} = \sum_{k=1}^{K} \beta_k(s_{\text{vm}})^2 \left(\Sigma_{k,X} - \Sigma_{k,XS} \Sigma^{-1}_{k,S} \Sigma^{\intercal}_{k,XS} \right). \tag{3.32}$$

3.3.1 GMR Normalization

As we explained in subsection 2.3.2, by using the phase s_{vm} as the independent variable for the kinematics equations (2.1) and (2.2), the Jacobian corresponds to the virtual mechanism's velocity which in our case is given by the derivate of GMR (3.29). Thus, the Jacobian is affected by the user demonstrations, meaning that the velocity of the virtual mechanism is variable and could be null. To avoid variations that could affect the user interaction with the virtual mechanism we use the interpolation defined in (2.23) and we substitute s_{vm} with l_{vm} in the mechanism equation (2.6).

To compute the position and velocity of the virtual mechanism we perform the following steps:

- First step:

  Compute the mechanism evolution through $\dot{l}_{\text{vm}}$:

  $$\dot{l}_{\text{vm}} = (\mathbf{J}_{\text{vm}}(l_{\text{vm}})^{\intercal} B \mathbf{J}_{\text{vm}}(l_{\text{vm}}))^{-1} \mathbf{J}_{\text{vm}}(l_{\text{vm}})^{\intercal} (K(\mathbf{x}_{\text{r}} - \mathbf{x}_{\text{vm}}) + B\dot{\mathbf{x}}_{\text{r}}). \tag{3.33}$$

  By integrating (3.33) we can compute l_{vm}.

- Second step:

  Transforming l_{vm} into s_{vm} by using $s_{\text{vm}} = s(l_{\text{vm}})$.

- Third step:

  Compute the mechanism position and velocity through GMR:

$$\mathbf{x}_{\text{vm}} = f(s(l_{\text{vm}})), \tag{3.34}$$

$$\dot{\mathbf{x}}_{\text{vm}} = \mathbf{J}_{\text{vm}}(l_{\text{vm}})\dot{l}_{\text{vm}}, \tag{3.35}$$

  with $\mathbf{J}_{\text{vm}}(l_{\text{vm}}) = \mathbf{J}_{\text{vm}}(s_{\text{vm}})\frac{\partial s}{\partial l_{\text{vm}}}$.

  For simplicity and clarity the phase s_{vm} will be used to define the independent variable of the virtual mechanism kinematics for the rest of the thesis.

3.4 Conclusions

In this chapter we explained how the kinematics of the virtual mechanism can be defined through probabilistic virtual guides based on Gaussian Mixture Models (GMM) and Gaussian Mixture Regression (GMR). In particular we defined two possible methods to train GMM, a batch method where the user provides all the demonstrations at once and an incremental method where the user can create a guide with a single demonstration and successively refine it by adding more demonstrations. Both the methods perform an unsupervised clustering on the demonstrations to detect the number of guides. With the batch method, the clustering is performed among all the demonstrations using DTW as metric, while with the incremental method we perform a clustering that takes advantage of the probability encoded in the GMMs in order to detect if a new demonstration belongs to an already existing guide. With both methods, after a demonstration has been associated with a cluster, an alignment step using DTW can be performed in order to reduce the entropy of the GMMs. Finally the kinematics equations (2.1,2.2)

are implemented by using the Gaussian Mixture Regression; given a specific phase value, it is possible to estimate the mean (expressed as space coordinates) and the covariance associated to that phase value. The mean value is used to realize the virtual guide on which the virtual mechanism evolves constraining the robot movements. The covariance instead, can be used to compute the probability of a virtual guide about being responsible for the task execution. We will present this important feature of the probabilistic virtual mechanisms in the next chapter (chapter 4).

Chapter 4

Multiple Virtual Mechanisms

In this chapter we present our main contribution: the possibility to use multiple guides in parallel. We now consider a control structure in which N guides are active in parallel, i.e. the robot end-effector is connected to multiple virtual mechanisms, as illustrated in Figure 4.1. This scenario applies when there are multiple tasks (e.g. transporting an object to one of multiple possible positions), and the robot does not know initially which task will be executed. In order to be able to use multiple guides, we need to define a control law that allows the user to switch between the guides, i.e. the user has to be able to select the guide to use by moving the robot end-effector on the desired guide, and at the same time we need to ensure that the selected guide correctly constrains the robot movement. Moreover we want to prove under which conditions the control law used to generate the multiple guides guarantees the stability of the whole system (mechanisms controller plus robot) [Raiola et al., 2015b].

4.1 Weighting scheme

Each of the N virtual mechanisms applies a force $\mathbf{F}^n_{\text{vm}}$ to the robot end-effector. Ideally, the resulting force $\mathbf{F}_{\text{res}} = \sum_{n=1}^{N} \mathbf{F}^n_{\text{vm}}$ brings the robot end-effector in a middle position between the virtual mechanisms. We need to define a $\mathbf{F}_{\text{res}}$ that brings the robot end-effector to "slide" along the desired guide by giving also the possibility to switch to another one when required. To this end we propose to *scale* the force of each VM with a weight p_n, determined by the relative

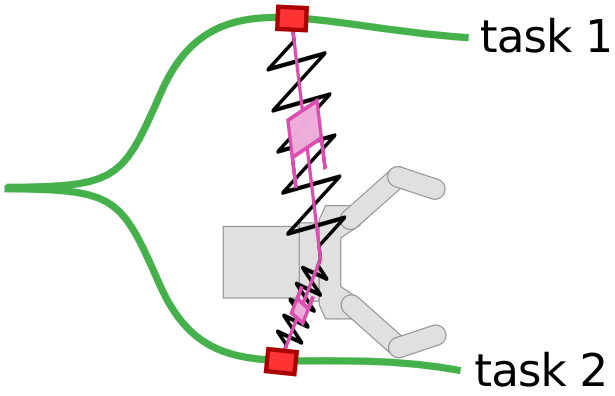


Figure 4.1: Multiple virtual mechanisms – one for each task – simultaneously connected to the robot end-effector.

distance with the robot end-effector $d = (\mathbf{x}_{\text{vm}} - \mathbf{x}_{\text{r}})$ so that the resultant force on the end-effector is:

$$\mathbf{F}_{\text{res}} = \sum_{n=1}^{N} p_n \cdot \mathbf{F}_{\text{vm}}^{n}. \tag{4.1}$$

Thus, the final force applied to the end-effector is a weighted sum of the forces from each guide, where the weights are determined by their distance from the robot end-effector. This weighting scheme is shown in the control diagram in Figure 4.2

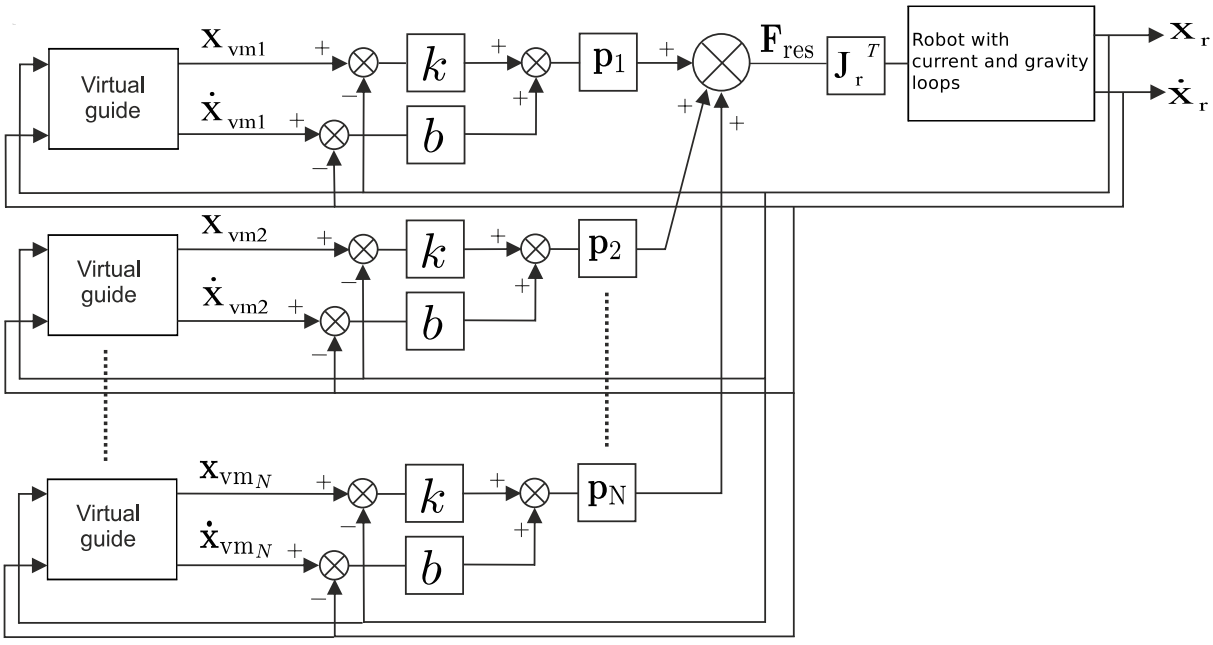


Figure 4.2: Control structure for multiple virtual guides.

The main question now is under which conditions on p_n this system is stable. This is studied in the next section. As we shall see, p_n must behave as a probability, that is $\sum_{n=1}^{N} p_n = 1$ and $\forall n, p_n \geq 0$. By using probabilities to compute the resulting force, we will prove that the robot converges on the most probable (i.e. closer) guide (Figure 4.3)

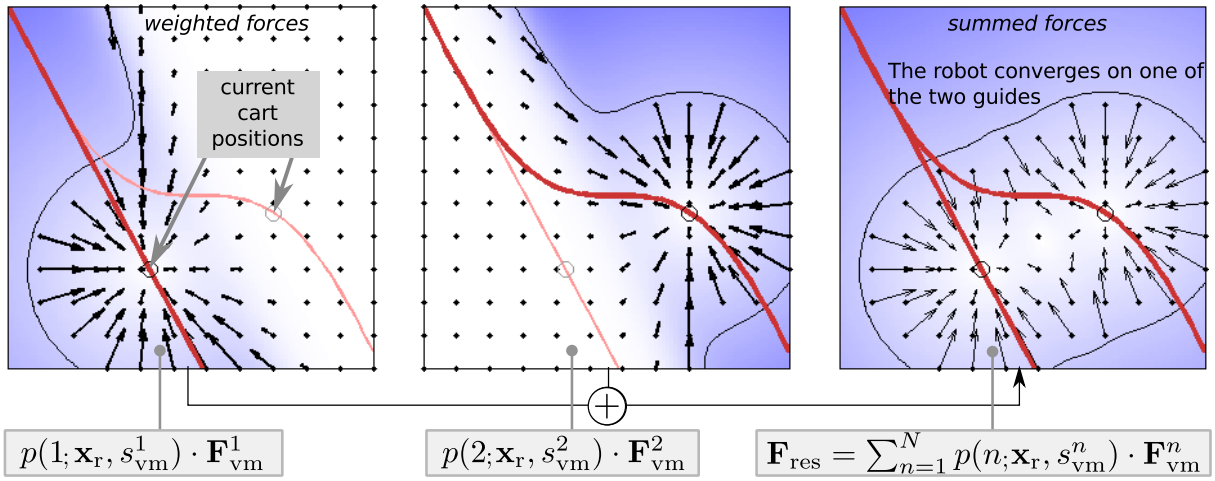


Figure 4.3: Example of force field generated by connecting the robot to two probabilistic virtual guides.

4.2 Stability Analysis

Our aim is now to prove the stability of using multiple guides in parallel. We do so with the Lyapunov direct method. This method has extensively used in robotics applications due the intrinsic nonlinearities of a high-dofs robots [Siciliano et al., 2009].

This method makes use of a Lyapunov function $V(x)$ which has an analogy to the potential function of physical dynamic systems.

Lyapunov direct method

Given a generic dynamical system $\dot{x} = f(x)$ with $f(0) = 0$, if $\exists V(x) : R^d \rightarrow R$:

- $V(0) = 0$
- $V(x) > 0 \;\; \forall\, x \in R^d - \{0\}$

- $\dot{V}(x) < 0 \ \ \forall\, x \in R^d - \{0\}$

- V is radial unbounded

The point 0[1] is an equilibrium globally and asymptotically stable. It is easier to visualize this method of analysis by thinking of a physical system (e.g. mass-spring-damper system) and considering its energy. If the system loses energy over time and it is never restored, the system could converge to a final resting state. This final state represents the equilibrium point of the system. The advantage of this method is the generality of its approach which allows to study the stability of a system independently from its nature.

The robot non linear dynamic model in joint space can be written as:

$$M(q_\mathrm{r})\ddot{q}_\mathrm{r} + C(q_\mathrm{r}, \dot{q}_\mathrm{r})\dot{q}_\mathrm{r} + F\dot{q}_\mathrm{r} + g(q_\mathrm{r}) = u, \tag{4.2}$$

where u represents the command torques, M the inertia matrix, C the Coriolis matrix, F is the viscous attrition, and g is the gravity term. Moreover we define the virtual mechanism's gain k and b as positive scalars[2]. We consider the Lyapunov function [Siciliano et al., 2009]

$$V = \tfrac{1}{2}\dot{q}_\mathrm{r}^\intercal M(q_\mathrm{r})\dot{q}_\mathrm{r} + \tfrac{1}{2}k\sum_{n=1}^{N} {d_n}^\intercal p_n d_n \ > \ 0, \ \forall\, \dot{q}_\mathrm{r}, d \neq 0, \tag{4.3}$$

where d_n is the error $d_n = \mathbf{x}_{\mathrm{vm}n} - \mathbf{x}_\mathrm{r}$, and p_n is the weight in (4.1). By differentiating (4.3) with respect to time we obtain:

$$\dot{V} = \dot{q}_\mathrm{r}^\intercal M(q_\mathrm{r})\ddot{q}_\mathrm{r} + \tfrac{1}{2}\dot{q}_\mathrm{r}^\intercal \dot{M}(q_\mathrm{r})\dot{q}_\mathrm{r} + k\sum_{n=1}^{N} \dot{d}_n^\intercal p_n d_n + D, \tag{4.4}$$

$$\text{with} \ \ D = \tfrac{1}{2}k\sum_{n=1}^{N} {d_n}^\intercal \dot{p}_n d_n. \tag{4.5}$$

[1] This result can be easily generalized to any other point x.

[2] The following demonstration can be generalized to matrix gains but we use scalar gains for sake of simplicity.

By substituting $M(q_r)\ddot{q}_r$ with the robot model, omitting the dependencies on $q_r, \dot{q}_r$ and rearranging the terms we acquire

$$\dot{V} = \tfrac{1}{2}\dot{q}_r^\intercal \left[\dot{M} - 2C\right] \dot{q}_r - \dot{q}_r^\intercal F \dot{q}_r + \dot{q}_r^\intercal \left[u - g\right] + k \sum_{n=1}^{N} \dot{d}_n^\intercal p_n d_n + D. \tag{4.6}$$

Due to the skew-symmetry property of the matrix $\dot{M} - 2C$ the term $\dot{q}_r^\intercal \left[\dot{M} - 2C\right] \dot{q}_r$ is null, and we simplify to

$$\dot{V} = -\dot{q}_r^\intercal F \dot{q}_r + \dot{q}_r^\intercal \left[u - g\right] + k \sum_{n=1}^{N} \dot{d}_n^\intercal p_n d_n + D. \tag{4.7}$$

4.2.1 Virtual mechanisms with fixed positions

We first study the case where all virtual mechanisms have a fixed position in the robot workspace, i.e. $\forall n, \dot{\mathbf{x}}_{\mathrm{vm},n} = 0$. This allows the simplification $\dot{d}_n = -\dot{\mathbf{x}}_r = -\mathbf{J}_r \dot{q}_r$, which leads to:

$$\dot{V} = -\dot{q}_r^\intercal F \dot{q}_r + \dot{q}_r^\intercal \left[u - g\right] - k \sum_{n=1}^{N} \dot{q}_r^\intercal \mathbf{J}_r^\intercal p_n d_n + D. \tag{4.8}$$

We choose a control input u that compensates the gravity term and introduces a proportional-derivative control: $u = g + \mathbf{J}_r^\intercal \sum_{n=1}^{N} p_n (k d_n - b \dot{\mathbf{x}}_r)$, where each term in the controller is weighted with its own probability p_n. By substituting the chosen control in (4.8) we obtain:

$$\dot{V} = -\dot{q}_r^\intercal (F + b) \dot{q}_r + D. \tag{4.9}$$

The term $-\dot{q}_r^\intercal (F + b) \dot{q}_r$ is negative. Thus, the system is stable if $D = \frac{1}{2} k \sum_{n=1}^{N} d_n{}^\intercal \dot{p}_n d_n$ is

also negative. We now study which conditions on p_n ensure this is the case.

4.2.2 Constraints on the weights p_n

First of all, we assume that p_n behaves as a probability, that is $\sum_{n=1}^{N} p_n = 1$, $\sum_{n=1}^{N} \dot{p}_n = 0$ and $\forall n, p_n \geq 0$. To simplify the study we consider the case in which only two virtual mechanisms are active, and thus $\dot{p}_2 = -\dot{p}_1$. By simplifying (4.9) we get:

$$\dot{V} = -\dot{q}_r^{\intercal}(F+b)\dot{q}_r + \tfrac{1}{2}k\dot{p}_1\left[d_1{}^{\intercal}d_1 - d_2{}^{\intercal}d_2\right]. \tag{4.10}$$

Because $k > 0$, the system is asymptotically stable *iff* $\dot{p}_1\left[d_1{}^{\intercal}d_1 - d_2{}^{\intercal}d_2\right] < 0$. This leads to the construction of a probability function that satisfy the following conditions:

$$\begin{cases} \dot{p}_1 > 0 \text{ when } \|d_1\| < \|d_2\|, \\ \dot{p}_1 < 0 \text{ when } \|d_2\| < \|d_1\|. \end{cases} \tag{4.11}$$

and vice-versa $\dot{p}_2$.

The conditions above show that the probability function p_n has to be dependent on the errors d_n, in particular, the inequalities provide the intuition that if the robot is closer to one guide the probability of belonging to that guide should increase over time, and consequently its error should decrease. For instance, we can define a probability function that increases over time when the square error is decreasing: $\dot{p}_n > 0 \iff \frac{d(d_n{}^{\intercal}d_n)}{dt} = 2\dot{d}_n^{\intercal}d_n < 0$. We first model this function as a probability over all the guides, given an activation weight over individual guides. The probability p_n that the n^{th} cart is responsible for guiding the end-effector at position $\mathbf{x}$ becomes:

$$p_n = p(n; \mathbf{x}_r, s_{vm}^n) = \frac{g(\mathbf{x}_r; s_{vm}^n)}{\sum_{j=1}^{N} g(\mathbf{x}_r; s_{vm}^j)}. \tag{4.12}$$

We implement the activation weight $g(\mathbf{x}_\mathrm{r}; s^n_\mathrm{vm})$ as a radial basis function of Gaussian type:

$$g(\mathbf{x}_\mathrm{r}; \mathbf{x}_\mathrm{vm}, \Sigma_\mathrm{vm}) = e^{\left(-\frac{1}{2}(\mathbf{x}_\mathrm{vm}-\mathbf{x}_\mathrm{r})^\intercal {\Sigma_\mathrm{vm}}^{-1}(\mathbf{x}_\mathrm{vm}-\mathbf{x}_\mathrm{r})\right)}. \tag{4.13}$$

Defined the probability function (4.12) we can compute the derivatives for the two guides[3]:

$$\begin{aligned} \dot{p}_1 &= p_1 p_2 (\dot{d}_2^\intercal {\Sigma_{\mathrm{vm},2}}^{-1} d_2 - \dot{d}_1^\intercal {\Sigma_{\mathrm{vm},1}}^{-1} d_1), \\ \dot{p}_2 &= p_1 p_2 (\dot{d}_1^\intercal {\Sigma_{\mathrm{vm},1}}^{-1} d_1 - \dot{d}_2^\intercal {\Sigma_{\mathrm{vm},2}}^{-1} d_2). \end{aligned} \tag{4.14}$$

By considering the virtual mechanisms to have fixed positions i.e. $\dot{d}_n = -\dot{\mathbf{x}}_\mathrm{r}$ and by changing system reference in order to have $\mathbf{x}_\mathrm{r} = 0$, the expressions above can be simplified to:

$$\begin{aligned} \dot{p}_1 &= p_1 p_2 \dot{\mathbf{x}}_\mathrm{r}^\intercal ({\Sigma_{\mathrm{vm},1}}^{-1} \mathbf{x}_{\mathrm{vm},1} - {\Sigma_{\mathrm{vm},2}}^{-1} \mathbf{x}_{\mathrm{vm},2}), \\ \dot{p}_2 &= p_1 p_2 \dot{\mathbf{x}}_\mathrm{r}^\intercal ({\Sigma_{\mathrm{vm},2}}^{-1} \mathbf{x}_{\mathrm{vm},2} - {\Sigma_{\mathrm{vm},1}}^{-1} \mathbf{x}_{\mathrm{vm},1}). \end{aligned} \tag{4.15}$$

We are interested in studying the inequality $\dot{p}_n > 0$, to do so we can simplify (4.15) in the following:

$$\begin{aligned} \dot{p}_1 &\sim \dot{\mathbf{x}}_\mathrm{r}^\intercal (\mathbf{x}_{\mathrm{vm},1} - \mathbf{x}_{\mathrm{vm},2}) > 0, \\ \dot{p}_2 &\sim \dot{\mathbf{x}}_\mathrm{r}^\intercal (\mathbf{x}_{\mathrm{vm},2} - \mathbf{x}_{\mathrm{vm},1}) > 0. \end{aligned} \tag{4.16}$$

Note that this simplification is possible because $p_1, p_2 > 0$ and the covariance matrices are positive definite. By construction these scalar products are positive if the robot moves toward one of the point $\mathbf{x}_{\mathrm{vm},n}$, so for example $\dot{p}_1 > 0$ if the robot moves toward the point $\mathbf{x}_{\mathrm{vm},1}$. The inequality (4.16) combined with (4.11) shows that the robot is attracted by the closest guide, that the probabilities of that guide thus increase, and the system is thus stable.

