Using Reinforcement Learning for Optimizing the Reproduction of Tasks in Robot Programming by Demonstration

THÈSE N^{O} 4311 (2009)

PRÉSENTÉE LE 23 FÉVRIER 2009

À LA FACULTÉ SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE D'ALGORITHMES ET SYSTÈMES D'APPRENTISSAGE
PROGRAMME DOCTORAL EN SYSTÈMES DE PRODUCTION ET ROBOTIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Florent GÜNTER

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2009

Abstract

As robots start pervading human environments, the need for new interfaces that would simplify human-robot interaction has become more pressing. Robot Programming by Demonstration (RbD) develops intuitive ways of programming robots, taking inspiration in strategies used by humans to transmit knowledge to apprentices. The user-friendliness of RbD is meant to allow lay users with no prior knowledge in computer science, electronics or mechanics to train robots to accomplish tasks the same way as they would with a co-worker.

When a trainer teaches a task to a robot, he/she shows a particular way of fulfilling the task. For a robot to be able to learn from observing the trainer, it must be able to learn what the task entails (i.e. answer the so-called "*What-to-imitate?*" question), by inferring the user's intentions. But most importantly, the robot must be able to adapt its own controller to fit at best the demonstration (the so-called "*How-to-imitate?*" question) despite different setups and embodiments. The latter is the question that interested us in this thesis. It relates to the problem of optimizing the reproduction of the task under environmental constraints. The "*How-to-imitate?*" question is subdivided into two problems.

The first problem, also known as the "*correspondence problem*", relates to resolving the discrepancy between the human demonstrator and robot's body that prevent the robot from doing an identical reproduction of the task. Even tough we helped ourselves by considering solely humanoid platforms, that is platforms that have a joint configuration similar to that of the human, discrepancies in the number of degrees of freedom and range of motion remained. We resolved these by exploiting the redundant information conveyed through the demonstrations by collecting data through different frames of reference. By exploiting these redundancies in an algorithm comparable to the damped least square algorithm, we are able to reproduce a trajectory that minimizes the error between the desired trajectory and the reproduced trajectory across each frame of reference.

The second problem consists in reproducing a trajectory in an unknown setup while respecting the task constraints learned during training. When the informations learned from the demonstration no longer suffice to generalize the task constraints to a new set-up, the robot must re-learn the task; this time through trial-and-error. Here we considered the combination of trial-and-error learning to complement RbD. By adding a trial-and-error module to the original Imitation Learning algorithm, the robot can find a solution that is more adapted to the context and to its embodiment than the solution found using RbD. Specifically, we compared Reinforcement Learning (RL) - to other classical optimization techniques.

We show that the system is advantageous in that: a) learning is more robust to unexpected events that have not been encountered during the demonstrations and b) the robot is able to optimize its own model of the task according to its own embodiment.

Keywords:

Robot programming by demonstration (RbD), Humanoid Robots, Imitation Learning, Trial-and-Error Learning, Reinforcement Learning