[3] To compute these derivatives we consider the covariance matrices to be constant in respect of time, this is practically true if we consider that the covariance changes slowly due the fact that is extracted by human demonstrations, and by considering the virtual mechanisms with fixed position.

4.2.3 Virtual mechanisms with moving positions

We have shown under which conditions on p_n the system is stable for guides that do not move. We now consider the more general case where they do move, i.e. $\dot{\mathbf{x}}_{\text{vm},n} \neq 0,\ n = 1..N$.

In this situation we can add to the control input the tracking of the velocities generated by the virtual mechanisms: $u = g + \mathbf{J}_\text{r}^\intercal \sum_{n=1}^{N} p_n(kd_n + b\dot{d}_n)$. By substituting in (4.7) we obtain:

$$\dot{V} = -\dot{q}_\text{r}^\intercal F \dot{q}_\text{r} + E + D. \tag{4.17}$$

with

$$E = \sum_{n=1}^{N} (\dot{q}_\text{r}^\intercal \mathbf{J}_\text{r}^\intercal p_n k d_n + \dot{q}_\text{r}^\intercal \mathbf{J}_\text{r}^\intercal p_n b \dot{d}_n + \dot{d}_n^\intercal p_n k d_n). \tag{4.18}$$

The first term in (4.17) is always negative. We now focus on E. By substituting the error derivate $\dot{d}_n = \mathbf{J}_{\text{vm},n}\dot{s}_{\text{vm},n} - \mathbf{J}_\text{r}\dot{q}_\text{r}$:

$$E = \sum_{n=1}^{N} (-\dot{q}_\text{r}^\intercal \mathbf{J}_\text{r}^\intercal p_n b \mathbf{J}_\text{r}\dot{q}_\text{r} + \dot{s}_{\text{vm},n}^\intercal \mathbf{J}_{\text{vm},n}^\intercal p_n k d_n + \dot{q}_\text{r}^\intercal \mathbf{J}_\text{r}^\intercal p_n b \mathbf{J}_{\text{vm},n}\dot{s}_{\text{vm},n}). \tag{4.19}$$

$$= \sum_{n=1}^{N} (\underbrace{-\dot{q}_\text{r}^\intercal \mathbf{J}_\text{r}^\intercal p_n b \mathbf{J}_\text{r}\dot{q}_\text{r}}_{E.1} + \underbrace{p_n(d_n{}^\intercal k + \dot{q}_\text{r}^\intercal \mathbf{J}_\text{r}^\intercal b)\mathbf{J}_{\text{vm},n}\dot{s}_{\text{vm},n}}_{E.2}). \tag{4.20}$$

The latter simplification can be made because $\dot{s}_{\text{vm},n}^\intercal \mathbf{J}_{\text{vm},n}^\intercal p_i k d_n$ is a scalar, we can transpose it. To study $(E.2)$ in (4.20) we can refer to the virtual mechanism equation in (2.6). By substituting (2.6) in $(E.2)$ we have:

$$p_n(kd_n + b\mathbf{J}_\text{r}\dot{q}_\text{r})^\intercal \mathbf{J}_{\text{vm},n}(\mathbf{J}_{\text{vm},n}^\intercal b \mathbf{J}_{\text{vm},n})^{-1}\mathbf{J}_{\text{vm},n}^\intercal(-kd_n + b\mathbf{J}_\text{r}\dot{q}_\text{r}). \tag{4.21}$$

By defining $A_n = \mathbf{J}_{\text{vm},n}(\mathbf{J}_{\text{vm},n}^\intercal b \mathbf{J}_{\text{vm},n})^{-1}\mathbf{J}_{\text{vm},n}^\intercal$, $(E.2)$ becomes:

$$-p_n k^2 d_n{}^\intercal A_n d_n + p_n \dot{q}_\text{r}^\intercal \mathbf{J}_\text{r}^\intercal b^2 A_n \mathbf{J}_\text{r}\dot{q}_\text{r}. \tag{4.22}$$

We can use (4.22) in the expression (4.20), therefore:

$$\begin{aligned} E &= \sum_{n=1}^{N} (-\dot{q}_{\mathrm{r}}^{\intercal} \mathbf{J}_{\mathrm{r}}^{\intercal} p_n b \mathbf{J}_{\mathrm{r}} \dot{q}_{\mathrm{r}} - p_n k^2 d_n{}^{\intercal} A_n d_n + p_n \dot{q}_{\mathrm{r}}^{\intercal} \mathbf{J}_{\mathrm{r}}^{\intercal} b^2 d_n A_n \mathbf{J}_{\mathrm{r}} \dot{q}_{\mathrm{r}}), \\ &= \sum_{n=1}^{N} (-p_n k^2 d_n{}^{\intercal} A_n d_n - p_n b \dot{q}_{\mathrm{r}}^{\intercal} \mathbf{J}_{\mathrm{r}}^{\intercal} (I - b A_n) \mathbf{J}_{\mathrm{r}} \dot{q}_{\mathrm{r}}). \end{aligned} \tag{4.23}$$

To have E negative definite we have to prove that the matrix $(I - bA_n)$ is positive semi-definite. Since $\mathbf{J}_{\mathrm{vm},n}$ is a column vector, the following inequality holds:

$$(I - \mathbf{J}_{\mathrm{vm},n} (\mathbf{J}_{\mathrm{vm},n}^{\intercal} \mathbf{J}_{\mathrm{vm},n})^{-1} \mathbf{J}_{\mathrm{vm},n}^{\intercal}) \succeq (I - I \lambda_{max} (\mathbf{J}_{\mathrm{vm},n} (\mathbf{J}_{\mathrm{vm},n}^{T} \mathbf{J}_{\mathrm{vm},n})^{-1} \mathbf{J}_{\mathrm{vm},n}^{T})) = 0, \tag{4.24}$$

as the maximal eigenvalue of matrix $\mathbf{J}_{\mathrm{vm},n} (\mathbf{J}_{\mathrm{vm},n}^{\intercal} \mathbf{J}_{\mathrm{vm},n})^{-1} \mathbf{J}_{\mathrm{vm},n}^{\intercal}$ is 1 by construction i.e.

$$\lambda_{max} (\mathbf{J}_{\mathrm{vm},n} (\mathbf{J}_{\mathrm{vm},n}^{\intercal} \mathbf{J}_{\mathrm{vm},n})^{-1} \mathbf{J}_{\mathrm{vm},n}^{\intercal}) = 1. \tag{4.25}$$

Due the presence of numeric issues, the scalar inverse $(\mathbf{J}_{\mathrm{vm},n}^{\intercal} \mathbf{J}_{\mathrm{vm},n})^{-1}$, could be calculated as $(\mathbf{J}_{\mathrm{vm},n}^{\intercal} \mathbf{J}_{\mathrm{vm},n} + b_d)^{-1}$ to avoid infinite value for certain configurations (2.16). In this case the inequality holds as:

$$\lambda_{max} (\mathbf{J}_{\mathrm{vm},n} (\mathbf{J}_{\mathrm{vm},n}^{\intercal} \mathbf{J}_{\mathrm{vm},n} + b_d))^{-1} \mathbf{J}_{\mathrm{vm},n}^{\intercal}) < \lambda_{max} (\mathbf{J}_{\mathrm{vm},n} (\mathbf{J}_{\mathrm{vm},n}^{\intercal} \mathbf{J}_{\mathrm{vm},n})^{-1} \mathbf{J}_{\mathrm{vm},n}^{\intercal}). \tag{4.26}$$

Finally we proved that E is negative semi-definite.

To prove that the Lyapunov function (4.17) is negative definite, we have to study D. Before we studied this term when the virtual mechanisms don't move, now we consider $\dot{\mathbf{x}}_{\mathrm{vm},n} \neq 0$, therefore we can write the derivative of the probability (4.14) for the first virtual mechanism as:

$$\dot{p}_1 = p_1 p_2 \left(\dot{\mathbf{x}}_{\mathrm{r}}^{\intercal} \left({\Sigma_{\mathrm{vm},1}}^{-1} \mathbf{x}_{\mathrm{vm},1} - {\Sigma_{\mathrm{vm},2}}^{-1} \mathbf{x}_{\mathrm{vm},2} \right) + \dot{\mathbf{x}}_{\mathrm{vm},2}^{\intercal} {\Sigma_{\mathrm{vm},2}}^{-1} \mathbf{x}_{\mathrm{vm},2} - \dot{\mathbf{x}}_{\mathrm{vm},1}^{\intercal} {\Sigma_{\mathrm{vm},1}}^{-1} \mathbf{x}_{\mathrm{vm},1} \right). \tag{4.27}$$

Considering (2.4) we have that the velocities $\dot{\mathbf{x}}_{\mathrm{vm},n}$ are orthogonal to the forces applied on the

virtual mechanisms, so, by plugging (2.3) into (2.4) and choosing $k \gg b$ we obtain the following simplification

$$\mathbf{J}_{\text{vm}}{}^{\intercal}\left[k(-\mathbf{x}_{\text{vm}}) + b(\dot{\mathbf{x}}_{\text{r}} - \dot{\mathbf{x}}_{\text{vm}})\right] \approx \mathbf{J}_{\text{vm}}{}^{\intercal}\mathbf{x}_{\text{vm}} \approx 0, \tag{4.28}$$

meaning that the velocities of the virtual mechanism are orthogonal to its position vector. By substituting the virtual mechanism equation (2.6) in (4.27) and considering the simplification done before (4.28), we get

$$\begin{aligned}\dot{p}_1 = p_1 p_2 \Big(& \dot{\mathbf{x}}_{\text{r}}^{\intercal}\left(\Sigma_{\text{vm},1}{}^{-1}\mathbf{x}_{\text{vm},1} - \Sigma_{\text{vm},2}{}^{-1}\mathbf{x}_{\text{vm},2}\right) \\ & + k\mathbf{x}_{\text{vm},1}{}^{\intercal}A_1{}^{\intercal}\Sigma_{\text{vm},1}{}^{-1}\mathbf{x}_{\text{vm},1} - k\mathbf{x}_{\text{vm},2}{}^{\intercal}A_2{}^{\intercal}\Sigma_{\text{vm},2}{}^{-1}\mathbf{x}_{\text{vm},2}\Big).\end{aligned} \tag{4.29}$$

The terms with A_n in the equation above can be proven to be null using (4.28):

$$k(\mathbf{J}_{\text{vm},n}^{\intercal} b \mathbf{J}_{\text{vm},n})^{-1} \underbrace{\mathbf{x}_{\text{vm}n}{}^{\intercal}\mathbf{J}_{\text{vm},n}}_{\approx 0} \mathbf{J}_{\text{vm},n}^{\intercal}\Sigma_{\text{vm},n}{}^{-1}\mathbf{x}_{\text{vm}n} \approx 0, \tag{4.30}$$

so equation (4.29) can be written as done before for the case with fixed virtual mechanisms (4.15), concluding with the same results. Finally the system with the chosen control converges asymptotically to the equilibrium point. The domain of attraction of the equilibrium point for each guide can be computed using (4.11). We investigated the stability of the control system in Figure 4.2. Using the direct Lyapunov method, we derived (4.9). By assuming that p_n behaves as a probability, we could derive the conditions (4.11). We showed that these conditions are met, and the system is thus stable, when a Gaussian function is used to relate probabilities p_n to errors d_n. We then generalized to the case when the virtual mechanism is moving.

4.3 Equilibrium points

In this section we study the equilibrium points of the system through the Lyapunov function defined in (4.3). We do so to understand where the robot converges when using the multiple

guides controller (4.1).

4.3.1 Equilibrium points in respect of the weights p

To study the equilibrium points of the system, we can analyse in which points V is null. It's easy to see that V is null when $\dot{q}_\mathrm{r} = 0$, and $\sum_{n=1}^{N} {d_n}^\intercal p_n d_n = 0$. As done before, to simplify the analysis we can consider the case with only two virtual guides $N = 2$:

$$\|d_1\|^2 p_1 + \|d_2\|^2 p_2 = 0. \tag{4.31}$$

Using the constraint $p_1 = 1 - p_2$:

$$\|d_1\|^2 + p_2(\|d_2\|^2 - \|d_1\|^2) = 0. \tag{4.32}$$

We can study the equilibrium points based on the values assumed by p_2, as we know $0 \leq p_2 \leq 1$. In the extreme values of p_2 equation (4.32) gives:

- $p_2 = 0$ and $p_1 = 1 \Rightarrow \|d_1\| = 0 \iff \mathbf{x}_\mathrm{r} = \mathbf{x}_{\mathrm{vm},1}$,
- $p_2 = 1$ and $p_1 = 0 \Rightarrow \|d_2\| = 0 \iff \mathbf{x}_\mathrm{r} = \mathbf{x}_{\mathrm{vm},2}$.

Thus by analyzing the equilibrium in respect of the weights p we find that there are two equilibrium points coinciding with the virtual mechanisms positions. For the values $0 < p_2 < 1$ The probability can be written as $p_2 = \frac{1}{c}$ with $c > 1$, therefore:

$$\underbrace{(c-1)}_{>0} \|d_1\|^2 = -\|d_2\|^2. \tag{4.33}$$

Thus for the values of p_2 between 0 and 1 there are no equilibrium points, since there are no

solutions for the equation above except in the case $d_1 = d_2 = 0$. The equilibrium points could be studied in respect of p_1 leading to the same results.

The problem of this analysis is that we are studying the equilibrium points without considering the dependence between the weights p and the errors d. This is the reason why we can only prove the implication in one direction ($\Rightarrow$) and not in the other. In general we have:

$$\|d_n\| = 0 \not\Rightarrow p_n = 1. \tag{4.34}$$

This means that if the error for a generic mechanism n is zero, it does not imply that the corresponding weight is 1. This result is a consequence of the fact that the weights are defined as probabilities (4.12).

4.3.2 Equilibrium points in respect of the errors d

First we can generalize the Lyapunov function (4.3) by weighting the d_n with the associated inverse covariance matrices[4]:

$$V = \tfrac{1}{2}\dot{q}_\mathrm{r}^\intercal M(q_\mathrm{r})\dot{q}_\mathrm{r} + \tfrac{1}{2}k\sum_{n=1}^{N} p_n {d_n}^\intercal \Sigma_{\mathrm{vm},n}^{-1} d_n \;>\; 0,\ \forall\ \dot{q}_\mathrm{r}, d_n \neq 0, \Sigma_{\mathrm{vm},n}^{-1} > 0. \tag{4.35}$$

As done before, we can study in which points V is null and we can consider the case with only two virtual guides $N = 2$ with the constraint $p_1 = 1 - p_2$:

$$d_{m_1}^2 + p_2(d_{m_1}^2 - d_{m_2}^2) = 0, \tag{4.36}$$

where $d_{m_1}^2 = {d_n}^\intercal \Sigma_{\mathrm{vm},n}^{-1} d_n$ represents the square of the Mahalanobis distance between the virtual mechanism and the robot end-effector. For the extreme values of p_2 (4.36) gives:

- $p_2 = 0$ and $p_1 = 1 \Rightarrow d_{m_1}^2 = 0 \iff \mathbf{x}_\mathrm{r} = \mathbf{x}_{\mathrm{vm},1}$,

[4]This generalization does not impact the stability analysis performed before since we assumed that the covariance matrices are constant in respect of time and positive definite.

- $p_2 = 1$ and $p_1 = 0 \Rightarrow d_{m_2}^2 = 0 \iff \mathbf{x}_\mathrm{r} = \mathbf{x}_{\mathrm{vm},2}$.

As done before with the simplification introduced in (4.33) we have:

$$\underbrace{(c-1)}_{>0} d_{m_1}^2 = -d_{m_2}^2. \tag{4.37}$$

That gives a unique solution for $d_{m_1}^2 = d_{m_2}^2 = 0$. Now we are interested in the other implication i.e:

$$d_{m_n}^2 = 0 \Rightarrow p_n = 1, \tag{4.38}$$

let's suppose we are in the case where $d_{m_1}^2 = 0$ and $d_{m_2}^2 \geq 0$, therefore:

$$p_1 = \frac{1}{1 + \underbrace{e^{\left(-\frac{1}{2} d_{m_2}^2\right)}}_{g_2}}, \tag{4.39}$$

$$p_2 = 1 - p_1.$$

If $d_{m_2}^2 = 0$ we have the case where the position of the two virtual mechanisms coincide (4.37), so the probabilities become:

$$d_{m_1}^2 = d_{m_2}^2 = 0 \Rightarrow p_1 = p_2 = \frac{1}{2}. \tag{4.40}$$

If $d_{m_2}^2 > 0$, we can study the values of g_2 for which the probability $p_1 = 1$. By imposing $p_1 = 1$ we have $g_2 = 0$. Because of the exponential this equation doesn't have an exact solution. We can find an approximate solution for a small value i.e. by imposing $g_2 < \varepsilon$ where ε is a very small value ($0 < \varepsilon \ll 1$). By applying the logarithm to both sides we obtain: $d_{m_2}^2 > 2l_\varepsilon$ where l_ε represents the natural logarithm computed in ε. So in this case the conditions for which the

probability is close to one becomes:

$$d_{m_2}^2 > 2l_\varepsilon \text{ and } d_{m_1}^2 = 0 \Rightarrow p_1 \simeq 1 \text{ and } p_2 \simeq 0, \tag{4.41}$$

and vice-versa. These results show that the equilibrium points coincide with one of the multiple mechanism position only if the related Mahalanobis distance between the robot and the mechanism is zero and only if the robot is far enough from the other mechanisms.

How far from the other mechanism depends on the covariance matrix defined by the demonstrations since we are using the Mahalanobis distance to define the basin of attraction of the equilibrium points.

For this reason and to simplify the study, we can substitute the covariance matrix with a positive scalar value α defined based on the norm square of the error:

$$e^{\left(-\frac{\alpha}{2}(d_2^T d_n)\right)} = e^{\left(-\frac{\alpha}{2}\|d_n\|^2\right)} < \varepsilon. \tag{4.42}$$

In this case we obtain $\|d_n\| > \sqrt{\frac{2l_\varepsilon}{\alpha}}$. So we can define alpha related to a maximum value of $\|d_n\|$ for which the exponential reaches zero. For example α could be defined as:

$$\alpha = \frac{2l_\varepsilon}{\|d_{max}\|^2}, \tag{4.43}$$

in this way we obtain a relation depending on the error that is not affected by the covariance (so that is not affected by the human demonstrations):

$$\|d_2\| > \|d_{max}\| \text{ and } \|d_1\| = 0 \Rightarrow p_1 \simeq 1 \text{ and } p_2 \simeq 0, \tag{4.44}$$

and vice-versa.

4.4 Conclusions

In applications of co-manipulation where users must be able to sequentially switch between multiple tasks, it is necessary to have multiple guides i.e. one for each task. We propose a control framework where multiple virtual guides are active in parallel, and the appropriate guide is recognized on-line during the execution of the movement. By weighting the controls of the individual guides with their probabilities, we are able to show that stability of the overall control system is ensured. We studied the equilibrium points of the system by analysing in which points the Lyapunov function is null. We found that the equilibrium points coincide with the mechanism positions with the basins of attraction defined by the inverse of the covariance matrix extracted from the demonstrations.

Chapter 5

Library of Virtual Guides

One issue in using multiple virtual guides is addressing contexts where new tasks may arise during operation. How can users intuitively demonstrate guides for new tasks or modify an existing guide? And if the robot movements are constrained to a guide, how can users 'escape' the guide to demonstrate new trajectories?

In this chapter we address these questions and define the concept of *library* of virtual guides. Furthermore, we define how the user interacts with the library and which kind of problems could arise when using multiple guides in parallel.

5.1 What is a library of virtual guides?

A library of virtual guides represents a framework where the user is able to:

- *Create* a new guide.
- *Modify* the existing guides.
- *Use* multiple guides in parallel.

Create a new guide:

We presented two different methods to *create* new guides through GMM: a batch and an incremental method, chapter 3. As we saw previously, with the batch method the user provides

several demonstrations of the different tasks at once, meaning that multiple guides are created in one step. With the incremental method instead, the user provides a first demonstration which is used to generate a new guide. Afterwards the user can provide new demonstrations to create new guides for different tasks.

Modify the existing guides:

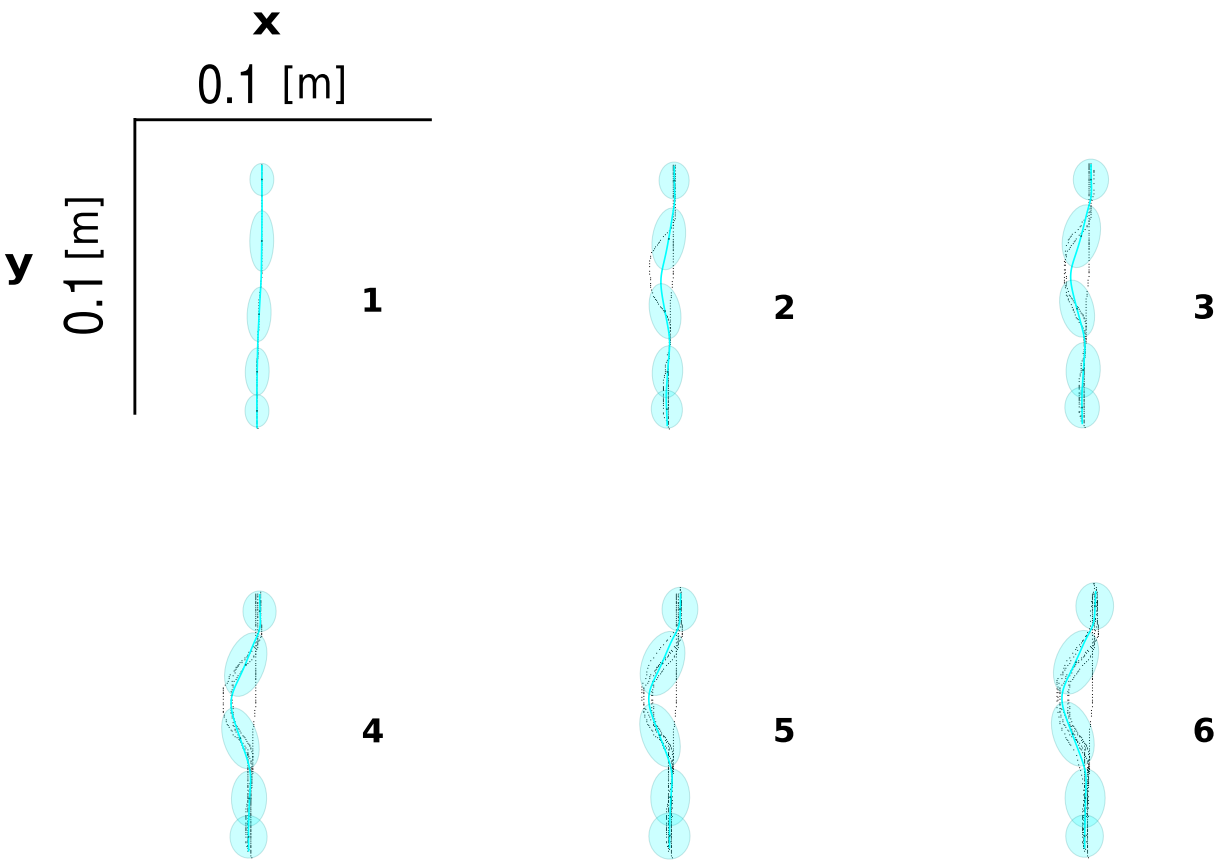


Figure 5.1: Thanks to the incremental training of GMM it is possible to iteratively modify an existing virtual guide. The figures ranging from 1 to 6 show the incremental modification of a GMM. Since GMM produces the mean of the demonstrations, it is necessary to provide several demonstrations in order to obtain the desired guide. For this reason in [Sanchez Restrepo et al., 2017] we explored the possibility of using splines to *locally* modify the virtual guide without the need of multiple demonstrations.

To *modify* a guide it is necessary to use the incremental method. After the user provides a new demonstration, the incremental clustering detects if the demonstration belongs to an existing guide, section 3.2.2. If this is the case, the demonstration is used to incrementally train the GMM which gets "updated" with the new data, see Figure 5.1. Otherwise the demonstration

is used to create a new guide.

Use multiple guides in parallel:

To *use* multiple guides in parallel we introduced in chapter 4 a control scheme where the influence of each mechanism is scaled with its probability to be responsible of the task execution. With this weighting scheme the robot end-effector is forced to remain on one of the multiple virtual guides available section 4.3. This is useful when the user has to execute one of the multiple tasks for which the guides are defined, and no new demonstrations are needed (i.e. there is no need to create or modify the guides).

In order to be able to create or modify the guides, the user should be able to 'escape' the active guides to record a new set of demonstrations. This is not possible with the actual control scheme. In this chapter we will address this problem and we will define a new weighting scheme that allows the user to *escape* the active guides.

Moreover we will define what are the interaction modes needed by the library to allow the user to *create*, *modify* and *use* the multiple guides.

5.2 Interaction modes

When using multiple virtual guides, an assignment problem arises: which guide is currently responsible for guiding the human? Probabilistic virtual guides enable us to address this question by weighting the contribution of each guide with the probability that this is the guide chosen by the human. We define three weighting schemes for three different modes of interaction:

1. Hard virtual guides. This weighting scheme enforces that the user is constrained to the guides, and can not escape them. This is useful in scenarios where no new tasks arise, and the user chooses to only *use* the current guides.

2. Soft virtual guides. With this weighting scheme, the user can 'escape' the virtual guides, for example to *create* or *modify* new guides. When the end-effector is not close to any of the guides, none of the guides is active. The robot then operates in zero-gravity mode.

3. Zero virtual guides. This turns off all of the virtual guides, and is equivalent to operating the robot in zero-gravity mode (i.e. gravity compensation mode). This mode may also be used to *create* new guides, but it is not useful to *modify* the guides since without the force feedback it is very difficult to understand where the guide is spatially placed.

Table (5.1) summarizes the most relevant differences between the three interaction modes, from the user's perspective.

	Type of guide		
	Zero	Soft	Hard
User can create new guides?	Yes	Yes	No
User can modify the guides?	No	Yes	No
Guides enabled when close to guide?	No	Yes	Yes
Guides enabled when far from guide?	No	No	Yes

Table 5.1: Main functionalities for the three interaction modes.

5.2.1 Hard Guides

If we have N virtual mechanisms, there are N mechanism positions $\mathbf{x}_{\text{vm}}^{n=1:N}$, and N probabilities. The probability that the n^{th} mechanism is responsible for guiding the end-effector at position $\mathbf{x}_{\text{r}}$ is

$$p(n;\mathbf{x}_{\text{r}},s_{\text{vm}}^{n}) = \frac{g(\mathbf{x}_{\text{r}};\mathbf{x}_{\text{vm}}^{n},\Sigma_{\text{vm}}^{n})}{\sum_{i=1}^{N} g(\mathbf{x}_{\text{r}};\mathbf{x}_{\text{vm}}^{i},\Sigma_{\text{vm}}^{i})} = \frac{g(\mathbf{x}_{\text{r}};s_{\text{vm}}^{n})}{\sum_{i=1}^{N} g(\mathbf{x}_{\text{r}};s_{\text{vm}}^{i})}, \tag{5.1}$$

where the means and covariance matrices of the mechanism position are determined from the mechanism phase s_{vm} with (3.26) and (3.32) 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:

$$\mathbf{F}_{\text{res}} = \sum_{n=1}^{N} p(n;\mathbf{x}_{\text{r}},s_{\text{vm}}^{n}) \cdot \mathbf{F}_{\text{vm}}^{n}. \tag{5.2}$$

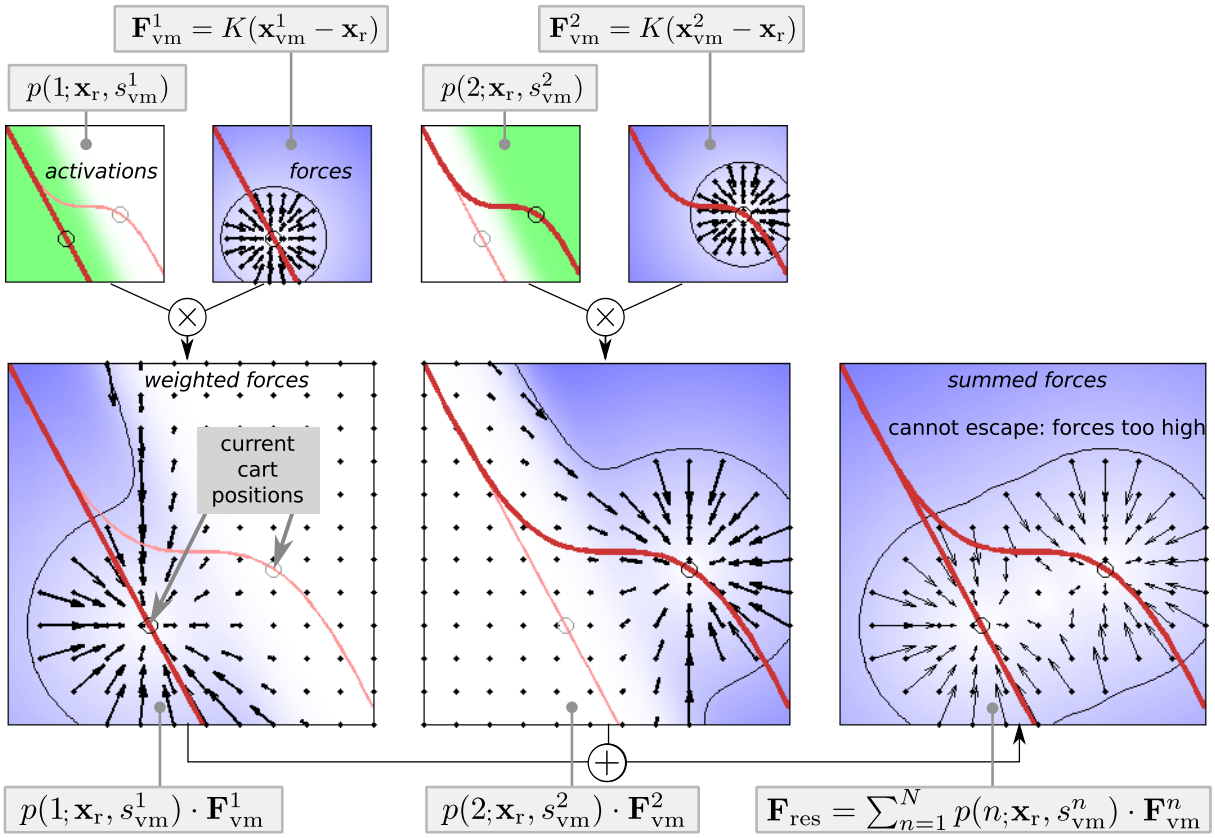


Figure 5.2: Scaling the forces of multiple virtual guides with probabilistic virtual mechanisms, using the first interaction mode (hard virtual guides). Each graph represents the value of one equation, depicted above/below the graphs. These illustrations are based on synthetic data. Since they represent a static snapshot, damping terms are omitted for simplicity.

The relevant probabilities and forces in this control mode are visualized in Figure 5.2. The stability of hard virtual guides is analyzed in chapter 4.

As a result of this interaction mode, the user can switch from a guide to another only when the guide's probabilities are close, otherwise the robot end-effector is forced to remain on the most probable guide. This is useful when the robot has to be constrained, and no new demonstrations are needed. In Figure 5.3, we illustrate switching between multiple virtual guides by using the hard interaction mode to scale the forces. In the left graph, we see that at the beginning of the trajectory, when the end-effector is close to both of the guides, the probabilities of both guides ≈ 0.5. Because the guides are close to each other, it is so easy to switch between them that it is difficult to feel the transition. In the middle plot, a switch is attempted half-way through the movement, when the probability that is the upper trajectory is 1. But with enough force, the

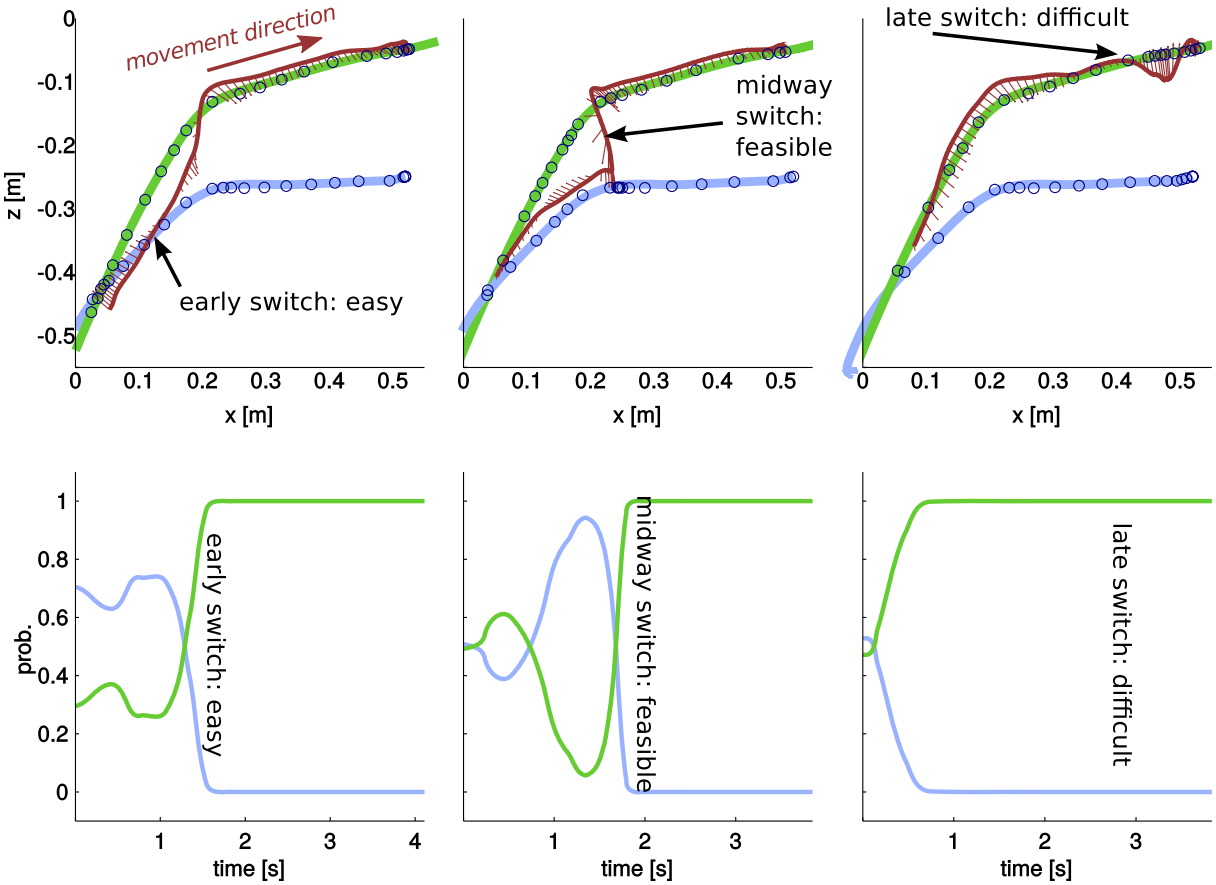


Figure 5.3: Illustration of switching between multiple probabilistic virtual guides, each for a different task/shelf. Top: Side view of virtual guides and end-effector movement. Bottom: probabilities of the two virtual guides. The dark red line represents the user movement the thin short lines represent the direction and magnitude of the resultant force applied by the virtual mechanisms.

user can still change to the other guide. The sensation is that the robot "gives way", and then locks into the other guide. Although sequential switching between multiple virtual (sub)guides for one task has been demonstrated [Aarno et al., 2005, Kuang et al., 2004], the switching between multiple virtual guides (that are active in parallel) for multiple tasks is a novel feature of our approach. Even further along the movements however (right plot), the distance between the guides is too large. Pushing the end-effector downwards hardly influences the probabilities, and the end-effector is locked to the upper guide.

5.2.2 Soft Guides

The underlying assumption in using the hard virtual guides is that $\mathbf{x}_r$ (i.e. the robot Cartesian position) *must* belong to one of the VMs. Another approach is to assume that if $\mathbf{x}_r$ is too far from the VMs, it does not belong to any of the VMs. To do so, we use a Gaussian function $h(\mathbf{x}_r; s_{vm})$ combined with the hard mode, i.e. a probability density function as in (3.5), but without the normalization factor $\sqrt{(2\pi)^k |\Sigma_{vm}|}$, as the Gaussian function has a known maximum of 1.

$$h(\mathbf{x}_r; s_{vm}) = e^{\left(-\frac{1}{2}(\mathbf{x}_r - \mathbf{x}_{vm}(s_{vm}))^T \Sigma_{vm}(s_{vm})^{-1} (\mathbf{x}_r - \mathbf{x}_{vm}(s_{vm}))\right)}. \tag{5.3}$$

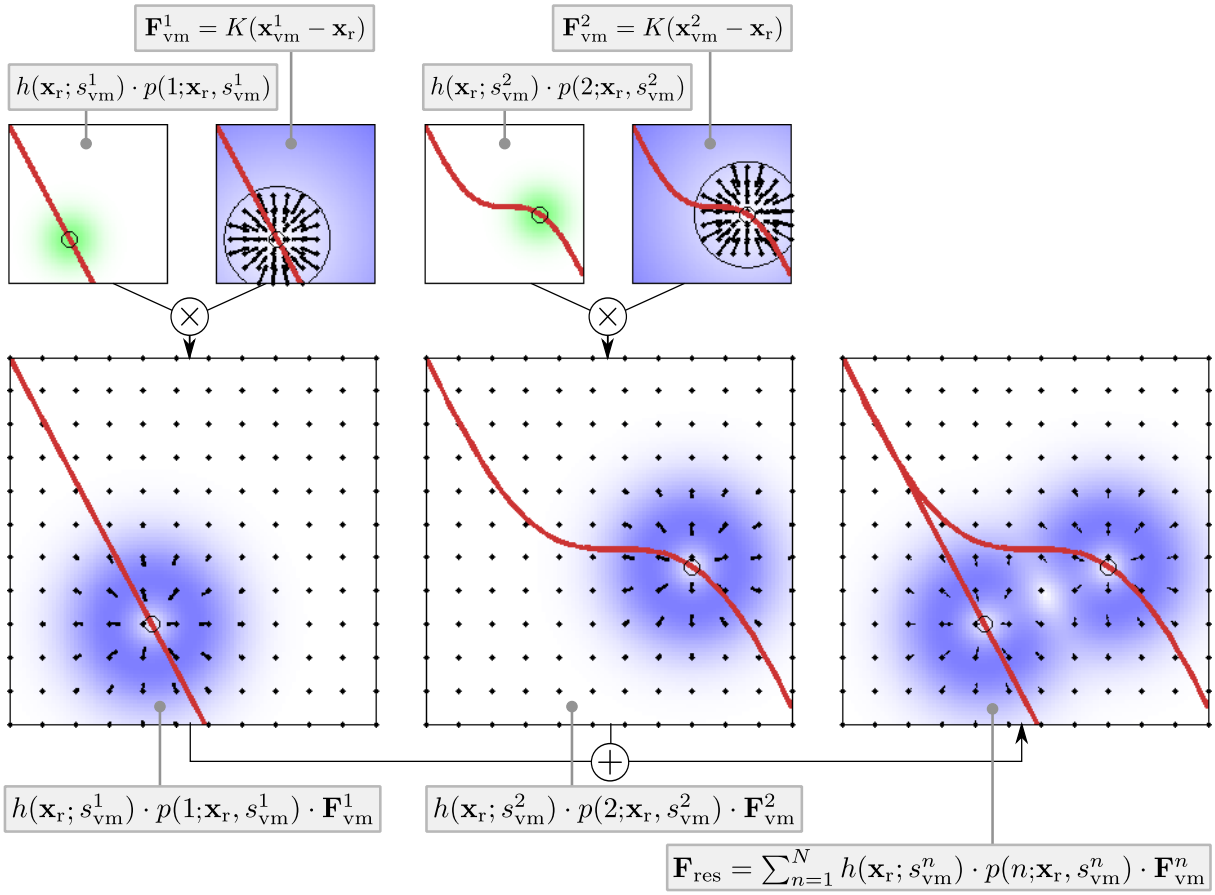


Figure 5.4: Scaling the forces of multiple virtual guides with probabilistic virtual mechanisms, using the second interaction mode (soft virtual guides).

By using these weights (to determine if an *individual* virtual guide is active in the first place), as well as the probability $p(n; \mathbf{x}_r, s^n_{vm})$ (to determine the relative weighting *between all*

the guides), the resultant force becomes

$$\mathbf{F}_{\text{res}} = \sum_{n=1}^{N} h(\mathbf{x}_{\text{r}}; s^n_{\text{vm}}) \cdot p(n; \mathbf{x}_{\text{r}}, s^n_{\text{vm}}) \cdot \mathbf{F}^n_{\text{vm}}. \tag{5.4}$$

The main difference between these methods arises when $\mathbf{x}_{\text{r}}$ is not close to any $\mathbf{x}^n_{\text{vm}}$, as illustrated in Figure 5.4. With the hard virtual guides, $\mathbf{x}_{\text{r}}$ is always pulled towards the closest $\mathbf{x}^n_{\text{vm}}$; the further you are, the stronger the force (standard proportional-derivative control). With the soft virtual guides, $\mathbf{x}_{\text{r}}$ does not belong to any $\mathbf{x}^n_{\text{vm}}$, and $\mathbf{F}^k_{\text{vm}} = 0$ for all n. The resultant force is therefore also zero. In Figure 5.5, we illustrate how probabilistic virtual guides enable a user

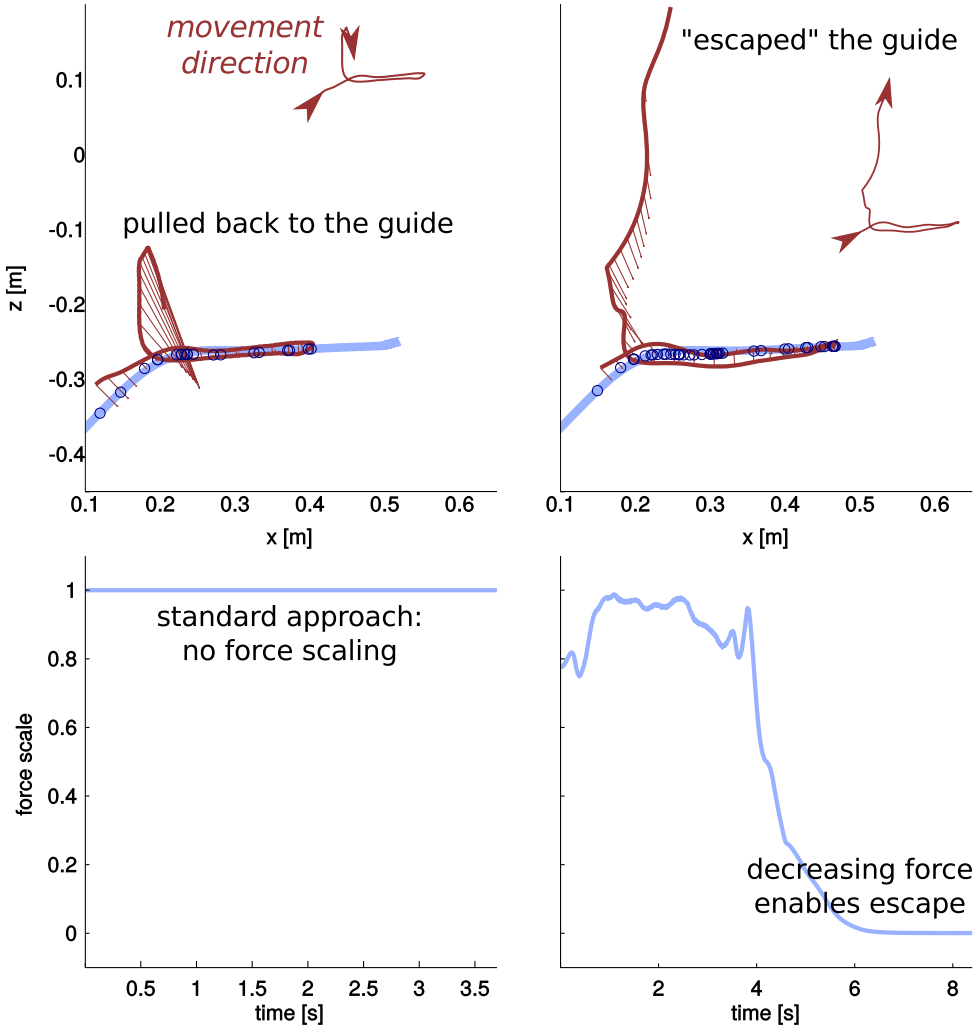


Figure 5.5: Illustration of escaping a virtual guide. Top: Side view of virtual guide and end-effector movement. Bottom: Probabilities of the virtual guides.

to escape the guide in a transparent way by using the soft interaction mode. Here, only the

virtual guide for positioning the object on the lower shelf is active. With a hard virtual guide (left graph), the user cannot escape the virtual guide, as the spring will exert high forces to pull the end-effector back to the guide. When using soft virtual guides (as in Figure 5.4), forces decrease if the distance to the guide becomes large. Once the user has 'escaped' the virtual guide, demonstrations for new tasks can be given. The advantage of using probabilistic virtual guides is that it leads to smooth transitions (rather than on/off switching as in [Aarno et al., 2005, Yu et al., 2005]), and does not require the manual design of distance thresholds (as in [Nolin et al., 2003]).

We now illustrate how escaping active guides and switching between multiple guides combine in a typical use case where new tasks arise during operation. As example we refer to the experiment conducted with the Meka robot to place objects in a cupboard with multiple shelves. Figure 5.6 shows the force scaling of virtual guides during 30 task executions. The use case is described in detail in the caption of Figure 5.6 to allow easy switching between the graph and its description.

5.2.3 Zero virtual guides

Finally, a second training mode in which only gravity-compensation is active is provided. Within our probabilistic framework, this training mode is interpreted as all virtual guides (if any) having a probability 0 of being responsible, i.e. $\forall n, p(n; s^n_{\text{vm}}) = 0$. This mode is used as initial state for the library when no guide is available.

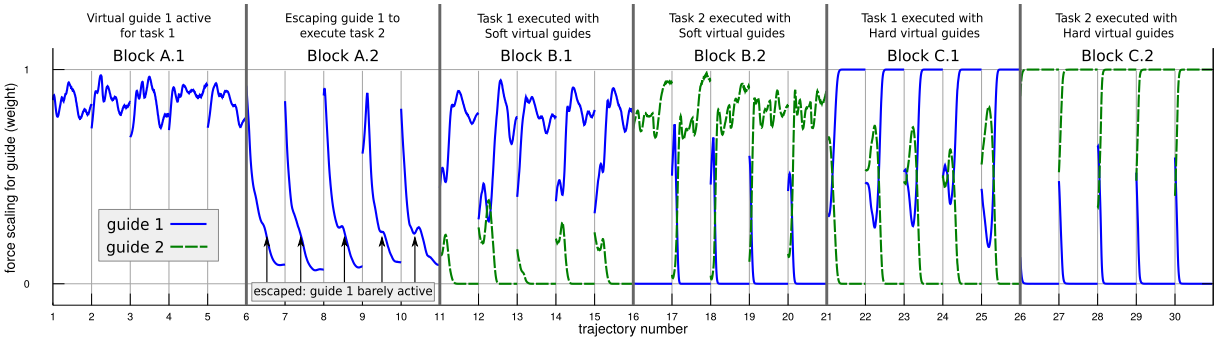


Figure 5.6: Use case illustrating how escaping virtual guides enables on-the-fly generation of new guides. In block A.1 the user executes task 1 (place the object on the lower shelf) using the previously trained guide 1. In block A.2 the user is able to execute task 2 – for which there is not yet a guide – by escaping guide 1. The trajectories from block A.2 are not considered to belong to guide 1 (see the low weights in block A.2), and are stored to train a new guide (guide 2), using the clustering and training methods from subsection 3.2.1. From then onwards, there are two guides. In block B.1 and B.2 (where task 1 and 2 are executed, respectively), we see that the correct guide is recognized during the movement, and used to guide the human. In this interaction mode the user can explore the functionality of the new task while still using the old one. Escaping the guides would still be possible, but not necessary since no new task has to be solved. Once the user is satisfied about the new task, the interaction mode can be switched to the hard mode. Finally, in block C.1 and C.2, it is assumed that no further tasks will arise, and the interaction mode is (manually) switched to provide hard virtual guides, which leads to earlier and even more accurate recognition of the appropriate guide for the task.

5.3 Deadlocks

When interacting with multiple guides there can be situations where one or multiple virtual mechanisms get stuck. This means that the mechanisms do not evolve along the guide any more, causing the robot to not move either. We call these unwanted situations "deadlocks". We distinguish between two different types of deadlocks: the first is related to the force acting on the robot end-effector, while the second is caused by the geometric disposition of the virtual guides. Next we will address two different solutions, one for each of them.

5.3.1 Force Deadlocks

As shown in subsection 4.3.2, using multiple virtual mechanisms brings the robot end-effector to converge on one of the N mechanism's position, in particular we have two types of equilibrium points:

1. $\dot{q}_r = 0,\ d_{m_i} = 0$ and $d_{m_j} > 2l_\varepsilon \ \ \forall j \neq i$

2. $\dot{q}_r = 0,\ d_{m_i} = 0 \ \ \forall i$

with $i, j = 1..N$

Where 1. represents the equilibrium point coinciding with one of the mechanisms position when its relative Mahalanobis distance from the robot end-effector is zero and the robot is far enough from the remaining mechanisms. The second type of equilibrium given by 2. appears when the mechanism positions coincide and their Mahalanobis distance from the robot end-effector is zero. The deadlock problem arises when we have two or multiple mechanisms which forces contrast each other.

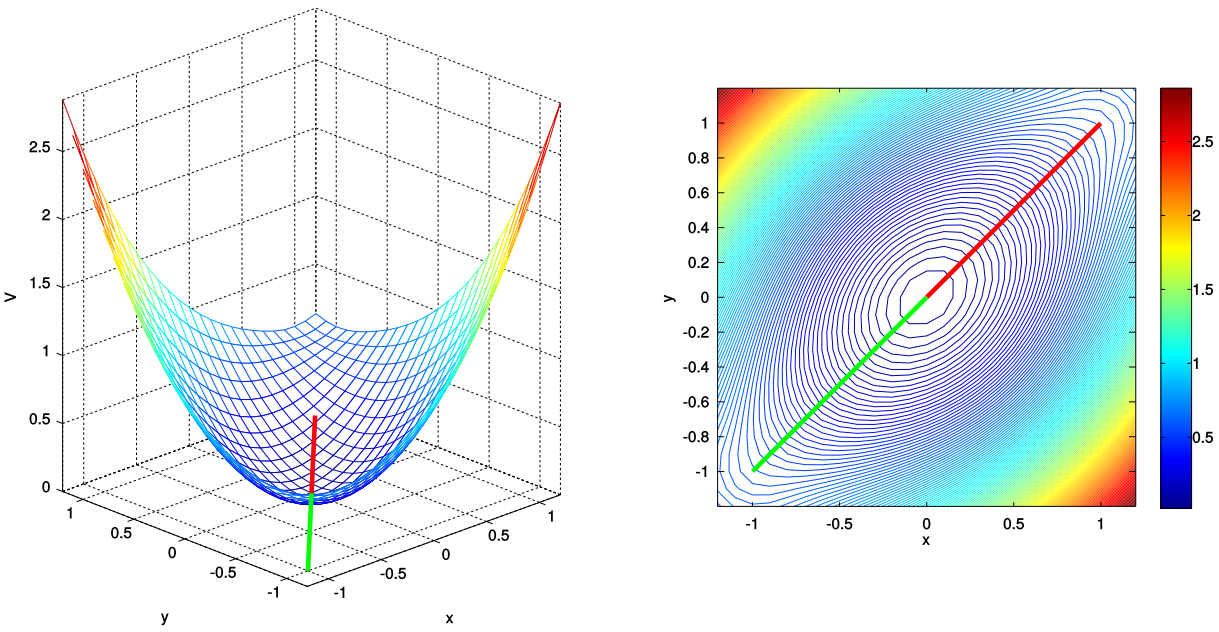


Figure 5.7: Left: In green and red are displayed the two virtual guides. The values of the Lyapunov function V are computed using a grid of points in x and y. Right: Contour plot for the function V, it is visible the deadlock point at coordinate $x = 0, y = 0$.

For example in Figure 5.7 we have two guides defined as two lines with a common point in P_0 with $x = 0, y = 0$. The point P_0 represents an equilibrium point of the second type 2. since the mechanisms position coincide. Given the particular disposition of the guides, P_0 represents a

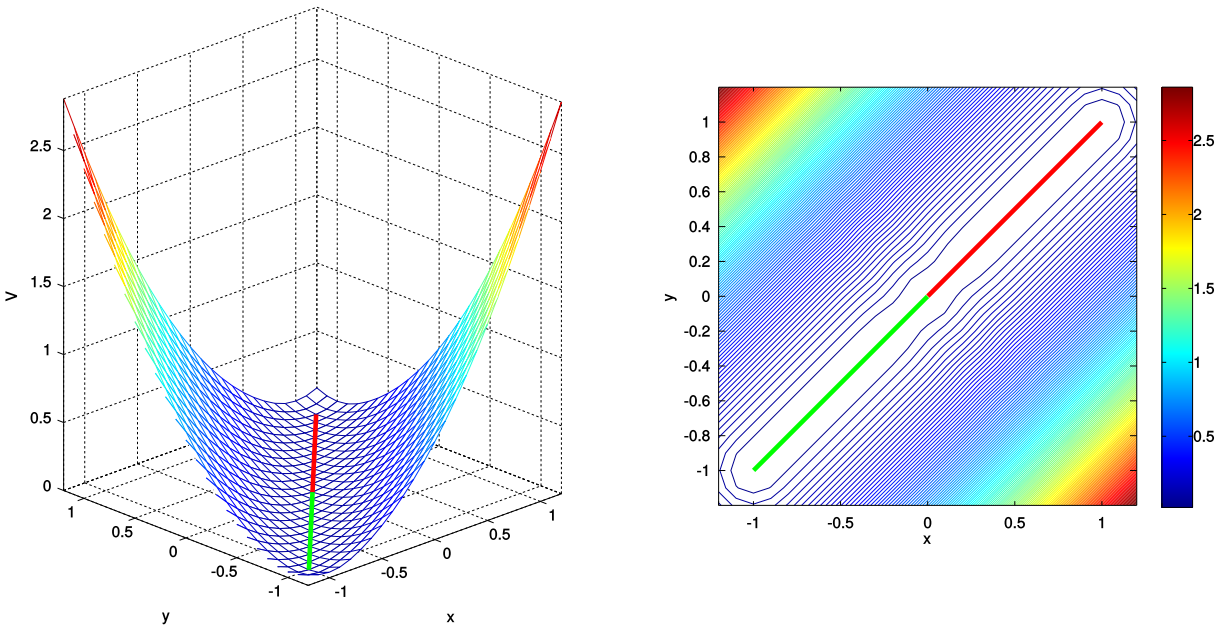


Figure 5.8: Right: The function V is now zero for all the points belonging to the two guides. The deadlock is disappeared.

deadlock. By pushing the robot in direction of the red guide, the green guide will pull back the mechanism with a force described by (2.3). Vice-versa if the robot is pushed to the green guide the red guide will pull back the robot. The overall effect is that the robot remains stuck in the point P_0. To solve this problem, we can remove the antagonist force produced by the green guide when the the red guide is the most probable (or vice-versa); by doing so, the robot would be free from the deadlock. Moreover by removing only the antagonist force to the red guide, we assure that the robot can still switch between the red and green guide. This solution can be generalized to the case where more than two guides are available. With more than two mechanisms, for each mechanism that is not responsible for the task execution (i.e. low probability), we remove the force component tangent to the mechanism's Jacobian responsible for the task execution. By doing so, the robot is no more locked in a point if one or more guide overlap on it. The resulting force applied on the robot end-effector (4.1) becomes:

$$\mathbf{F}_{\text{vm}} = \sum_{i=1}^{N} p_i \left(\mathbf{F}_{\text{vm}i} - \sum_{j \in \{1,\dots,N\}-i} \alpha_j^* \hat{T}_j (\mathbf{F}_{\text{vm}i} \hat{T}_j) \right), \tag{5.5}$$

with α_j^* scalar value defined as $\alpha_j^* = 1$ if the guide j is *active* i.e. it has the maximum p among

all the guides, otherwise $\alpha_j^* = 0$[1]. Instead $\hat{T}_j = \frac{\mathbf{J}_{\text{vm},j}}{\|\mathbf{J}_{\text{vm},j}\|}$ represents the versor of the Jacobian for the guide j. Results are shown in Figure 5.8. Alternatively we can write the formula above as an algorithm:

Find the active mechanism (i.e. the one with the max p);
$active_{idx} = 1$;
$p_{max} = 0.0$;
for $i \leftarrow 1$ **to** N **do**
  if $p_i > p_{max}$ **then**
    $active_{idx} = i$;
    $p_{max} = p_i$;
  end
end
Define the filter values α_i^*;
for $i \leftarrow 1$ **to** N **do**
  if $i == active_{idx}$ **then**
    $\alpha_i^* = 1$;
  else
    $\alpha_i^* = 0$
  end
end
Compute the force for each mechanism, remove the antagonist force components;
for $i \leftarrow 1$ **to** N **do**
  $\mathbf{F}_{\text{vm}i} = k(\mathbf{x}_{\text{vm},i} - \mathbf{x}_{\text{r}}) + b(\dot{\mathbf{x}}_{\text{vm},i} - \dot{\mathbf{x}}_{\text{r}})$;
  $\mathbf{F}_{\text{vm}} += p_i \mathbf{F}_{\text{vm}i}$;
  for $j \leftarrow 1$ **to** N **do**
    if $j != i$ **then**
      $\hat{T}_j = \frac{\mathbf{J}_{\text{vm},j}}{\|\mathbf{J}_{\text{vm},j}\|}$;
      $\mathbf{F}_{\text{vm}} -= p_i \alpha_j^* \hat{T}_j (\mathbf{F}_{\text{vm}i} \hat{T}_j)$;
    end
  end
end

Algorithm 1: Algorithm to remove the antagonist forces

5.3.2 Geometric Deadlocks

These deadlocks could appear because of the particular geometric disposition of the virtual guides, see the example in Figure 5.9.

To remove these deadlocks we discretize the guides with a fixed number of points M. When the p_i related to a guide becomes lower than a certain threshold (e.g. $p_i < 0.01$), instead of com-

[1] In order to avoid discontinuities, α_j^* can be filtered by a first order dynamical system.

puting the mechanism evolution by integrating the dynamical system (2.6), we select among the M points the one with the minimum distance from the robot end-effector, and we use this point as the current virtual mechanism position. When the p_i becomes higher than the selected threshold, the minimum distance point is used as initial point for the integration of the mechanism's dynamical system.

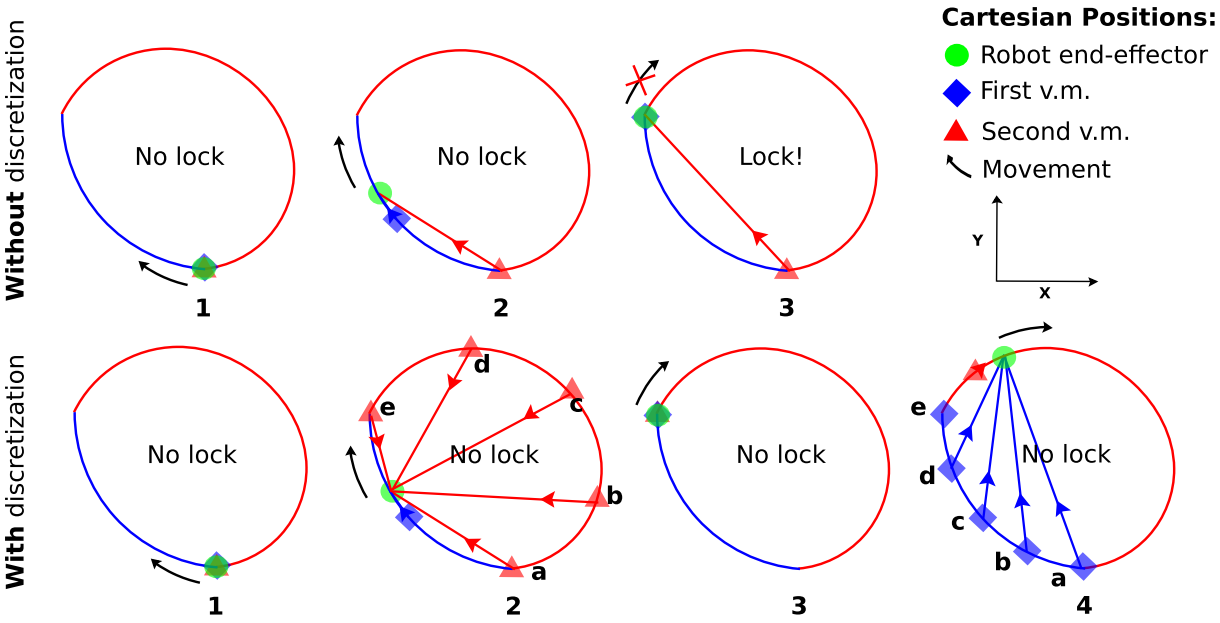


Figure 5.9: *Without discretization:* In 1 and 2 the robot end-effector can freely move along the blue guide, arrived to 3 the robot end-effector can not proceed on the red guide because the red virtual mechanism is "stuck" at the beginning of the red guide.
With discretization: In 1 and 3 the evolution of both the virtual mechanisms is computed through integration of (2.6) since their probability is higher than the selected threshold. Instead in 2 and 4 the discretization points $\{a,b,c,d,e\}$ are used to avoid the lock.

The success of this method is highly dependent on the number of points M used for the discretization and on the shape of the guide. In general, few points are needed if the guide presents a smooth shape.

5.4 Conclusions

In this chapter, we presented the concept of library of virtual guides. Thanks to the different weighting scheme explained in section 5.2 it is possible to select between a *hard* mode where the robot is constrained to one of the possible guides and a *soft* mode where the robot can

escape the guide when the user exerts enough force on it. Thanks to this last method, the user can provide new demonstrations which can be used to *create* or *modify* the guides using the incremental training explained in subsection 3.2.2.

When using multiple guides there could be the presence of deadlocks due to their relative disposition. We identified two types of deadlocks, one caused by the contrasting forces of different guides applied on the robot tool-tip and another type caused by the particular geometric disposition of the virtual guides. We proposed two solutions to remove these deadlocks.

Chapter 6

Experiments

In this chapter we present three different experiments conducted using the virtual guides explained in the previous chapters. The general goal of the following experiments is to prove that using virtual guides is useful to improve task performances, in terms of time and safety. A first pilot study has been conducted on a Meka robot at ENSTA-ParisTech as proof of concept for the library of virtual guides, while the second and the third experiments have been carried out on a co-manipulation robot developed at CEA-List[1] (Figure 6.1) by simulating two common industrial duties: A sanding task and a Pick And Place task.

Figure 6.1: Left: Meka robot at ENSTA-ParisTech. Right: ISybot co-manipulation robot.

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

6.1 Pilot Study with Meka

Our first experiment has been carried out on the Meka robot at ENSTA-ParisTech. The Meka is a humanoid-size robot with an anthropomorphic arm composed by 7-DOF, for the experiment, we fixed the 3-DOF related to the wrist and we used the remaining to perform the task. The application task, illustrated in Figure 6.2, is to place objects on two different shelves in a cupboard[2]. During one episode, users take the robot by the wrist, and guide it to one of two positions in the cupboard. At the desired position, the robot releases the object for placement on the shelf (when the phase of the guide > 0.9). In this pilot study, we first illustrate how we

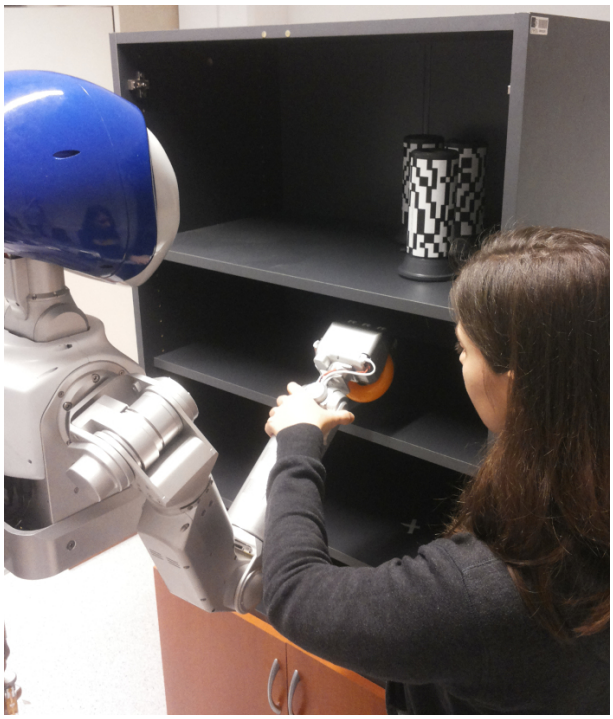


Figure 6.2: The task with meka consisted in using virtual guides assistance to facilitate the placement of objects in a cupboard with shelves.

generated the virtual guides by training two different GMMs using the batch method explained in subsection 3.2.1, afterwards, we compare single and multiple virtual guides with respect to 3 evaluation criteria.

[2]Although using heavy objects would have been better to demonstrate the advantages of using virtual guides, we did not have the possibility to do so due to the limited payload of the Meka robot.

6.1.1 Training

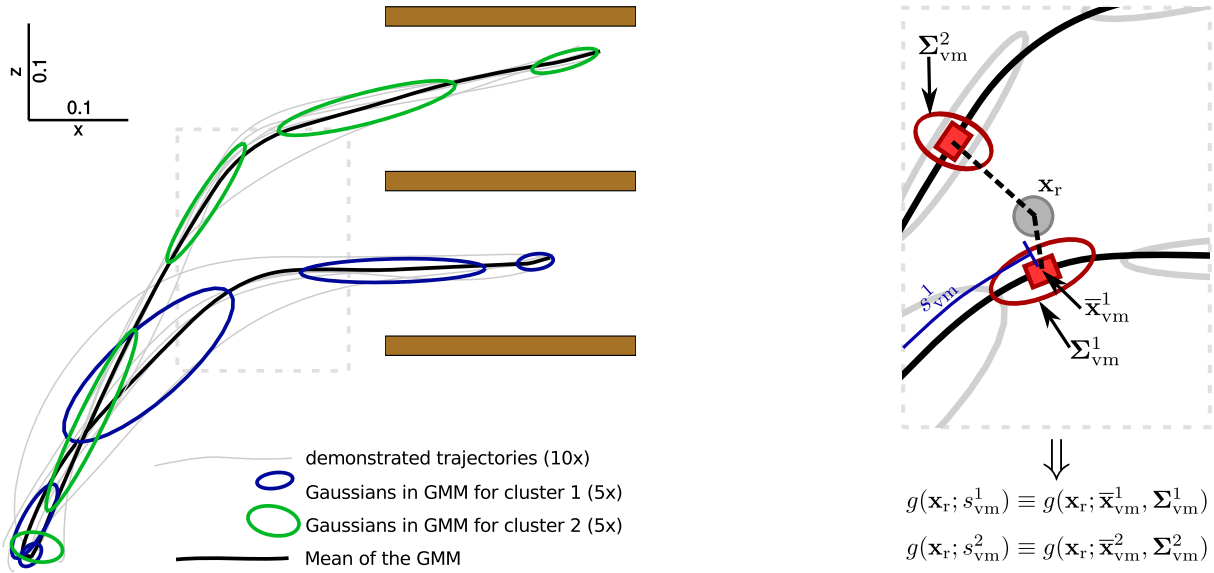


Figure 6.3: Left: The 10 demonstrated trajectories (light gray) and the two GMMs that are fitted to the 5 trajectories in each cluster. For visualization purposes, the data is projected on the *xz*-plane. Right: Relevant variables for computing $g(\mathbf{x}_\text{r}; s_\text{vm})$.

First the virtual guides related to the two different shelves have to be created. To do so we refer to the method explained in subsection 3.2.1. Each movement to place the object in the shelves is performed multiple times by the user through kinesthetic teaching; the user holds the end-effector of the Meka robot, and demonstrates by guiding it along the desired trajectory, trying to avoid collisions with the shelves and with the final goal of reaching the position where the object has to be placed. The demonstrations are gathered without a specific order, in this way the number of tasks is not known a priori.

Then dynamic time warping (DTW) section 3.2.1 is used to compute distances between trajectories that are independent of time. These distances are used to perform hierarchical clustering of the trajectories, see Figure 6.4. Afterwards, DTW is used to align the clustered trajectories with respect to time section 3.2.1, in order to reduce the entropy [Huber et al., 2008], see Figure 3.4 and Figure 3.5. The final step consists of fitting two GMMs which represent the two different tasks. Figure 6.3 depicts the result of fitting the GMMs to the clusters, where the number of Gaussians is manually set to 5 per cluster.

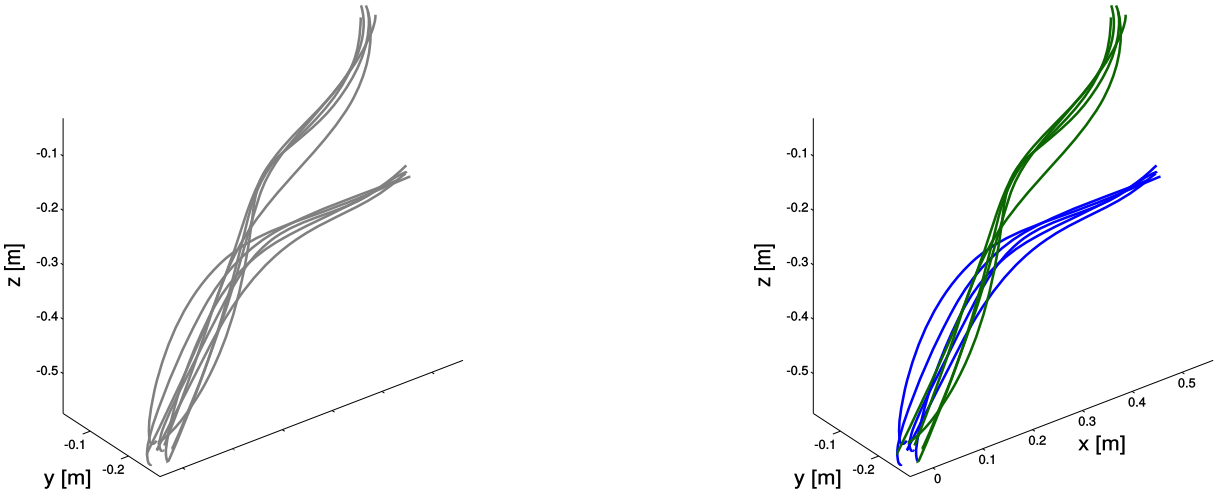


Figure 6.4: Left: Demonstrated trajectories. Right: Trajectories after the clustering. Each cluster represents a different movement to place the objects in the shelves.

6.1.2 Comparing Safety and Efficiency

We perform a pilot study with four users, and the set-up in Figure 6.2. We compare three assistance modes: gravity compensation only (no virtual guides), a single virtual guide (for the current task), or multiple virtual guides (by using hard virtual guides). With these 3 modes, each user executes 10 episodes for each of the 2 tasks (object goal positions), i.e. a total of $60 = 3 \times 10 \times 2$ episodes per user. Each user thus tries all modes and positions, but were presented in a randomized order to the users to avoid training effects.

The measures we are interested in are:

- Execution time, to measure efficiency.
- Accuracy of tracking.
- Actually observed collisions, to measure safety.

These results are summarized in Figure 6.5

From these results, we draw the following conclusions:

- Tracking errors substantially and significantly decrease when using virtual guides (from 3.0cm to 2.0/2.0 for Task 1 and from 5.2 to 2.5/2.7 for Task 2).

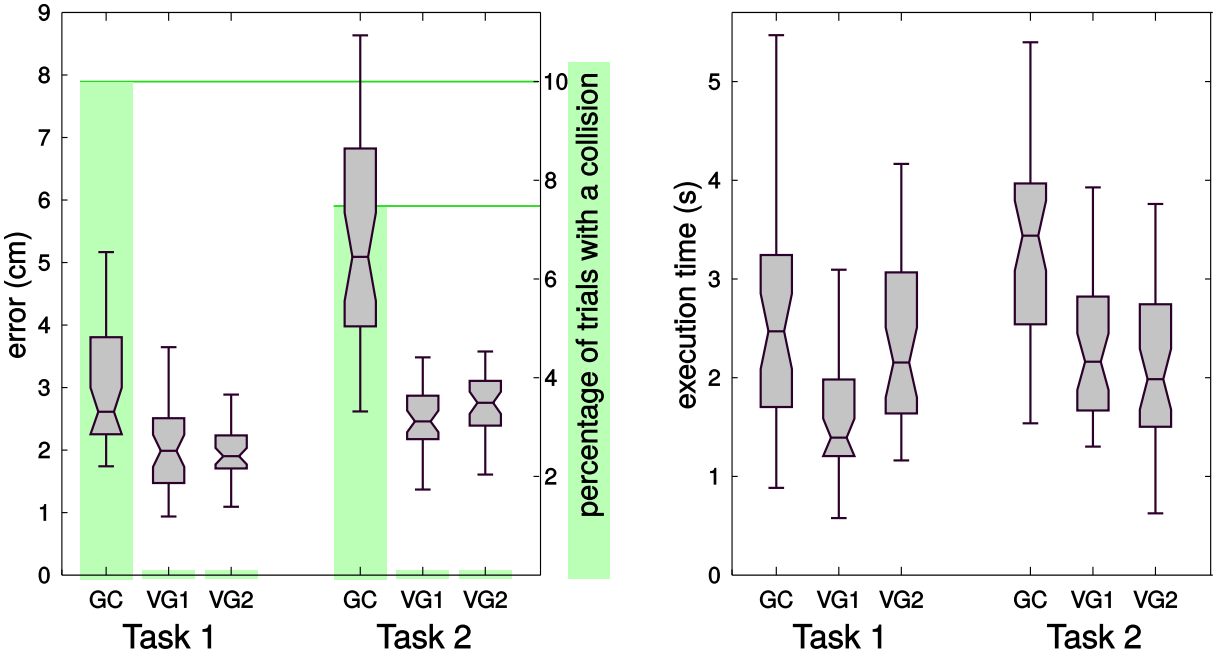


Figure 6.5: Comparison of the three assistance modes (gravity compensation only, only one virtual guide, multiple virtual guides), for both guides (for the upper and lower shelf), and three measures (execution time, position error, number of collisions)

- This enables virtual guides reduce the percentages of trials in which a collision occurs to 0.

- Execution times also decrease (from 2.7s to 1.7/2.3 for Task 1 and from 3.3 to 2.3/2.1 for Task 2).

- Using multiple guides in parallel (instead of activating only the appropriate guide for the task at hand) does not lead to a significant deterioration in performance.

Some of the spontaneous user comments about the virtual guides, although anecdotal, are nevertheless interesting to report. One user reported "I feel more confident in moving the robot"; this confidence is verified by the fact that users are able to execute tasks more quickly, because they are more confident that the robot will not collide with the cupboard. Another comment includes "It feels like there are virtual shelves". Thus the user interprets the robot as avoiding the shelves, rather than following a guide, which corresponds to what the demonstrator considers when demonstrating the trajectories.

6.2 Sanding task with ISybot robot

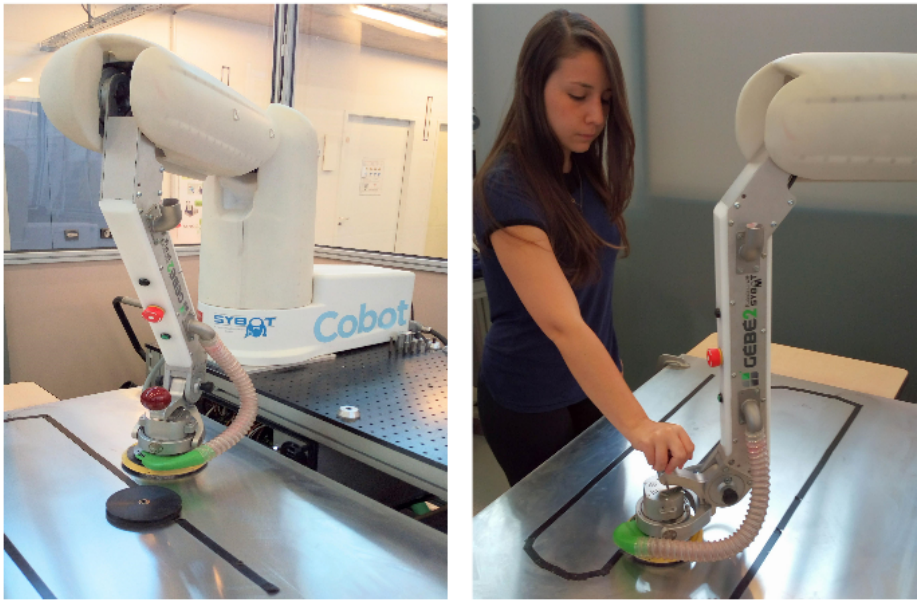


Figure 6.6: Left: ISybot robot with sanding tool and obstacle. Right: setup for the sanding experiment

The following experiment was conducted on a 3-DOF ISybot co-manipulation robot see Figure 6.6 with a sander. Despite in this experiment the creation of the virtual guides was performed by using a different algorithm than GMM, this experiment represents an important step to show the usefulness of virtual guides in collaborative tasks. The task consisted in using the cobot to perform a sanding duty. The simulated task consisted on using the tool to clean a metal sheet while following a sweeping trajectory defined by 2 red erasable marks. We used black tape to mark the trajectory in order to always draw the same red marks.

Two tasks were studied:

- *Case A*: cleaning task without obstacle.
- *Case B*: same cleaning task as in Case A but with an obstacle blocking the trajectory, see Figure 6.6 for reference.

In both cases, we compared two assistance modes:

- *Mode A*: the robot assists the user with the gravity compensation, i.e. the robot compensates the tool weight and its own weight.

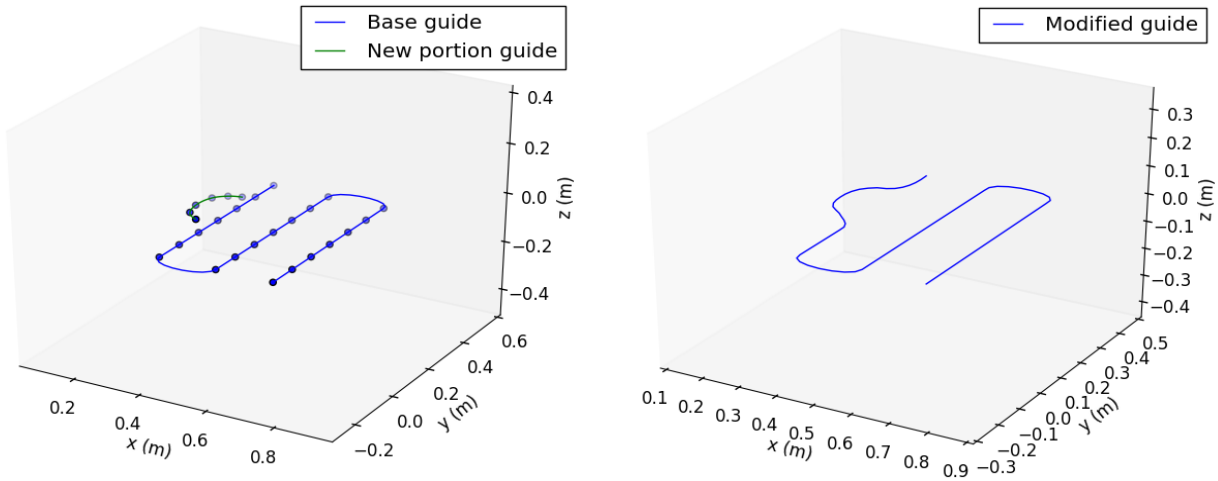


Figure 6.7: Left: Base guide and new portion guide. Right: Virtual guide's local modification.

- *Mode B*: the robot assists the user with the virtual guiding.

It was necessary to choose a simple task to minimize the impact of different skill levels of the user. However, the system could be adapted to more dynamic and complex scenarios, for example to a case where multiple guides are used to clean different paths chapter 4. In Mode A (gravity compensation mode) the user was able to move the robot freely. For Mode B (virtual guiding mode), we asked an expert user to program the guide. For Case B, the expert user modified a portion of the guide generated for Case A, using the iterative approach explained in [Sanchez Restrepo et al., 2017]. For the virtual guide's modification we used an alternative definition of the soft interaction mode (for details see [Sanchez Restrepo et al., 2017]).

6.2.1 Programming virtual guides by an expert user

To program a virtual guide for Case A, we used a function that takes two non-adjacent vertices of a rectangle and generates a set of points describing a sweeping trajectory. These vertices were shown by demonstration to the robot and recorded using the lower button located on the robot's third axis. With this set of points $\{x_m, y_m, z_m\}_{m=1:M}$, we generated a virtual guide using an Akima spline [Akima, 1970] see Figure 6.7 (base guide).

For Case B, it was not possible to entirely use the previous guide since there was an obstacle blocking the trajectory. We asked to an expert user to modify the guide using the local modifi-

Question	Mode A Mean	SD	Mode B Mean	SD	F	P-value
Do you think that you performed the task well?	5.21	1.19	5.89	0.91	F(1,12)=9.7028	0.0089
Did you feel you performed the task precisely?	4.82	1.19	5.64	1.06	F(1,12)=11.207	0.0058
Do you think the robot was helpful during the task execution?	3.82	1.58	5.29	1.65	F(1,12)=4,6126	0.0529
Do you think the robot was easy to work with?	5.11	1.26	5.39	1.37	F(1,12)=,50585	ns

Table 6.1: Survey results of the user study in mode A and mode B

cation algorithm explained in [Sanchez Restrepo et al., 2017]. While the guide was active, the user was able to move along the trajectory. When the user arrived near the obstacle, he/she was able to escape the guide thanks to the soft mode.

Then, the new points of the partial modification were shown by demonstration to the robot and recorded using the lower button located on robot's third axis see Figure 6.7 (new guide portion). After the last point was recorded, the upper black button launched the refining algorithm and a new guide was created see Figure 6.7 (modified guide).

6.2.2 User study

We designed a pilot study to observe how novice users perceived the virtual guide assistance and performed with the cobot. We recruited 14 participants from our research laboratory (with age between 22 and 33 years old, 5 females). Nine participants stated they had prior experience with robots, ranging from robotic courses to hands-on experience with industrial robots.

Three hypotheses were tested:

- H1: Virtual guides assistance reduces the task's execution time.
- H2: Virtual guides assistance improves task's performances.
- H3: Virtual guides are easy to use.

All participants were asked to perform the tasks Case A and Case B (i.e. cleaning task;

cleaning task with an obstacle), in both modes Mode A and Mode B (i.e. with guide; without guide), resulting to four test conditions. The four test conditions were presented in a randomized order to avoid training effects. For each condition, the participants were asked to perform the task 3 times in a row (Repetitions). At the beginning of each condition, the Case and the Mode were presented to the participants and they were able to familiarize themselves with the system. When a condition was completed, it was asked to the participants to fill a post-condition survey (Likert-scale survey with a rating from 1 to 7, with 1 as strong disagreement and 7 as strong agreement, the same for each condition). In total, the participants performed $12 = 2x2x3$ (Case*Mode*Repetition) cleaning tasks. To validate our hypotheses, we recorded the participants' times of execution of the 12 tasks and the answers to the four post-condition surveys.

The measures we were interested in are the following:

- Execution time, to validate H1 and H3.
- Observed collisions in case B, to validate H2.
- Survey results, to validate H2 and H3.

These results are summarized in Figure 6.8, Figure 6.9 and Table 6.1.

We performed a repeated-measure ANOVA on three factors: (1) the Cases, (2) the Modes, and (3) the Repetitions. Participants were grouped by their habit to use a robot. Post-hoc analyses were performed with Tuckey's HSD test. The significance threshold was set to $p < 0.05$.

6.2.3 Results

We found a significant main effect of the Modes on the execution time of the participants ($F(1,12) = 14.78; p < .01$). In Mode A, participants were slower than with in Mode B (Mode A: $M = 15, SD = 4.73$; Mode B: $M = 12.06, SD = 3.90$) see (6.8). This indicates that the virtual guides reduced the execution time. This validates H1. We also found a significant main

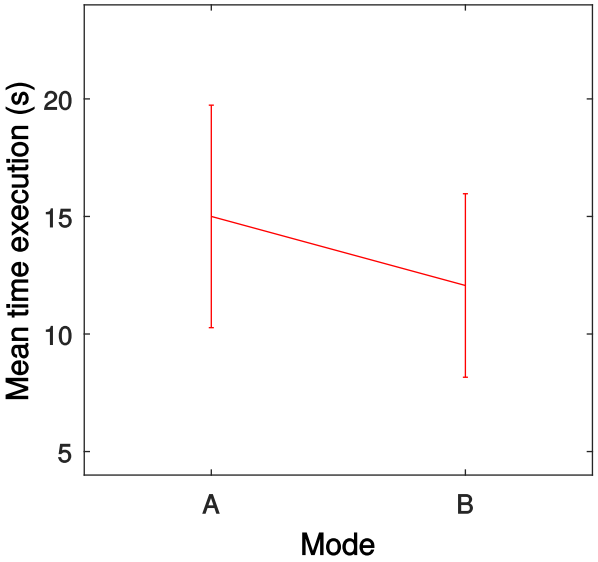


Figure 6.8: Task's execution time. Comparison between the two modes A and B (gravity compensation and virtual guiding).

effect of the Repetitions on the execution time of the participants $(F(2,24) = 14.79; p < .001)$. Posthoc analyses showed that the first repetition was longer than the second $(p < .001)$ and the third repetitions $(p < .001)$. This confirms there is a training effect. Moreover, there is a significant effect of the interaction between repetitions and one's experience with a robot $(F(2,24) = 5.76, p < .01)$ see Figure 6.9. Posthoc analyses showed that the first repetition was longer than the second $(p < .05)$ and the third repetitions $(p < .001)$ only in the group of participants not used to work with robots. This indicates us that the participants who had never worked with a robot before had a greater improvement in their execution time than those who had. This is a first step to validate H3.

For clarity reasons, the answers to four questions of our user study for each Mode are summarized in Table (6.1). With this questionnaire, we observed a significant effect on the Mode. In Mode B, i.e. with the virtual guide, participants found that: (1) they performed better the task; (2) they felt more precise in the execution in the task; and (3) the robot was more helpful than in Mode A. Moreover, for Case B, collisions only occurred when the users were not assisted. These results validate H2. In addition, we can observe that participants found it more easy to work with the robot in Mode B than in Mode A, even if the values have low significance. The lack of statistic relevance between both Modes could be due to the simplicity of the task.

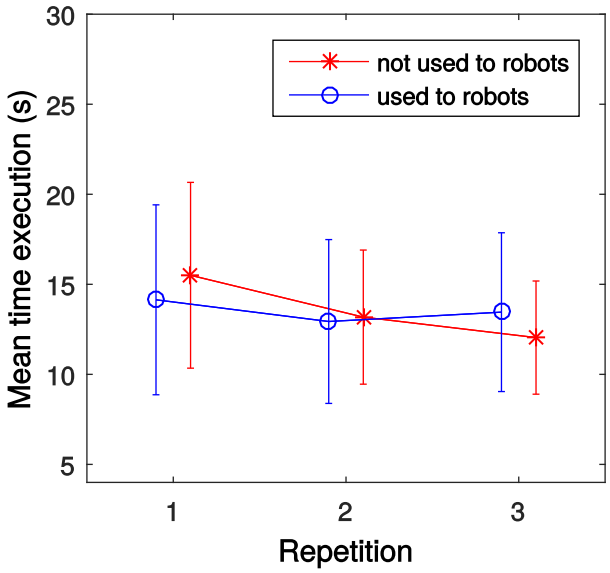


Figure 6.9: Task's execution time. Comparison between repetitions for participants used and not used to work with robots.

However, these results go in the direction of H3.

6.3 Pick and Place task with ISybot robot

We designed a user study to observe: (1) how novice users perceived the virtual guide assistance with multiple guides, (2) to determine if creating new virtual guides through kinesthetic teaching 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/her *personal* set of guides. (tr)

4 The user performs the pick and place task with the guides created in the previous session. (pp_3)

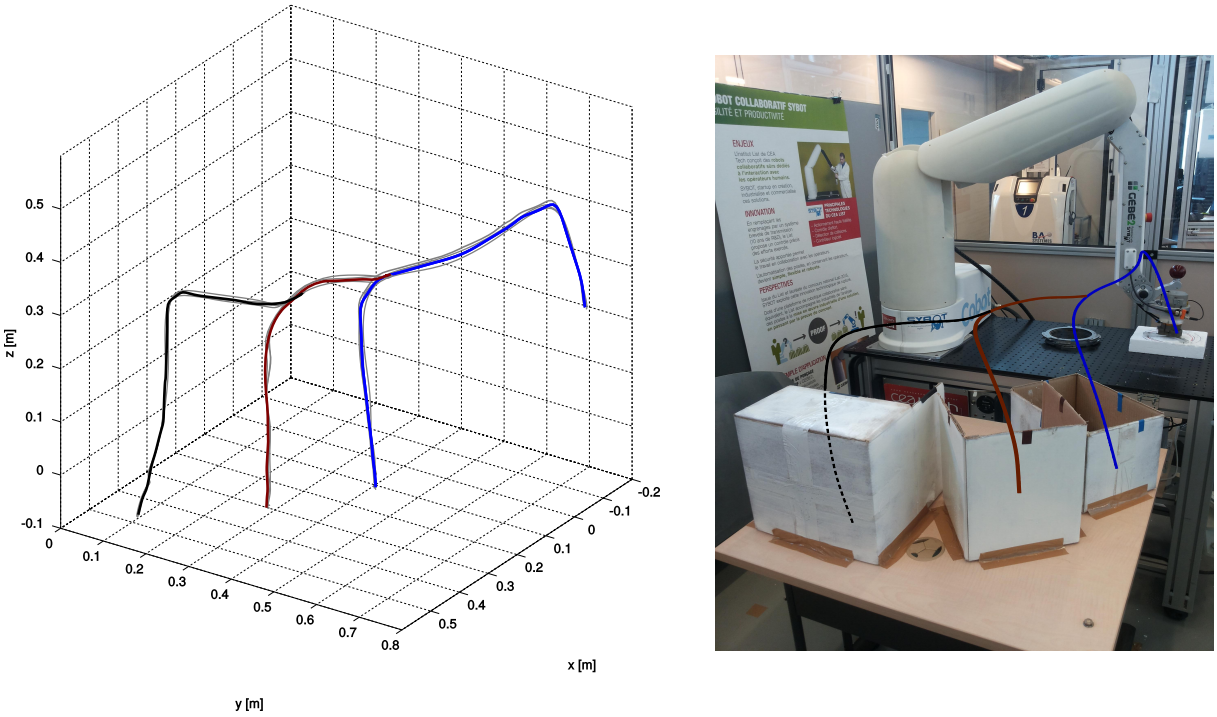


Figure 6.10: Default virtual guides created by the expert user. Left: In gray are shown the demonstrations, while the coloured lines represent the mean of the GMMs. Right: Virtual guides in the robot workspace.

The *default* guides for the session pp_2 were generated by an expert user with the incremental method explained in subsection 3.2.2. For each box only 3 repetitions were needed to create the final (default) guides (Figure 6.10).

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.

6.3.1 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 Figure 6.11. 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 Figure 6.11.

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 Figure 6.11. 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.

Resulting guides from eight different users are shown in Figure 6.12.

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 6.2). For session tr the participants answered to a different survey focusing on the creation of the virtual guides (see Table 6.3).

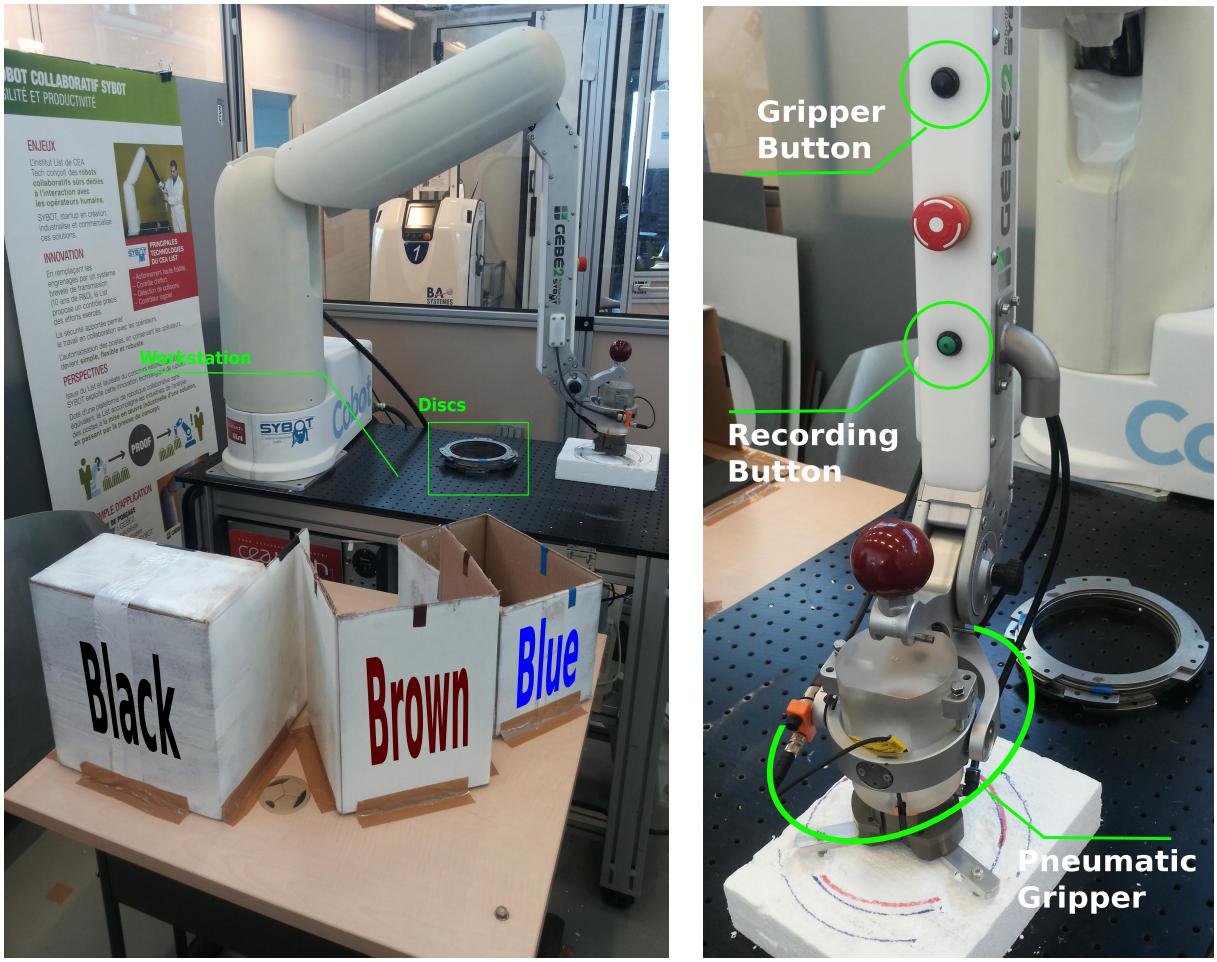


Figure 6.11: Setup for the user study (left) and buttons used for the experiment (right). The upper button was used to hold and release the discs with the pneumatic gripper, while the lower button was used to start and stop the recording of the demonstrations.

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 on T_j, $t_{i,j}$ and the questionnaire on three factors: (1) the sessions pp_j with $j = 1,2,3$, (2) the difficulty, represented by the three boxes (blue; brown;

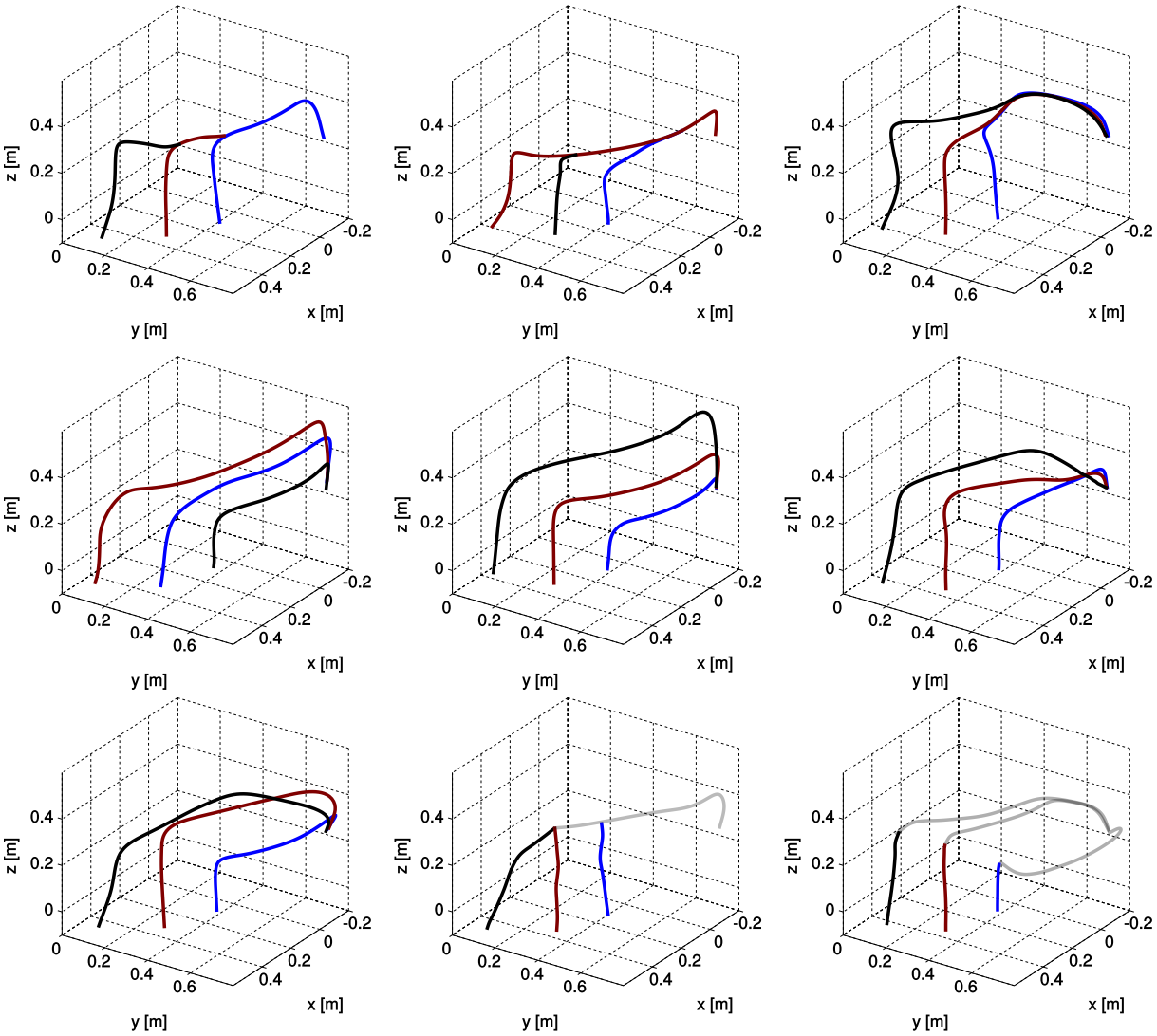


Figure 6.12: Different approaches to the guides creation done by eight of our participants. Blue, brown and black curves are the guides created to place the disc inside the respective box. Gray curves are extra guides created to help connecting the guides. For comparison in the upper-left corner there are the guides created by the expert user.

black) and (3) the repetitions for each box $(r_1; r_2)$. Participants were grouped by their habit to use a robot. Posthoc analyses were performed with Tuckey's HSD test. 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 $p < 0.05$.

6.3.2 Results

1 - Effect of habit to use a robot

No statistical difference was found between participants used to work with robots and those not used to. This could indicate that virtual guides are easy to use because novice users were able to perform as good as users used to robots.

2 - Time Analysis

Effects of sessions on time:

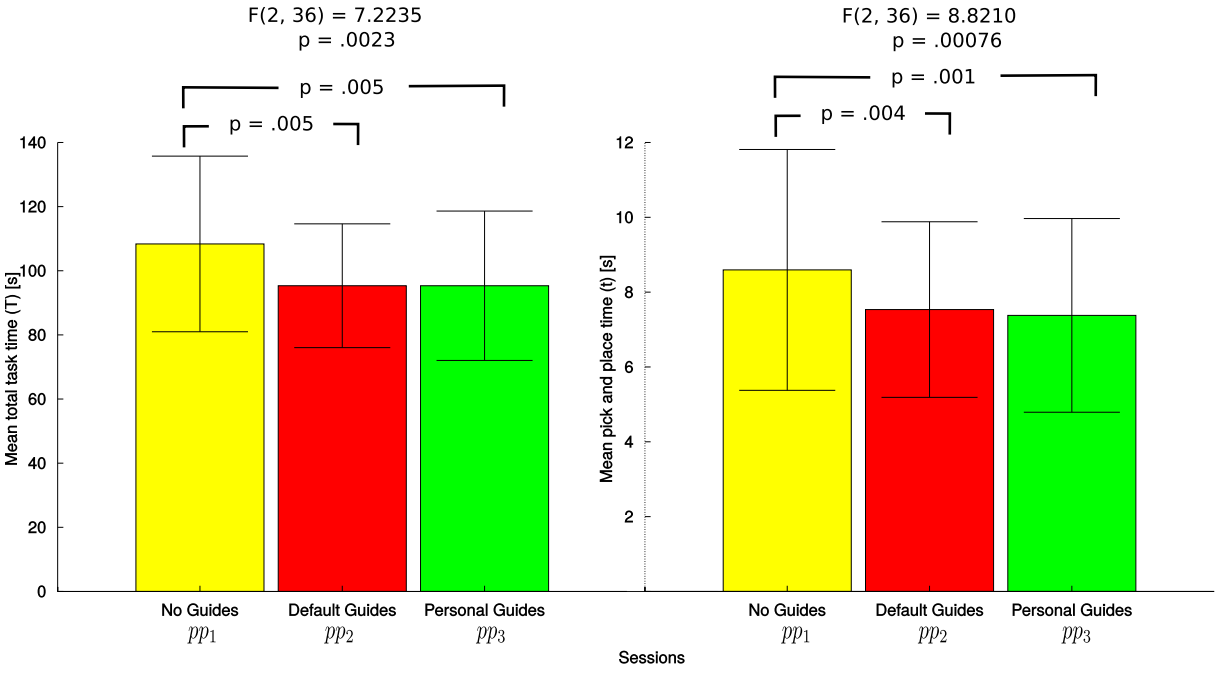


Figure 6.13: Left: Mean of the total task time (T). Right: Mean of pick and place time (t).

We found a significant difference between the sessions $pp_{1,2,3}$ on the total times $T_{1,2,3}$ $(F(2,36) = 7.2235; p = .0023)$. Posthoc analysis shows that $pp_{1,2}$ and $pp_{1,3}$ are statistically different $(p = .005, p = .005)$. In pp_1 the participants were slower than in $pp_{2,3}$ (pp_1: $M = 108.36, SD = 27.38$; pp_2: $M = 95.33, SD = 19.30$; pp_3: $M = 95.32, SD = 23.29$) see Figure 6.13. In addition we found a significant difference between the sessions on t $(F(2,36) = 8.8210; p = .00076)$. Also in this case, the posthoc analysis shows that $pp_{1,2}$ and

$pp_{1,3}$ are statistically different $(p = .004, p = .001)$ respectively. We found again that in pp_1 the participants were slower than in $pp_{2,3}$ (pp_1: $M = 8.59, SD = 3.22$; pp_2: $M = 7.53, SD = 2.3467$; pp_3: $M = 7.38, SD = 2.59$) see Figure 6.13. 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 goes in direction of H4.

Effects of the difficulty on time:

As pointed out in subsection 6.3.1, the difficulty related to the disc insertion is different between

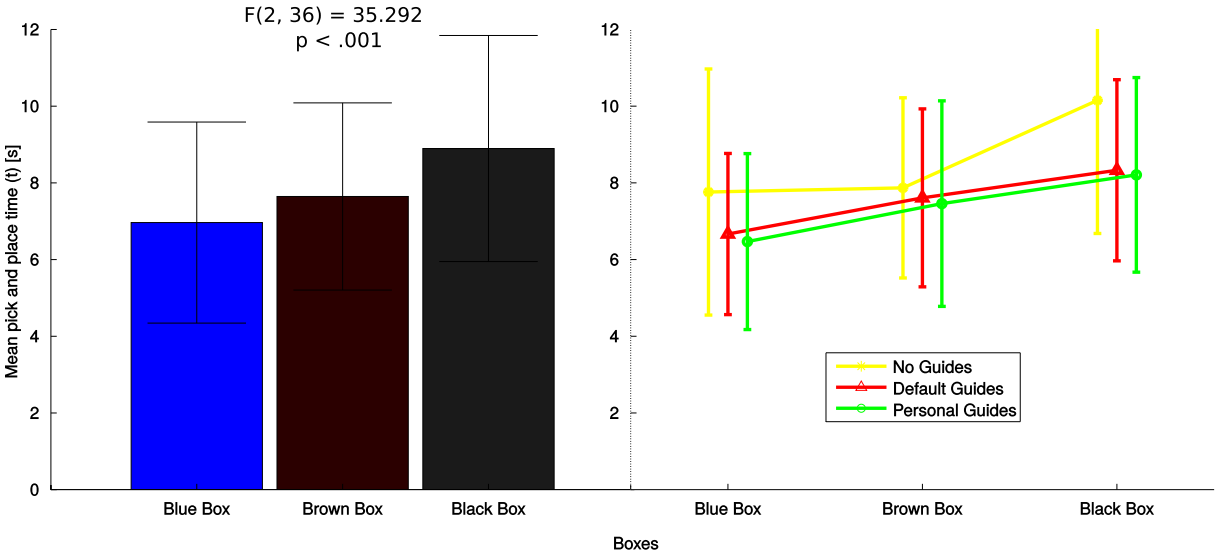


Figure 6.14: Left: Mean of t for each box. Right: Mean of t for each box and each session.

the boxes. This can be seen in Figure 6.14 $(F(2,36) = 35.292; p < .001)$ with (blue: $M = 6.96, SD = 2.62$; brown: $M = 7.64, SD = 2.44$; black: $M = 8.89, SD = 2.95$). 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 the box disposition does not affect the insertion but only the distance does. This

goes in the direction of H2.

Effects of repeated disc insertions on time:

We found a significant effect of the repeated insertions on t ($F(1,18) = 26.665; p < .001$). The

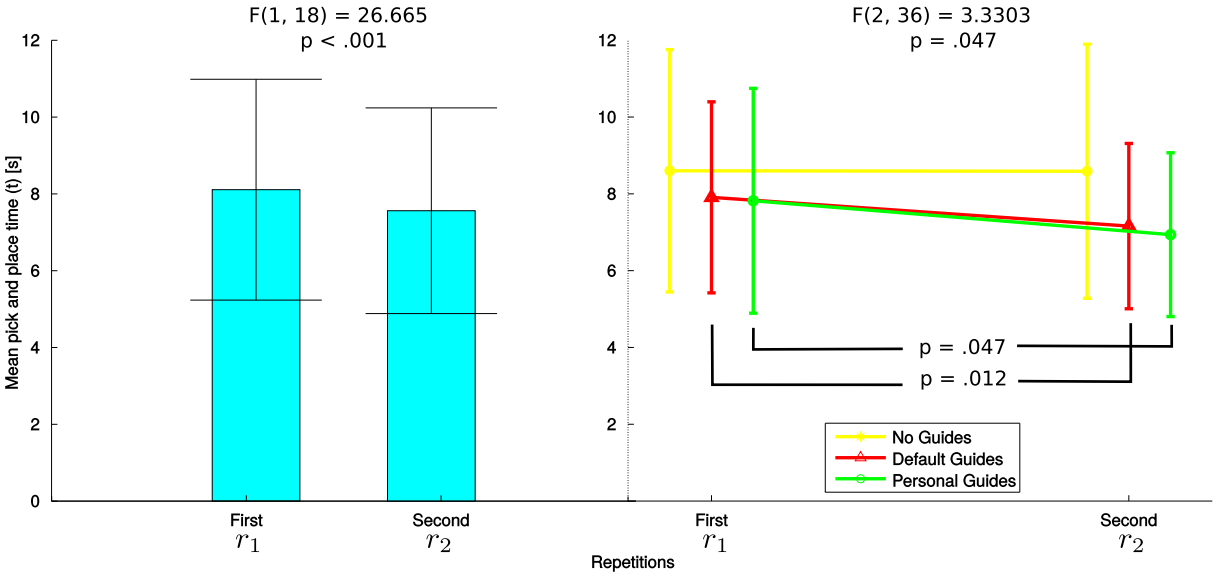


Figure 6.15: Left: Mean of t for the two repetitions. Right: Mean of t for the two repetitions and each session.

second repetition is shorter than the first (r_1: $M = 8.11, SD = 2.87$; r_2: $M = 7.56, SD = 2.68$). This represents a training effect on repeated disc insertions. However we find a significant interaction effect between the sessions and the repetition ($F(2,36) = 3.3303; p = .047$) see Figure 6.15. 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 ($p = .047, p = .012$) respectively. This informs us that with guides there is a training effect: a repetitive use of virtual guides can improve the user performances. This is not necessary to prove H1 but it is a factor to take in account when using virtual guides.

3 - 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 thanks to the guides created by the expert user (Figure 6.10). 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 on the 20 participants did. This indicates that using virtual guides leads to a safer task execution (see Figure 6.16), which goes in the direction of H1. Another observation could be done on the number of collisions for each box. As shown in Figure 6.16 the majority of collisions occurred with the black box when the guides are 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.

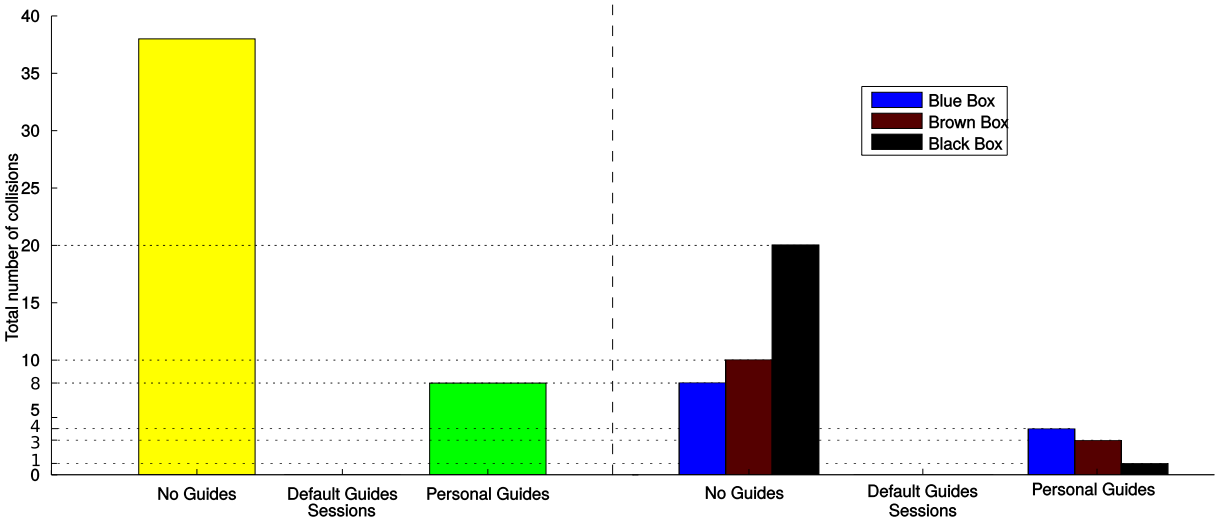


Figure 6.16: Left: Total number of collisions by session. Right: Total number of collisions for each box and session.

4 - Survey on Pick and Place

Table 6.2 and Figure 6.17 show the results of the survey. From them we can observe the following:

1 The task was perceived as easier to perform when using guides (both with default and personal guides).

	$pp1$		$pp2$		$pp3$			
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 the task well?	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 6.2: Survey results for the three pick and place sessions.

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 (Figure 6.10), 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.

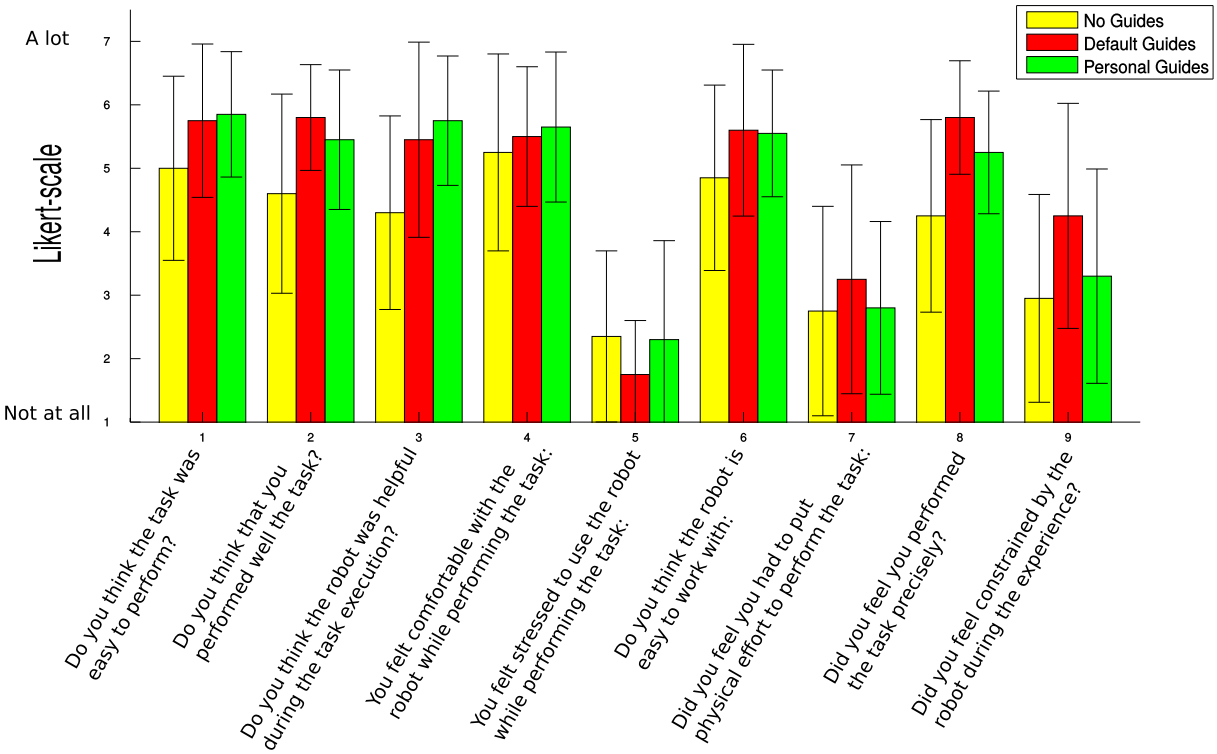


Figure 6.17: Visualization of the survey results for the pick and place sessions.

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. 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. We can see that with their own guides the participants had the feeling that less efforts were necessary. 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.

By looking at the results for questions $1, 2, 3, 6, 8$ we can confirm that virtual guide assistance are perceived as useful by the users, which validates H3.

5 - Survey Training

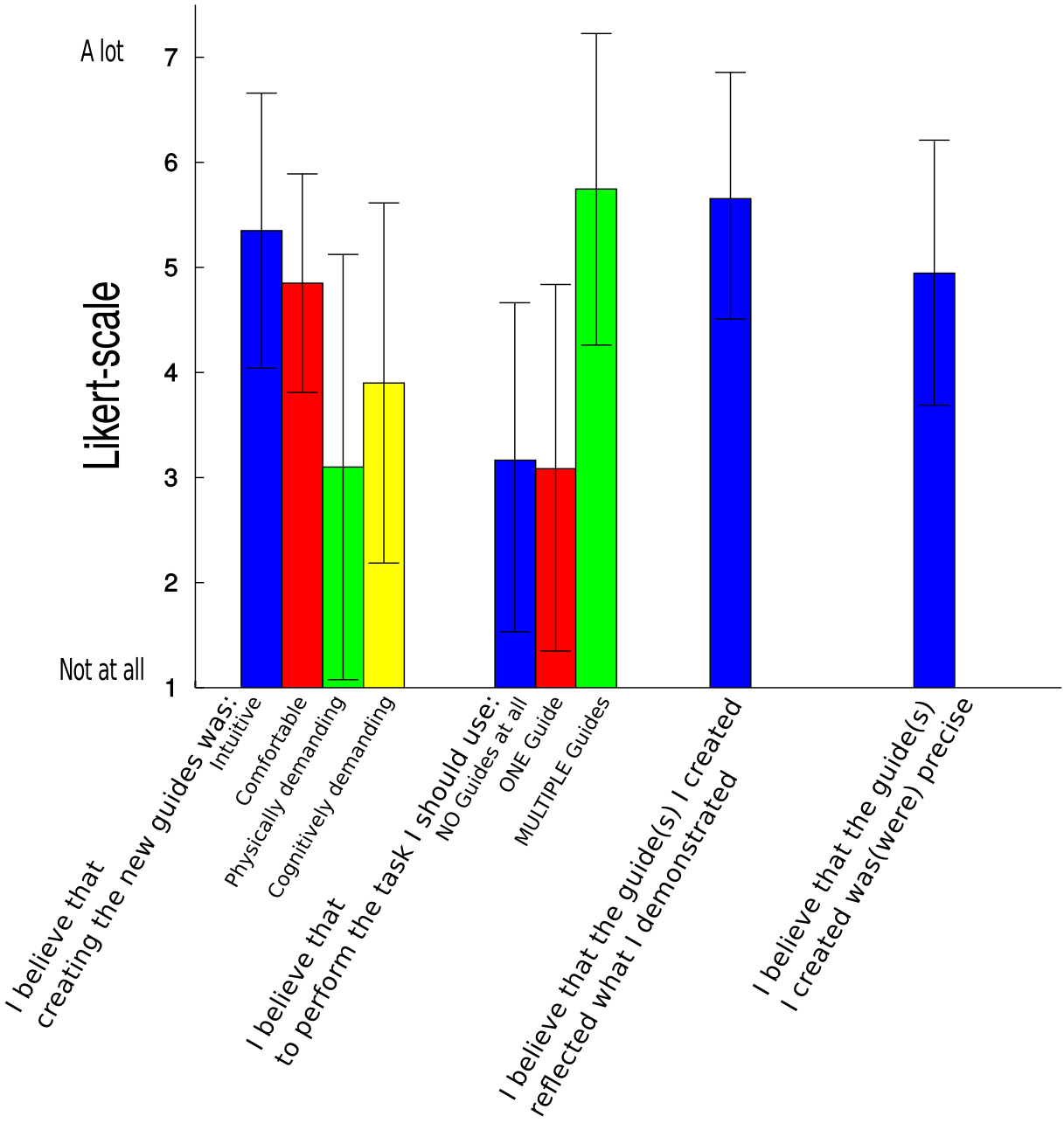


Figure 6.18: Visualization of the survey results for the training session.

In Table 6.3 and Figure 6.18 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

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 6.3: Survey results for the training session.

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.

Notes from the participants:

At the end of the the experiment we invited the participants to comment on the experience. Here we report some of their comments:

- It is intuitive and "ludique" to use and create the guides. The training is very easy to perform but it requires a bit of experience to have precise trajectories.
- In some way, it is difficult to identify the point where I can change from one guide to another. A way to visualize the guides would be useful.
- It would be nice to have augmented reality (AR) glasses to visualize the guides.
- I really liked the experiment. At first is not that intuitive to see the guides, but after a few tries it gets really easy.
- Nice experiment. Specially the training part where we can record the guides, "play" with them, erase them, record again etc. This gives the system a nice felling of flexibility and adaptability.
- I found the robot a little bit "stiff" (as if there was a jump to go from no motion to some motion) when close to the starting position in all three modes. Otherwise it was very easy to use.
- The experience is very interesting. I believe that this type of interface could be helpful to decrease work time when performing important tasks that require a high precision. However, I think that the guides must to be registered by the person who'll use them because you feel more comfortable and sure of what are you doing.
- To create guides is easy and "pratique" but it is hard to be precise. Having multiple trials with the system can increase the precision.

From the comments we can see that the participants would find helpful to have some sort of visualization, moreover some of them pointed out the lack of precision when using their own guides (this is confirmed by the survey answers and the number of collisions) while they felt more precise with the default guides created by the expert user. This is mostly caused by the fact that all the users were novice with the system i.e. they never used prior to the experiment the library of virtual guides.

Chapter 7

Conclusions and Future work

The development of robotics tool 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 our work, 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 (GMM). Thanks to the probabilistic nature of GMM we are able to create guides from multiple demonstrations to reduce the uncertainty residing in human motions. 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 defined a controller that allows the user to use multiple virtual guides in parallel and choose which one is responsible for the task execution. Thanks to the virtual guides definition with GMM, the multiple guides control scheme and three different interaction modes (Hard,Soft,Zero), we are able to create 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 a proof-of-concept experiment with the Meka robot and two simulated industrial tasks with ISybot. We concluded that virtual guides can improve the human performances in terms of time and collisions, and they can relieve the workload from the user.

7.1 Future work

7.1.1 Adaptive stiffness based on uncertainty

One of the main advantages of using probabilistic models such as GMM is the possibility to extract the uncertainty from the demonstrations in the form of covariance matrices. We could think to use this uncertainty to regulate the stiffness of the guides. For example, if high variability is observed, the stiffness can be reduced since the guiding does not need to be precise; vice-versa the stiffness can be increased if the variability is low, so that the guiding is more precise, see the example in (Figure 7.1).

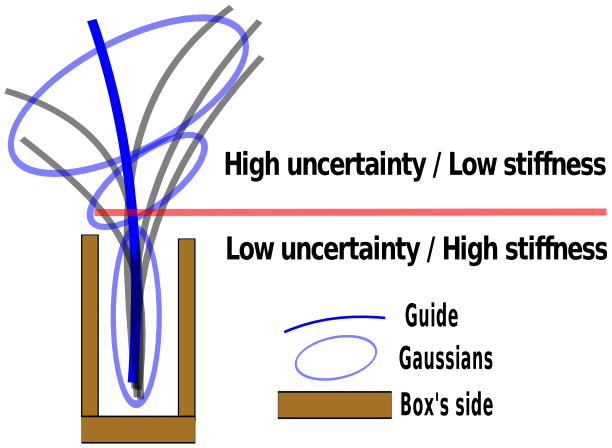


Figure 7.1: Thanks to GMM the uncertainty can be exploited to enforce higher stiffness whereas is needed, for example inside the box in order to avoid collisions with the box's sides. Outside the box the stiffness can be lower to facilitate the approach to it.

Previous work such as [Calinon et al., 2010] and [Calinon et al., 2014] exploited successfully the uncertainty for co-manipulation tasks. In [Calinon et al., 2010] the authors present a way to define the stiffness matrix *proportional* to the inverse of the observed covariance. This method could be used to regulate the stiffness of the virtual guide but it is necessary to the define the relation's boundaries.

For this reason in [Calinon et al., 2014] and similarly in [Hernández et al., 2012] the authors propose to use a LQR (linear quadratic regulator) where the state cost-weight matrix is given by the inverse of the covariance matrix. This regulator is used to estimate the stiffness and damping of the spring-damper systems, resulting in a minimal intervention control strategy.

7.1.2 Active virtual mechanisms

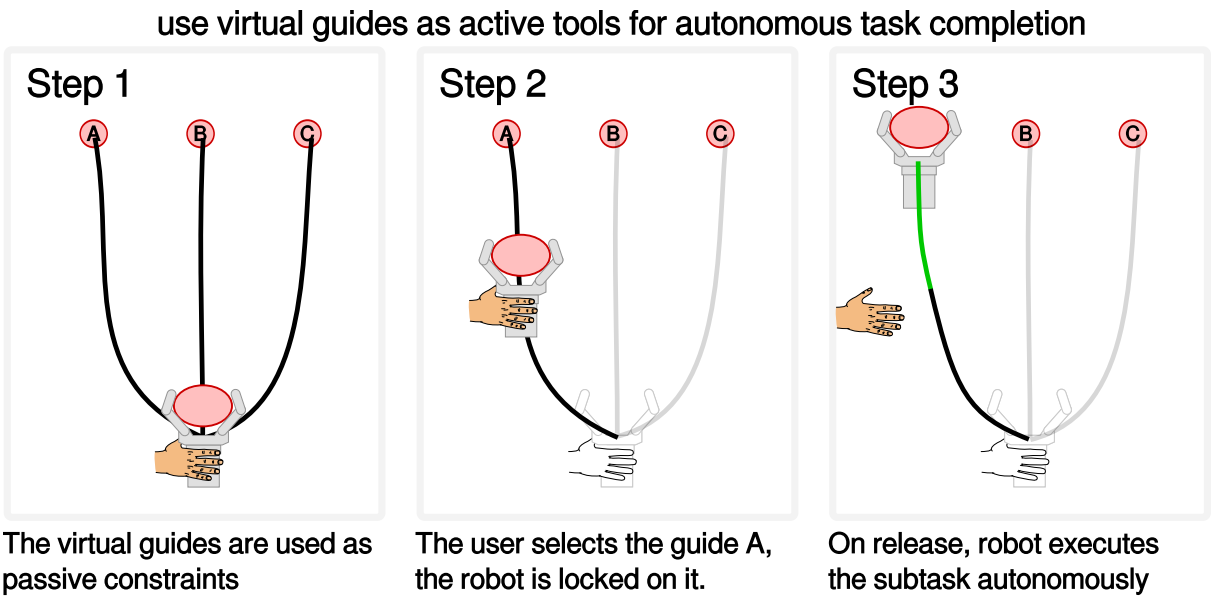


Figure 7.2: The task is composed by three different subtasks where each subtask consists in transporting an object to a different location (A,B,C). Step 1: the guides act as passive constraints. Step 2: the user moves the robot end-effector on the guide dedicated to perform the desired subtask, in this step the guide is still acting as a passive constraint. Step 3: the robot is locked on the desired guide, by releasing the end-effector the robot executes the subtask autonomously.

Virtual guides are particularly useful when the user has to perform a non-structured task. A non-structured task could require a set of different subtasks to be performed in a flexible order. The order can be determined on-the-fly by the user in respect of workspace constraints such as the availability of assembly parts, busy tools or occupied portions of the workspace. We can think to use the library of virtual guides as a library of learnt subtask where the guide is used to communicate to the robot which of the many subtasks has to be executed. By doing so, the goal of the library is to work as a haptic interface for the task selection. A similar concept is presented in [Pistillo et al., 2011] in the context of autonomous task execution. In this work, the user can select a task by moving the robot in proximity of the correspondent task region.

As we presented in our work, the virtual guides act as a passive tool that help the user during the task execution by constraining the robot movement on a specific path, but leaving the task execution to the user. We could think to define a guide that acts as an active tool i.e. the user select one of the multiple guides available in the library thanks to the controller we defined in

chapter 4, when the user is confident that the robot is *locked* on the right guide he/she releases the robot. The robot detects the release, and instead of using the constraint as a virtual guide, the robot autonomously moves along the constraint in order to complete the subtask, (Figure 7.2).

7.1.3 Guide visualization

One problem raised by some participants during the pick and place experiment with ISybot is the lack of visualization for the virtual guides (See the participants notes in section 6.3.2). The absence of visualization makes harder for the user to find where the guides are placed in the robot's workspace.

Figure 7.3: In [Rosenberg, 1993] is implemented one of the first functioning augmented reality systems. Louis Rosenberg demonstrated the benefits to human performances when using virtual fixtures combined with augmented reality.

This problem affects both soft and hard interaction modes. When using the soft interaction mode the user can escape the guides, but the lack of visualization makes it harder to find them back again. When using the hard interaction mode instead, the lack of visualization causes the user to move involuntarily against the guide generating the correction by the spring-damper system that causes the user to exert unnecessary efforts. In fact, the only feedback to the user about the guide position is given by the correction force exerted by the spring-damper system on the robot tool-tip (and indirectly to the user).

The idea of visualizing the guides is not new in literature. The first definition of virtual fixtures already included the visualization of them in order to fully exploit their advantages (Figure 7.3).

Figure 7.4: Hololens for augmented reality. The use of augmented reality would improve the immersion of the user in the robot workspace. For example, the user could be able to visualize a preview of the robot movements before launching their execution.

For these reasons we strongly believe that the interaction with the virtual guides would be improved by using the augmented reality (Figure 7.4). Thanks to augmented reality devices, the user could visualize the guides directly in the robot's workspace, making easier to select them when using the soft and hard interaction mode; the user would be able to see the trajectories related to the virtual guides and to select the guide to use by easily moving the robot in the right spot. Moreover the augmented reality would facilitate the creation and the modification of the virtual guides by increasing the user immersion in the robot workspace.

Bibliography

[Aarno et al., 2005] Aarno, D., Ekvall, S., and Kragic, D. (2005). Adaptive virtual fixtures for machine-assisted teleoperation tasks. In *ICRA*, pages 897–903.

[Abbott, 2005] Abbott, J. J. (2005). *Virtual Fixtures for Bilateral Telemanipulation*. PhD thesis, Johns Hopkins University.

[Abbott and Okamura, 2003] Abbott, J. J. and Okamura, A. M. (2003). Virtual fixture architectures for telemanipulation.

[Akima, 1970] Akima, H. (1970). A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM*, 17(4):589–602.

[Arandjelovic and Cipolla, 2005] Arandjelovic, O. D. and Cipolla, R. (2005). Incremental learning of temporally-coherent gaussian mixture models. In *Proc. BMVC*, pages 59.1–59.10. doi:10.5244/C.19.59.

[Becker et al., 2013] Becker, B. C., Maclachlan, R. A., Lobes, L. A., Hager, G. D., and Riviere, C. N. (2013). Vision-based control of a handheld surgical micromanipulator with virtual fixtures. *IEEE Trans Robot*, 29(3):674–683.

[Ben Amor et al., 2014] Ben Amor, H., Neumann, G., Kamthe, S., Kroemer, O., and Peters, J. (2014). Interaction primitives for human-robot cooperation tasks. In *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)*.

[Bettini et al., 2004] Bettini, A., Marayong, P., Member, S., Lang, S., Okamura, A. M., and Hager, G. D. (2004). Vision assisted control for manipulation using virtual fixtures. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1171–1176.

[Bowyer et al., 2014] Bowyer, S. A., Davies, B. L., and y Baena, F. R. (2014). Active constraints/virtual fixtures: A survey. *IEEE Transactions on Robotics*, 30(1):138–157.

[Bowyer and y Baena, 2013] Bowyer, S. A. and y Baena, F. R. (2013). Dynamic frictional constraints for robot assisted surgery. In *World Haptics Conference (WHC), 2013*, pages 319–324.

[Boy et al., 2007] Boy, E. S., Burdet, E., Teo, C. L., and Colgate, J. (2007). Investigation of Motion Guidance With Scooter Cobot and Collaborative Learning. *IEEE Transactions on Robotics*.

[Burghart et al., 1999] Burghart, C., Keitel, J., Hassfeld, S., Rembold, U., and Woern, H. (1999). Robot controlled osteotomy in craniofacial surgery.

[Calinon, 2007] 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*, pages 255–262.

[Calinon et al., 2014] Calinon, S., Bruno, D., and Caldwell, D. G. (2014). A task-parameterized probabilistic model with minimal intervention control. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 3339–3344, Hong Kong, China.

[Calinon et al., 2007] Calinon, S., Guenter, F., and 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 et al., 2010] Calinon, S., Sardellitti, I., and Caldwell, D. G. (2010). Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In

Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pages 249–254, Taipei, Taiwan.

[Colgate et al., 2003] Colgate, J. E., Peshkin, M. A., and Klostermeyer, S. H. (2003). Intelligent assist devices in industrial applications: a review. In *IROS*, pages 2516–2521.

[David et al., 2014] David, O., Russotto, F.-X., Simoes, M. D. S., and 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 et al., 2006] Davies, B., Jakopec, M., Harris, S. J., Baena, F. R. Y., Barrett, A., Evangelidis, A., Gomes, P., Henckel, J., and Cobb, J. (2006). Active-constraint robotics for surgery. *Proceedings of the IEEE*, 94(9):1696–1704.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38.

[Dumora, 2014] Dumora, J. (2014). *Contribution à l'interaction physique homme-robot: application à la comanipulation d'objets de grandes dimensions*. PhD thesis, Montpellier 2.

[Held and Bov, 2013] Held, L. and Bov, D. S. (2013). *Applied Statistical Inference: Likelihood and Bayes*. Springer Publishing Company, Incorporated.

[Hermann et al., 2016] Hermann, M., Pentek, T., and Otto, B. (2016). Design principles for industrie 4.0 scenarios. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 3928–3937. IEEE.

[Hernández et al., 2012] Hernández, J. M., Lee, D., and Hirche, S. (2012). Risk-sensitive optimal feedback control for haptic assistance. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[Ho et al., 1995] Ho, S. C., Hibberd, R. D., and Davies, B. L. (1995). Robot assisted knee surgery. *IEEE Engineering in Medicine and Biology Magazine*, 14(3):292–300.

[Hogan, 1988] Hogan, N. (1988). On the stability of manipulators performing contact tasks. *IEEE Journal on Robotics and Automation*, 4(6):677–686.

[Huber et al., 2008] Huber, M. F., Bailey, T., Durrant-Whyte, H., and Hanebeck, U. D. (2008). On entropy approximation for gaussian mixture random vectors. In *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pages 181–188.

[Joly and Andriot, 1995] Joly, L. and 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*, volume 1, pages 357–362 vol.1.

[Khalil and Grizzle, 1996] Khalil, H. K. and Grizzle, J. (1996). *Nonlinear systems*, volume 3. Prentice hall New Jersey.

[Kuang et al., 2004] Kuang, A., Payandeh, S., Zheng, B., Henigman, F., and 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*, pages 367–374.

[Li and Okamura, 2003] Li, M. and Okamura, A. M. (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*, pages 125–131.

[Maimon and Rokach, 2005] Maimon, O. and Rokach, L. (2005). *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Marayong et al., 2003] Marayong, P., Li, M., Okamura, A. M., and Hager, G. D. (2003). Spatial motion constraints: theory and demonstrations for robot guidance using virtual fixtures. In *ICRA*, pages 1954–1959. IEEE.

[Mollard et al., 2015] Mollard, Y., Munzer, T., Baisero, A., Toussaint, M., and Lopes, M. (2015). Robot programming from demonstration, feedback and transfer. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1825–1831.

[Nolin et al., 2003] Nolin, J. T., Stemniski, P. M., and Okamura, A. M. (2003). 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.

[Pezzementi et al., 2007] Pezzementi, Z., Hager, G. D., and Okamura, A. M. (2007). Dynamic guidance with pseudoadmittance virtual fixtures. In *IEEE International Conference on Robotics and Automation*, pages 1761–1767.

[Pistillo et al., 2011] Pistillo, A., Calinon, S., and Caldwell, D. G. (2011). Bilateral physical interaction with a robot manipulator through a weighted combination of flow fields. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3047–3052. IEEE.

[Pratt and Williamson, 1995] Pratt, G. A. and Williamson, M. M. (1995). 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.

[Raiola et al., 2015a] Raiola, G., Lamy, X., and Stulp, F. (2015a). Co-manipulation with Multiple Probabilistic Virtual Guides. In *International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany.

[Raiola et al., 2015b] Raiola, G., Rodriguez-Ayerbe, P., Lamy, X., Tliba, S., and Stulp, F. (2015b). Parallel guiding virtual fixtures: Control and stability. In *IEEE Multi-Conference on Systems and Control (MSC)*.

[Rosenberg, 1993] Rosenberg, L. (1993). Virtual fixtures: perceptual tools for telerobotic manipulation. In *Proc. IEEE Virtual Reality International Sympsoium*.

[Rozo et al., 2016] Rozo, L., Calinon, S., Caldwell, D. G., Jiménez, P., and Torras, C. (2016). Learning physical collaborative robot behaviors from human demonstrations. *IEEE Transactions on Robotics*, PP(99):1–15.

[Ryden et al., 2013] Ryden, F., Stewart, A., and Chizeck, H. (2013). Advanced telerobotic underwater manipulation using virtual fixtures and haptic rendering. In *Oceans - San Diego, 2013*, pages 1–8.

[Sakoe and Chiba, 1978] Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49.

[Sanchez Restrepo et al., 2017] Sanchez Restrepo, S., Raiola, G., Chevalier, P., Lamy, X., and Sidobre, D. (2017). Iterative virtual guides programming for comanipulation robots. In *Under review: Proceedings of 2017 IEEE International Conference on Robotics and Automation (ICRA)*.

[Siciliano et al., 2009] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer.

[Song and Wang, 2005] Song, M. and Wang, H. (2005). Highly efficient incremental estimation of gaussian mixture models for online data stream clustering. In Priddy, K. L., editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 5803 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 174–183.

[Stulp et al., 2014] Stulp, F., Herlant, L., Hoarau, A., and Raiola, G. (2014). Simultaneous online discovery and improvement of robotic skill options. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*.

[Stulp et al., 2013] Stulp, F., Raiola, G., Hoarau, A., Ivaldi, S., and Sigaud, O. (2013). Learning compact parameterized skills with a single regression. In *IEEE-RAS International Conference on Humanoid Robots*.

[Tonietti et al., 2005] Tonietti, G., Schiavi, R., and Bicchi, A. (2005). 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.

[Tykal et al., 2016] Tykal, M., Montebelli, A., and Kyrki, V. (2016). Incrementally assisted kinesthetic teaching for programming by demonstration. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 205–212.

[Vakanski et al., 2012] Vakanski, A., Mantegh, I., Irish, A., and 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.

[Van Damme et al., 2010] Van Damme, M., Beyl, P., Vanderborght, B., Van Ham, R., Vanderniepen, I., Matthys, A., Cherelle, P., and Lefeber, D. (2010). 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.

[WorldRobotics, 2016] WorldRobotics (2016). World robotics. www.ifr.org/.

[Xu and Jordan, 1995] Xu, L. and Jordan, M. I. (1995). On convergence properties of the em algorithm for gaussian mixtures. *Neural Computation*, 8:129–151.

[Yoon et al., 2014] Yoon, H., Wang, R., and 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*, pages 625–630.

[Yu et al., 2005] Yu, W., Alqasemi, R., Dubey, R., and 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*, pages 1121–1126.

université PARIS-SACLAY

ÉCOLE DOCTORALE
INTERFACES
Approches Interdisciplinaires : Fondements, Applications et Innovation

Titre : Co-manipulation avec une bibliothèque de Guides Virtuels
Mots clés : Robotique, contrôles, apprentissage automatique, software engineering

Résumé: Les robots ont un rôle fondamental dans la fabrication industrielle. Non seulement ils augmentent l'efficacité et la qualité des lignes de production, mais aussi diminuent considérablement la charge de travail des humains. Cependant, en raison des limites des robots industriels en termes de flexibilité, de perception et de sécurité, Leur utilisation est limitée à un environnement structuré bien connu. En outre, il n'est pas toujours rentable d'utiliser des robots autonomes industriels dans de petites usines à faibles volumes de production. Cela signifie que des travailleurs humains sont encore nécessaires dans de nombreuses chaînes d'assemblage pour exécuter des tâches spécifiques. Par conséquent, ces dernières années, une grande impulsion a été donnée à la co-manipulation homme-robot. En permettant aux humains et aux robots de travailler ensemble, il est possible de combiner les avantages des deux; La compréhension des tâches abstraites et la perception robuste typique d'un être humain avec la précision et la force d'un robot industriel.

Une approche réussie pour faciliter la co-manipulation homme-robot, est l'approche de guides virtuels qui contraint le mouvement du robot sur seulement certaines trajectoires pertinentes. Le guide virtuel ainsi réalisé agit comme un outil passif qui améliore les performances de l'utilisateur en termes de temps de tâche, de charge de travail mentale et d'erreurs.
L'aspect innovant de notre travail est de présenter une bibliothèque de guides virtuels qui permet à l'utilisateur de facilement sélectionner, générer et modifier les guides grâce à une interaction intuitive haptique avec le robot.
Nous avons démontré, dans deux tâches industrielles, que ces innovations fournissent une interface novatrice et intuitive pour l'accomplissement des tâches par les humains et les robots.

Title : Co-manipulation with a library of Virtual Guides
Keywords : robotics, controls, machine learning , software engineering

Abstract: Robots have a fundamental role in industrial manufacturing. They not only increase the efficiency and the quality of production lines, but also drastically decrease the work load carried out by humans. However, due to the limitations of industrial robots in terms of flexibility, perception and safety, their use is limited to well-known structured environment. Moreover, it is not always cost-effective to use industrial autonomous robots in small factories with low production volumes. This means that human workers are still needed in many assembly lines to carry out specific tasks. Therefore, in recent years, a big impulse has been given to human-robot co-manipulation. By allowing humans and robots to work together, it is possible to combine the advantages of both; abstract task understanding and robust perception typical of human beings with the accuracy and the strength of industrial robots.

One successful method to facilitate human-robot co-manipulation, is the Virtual Guides approach which constrains the motion of the robot along only certain task-relevant trajectories. The so realized virtual guide acts as a passive tool that improves the performances of the user in terms of task time, mental workload and errors.
The innovative aspect of our work is to present a library of virtual guides that allows the user to easily select, generate and modify the guides through an intuitive haptic interaction with the robot.
We demonstrated in two industrial tasks that these innovations provide a novel and intuitive interface for joint human-robot completion of tasks.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France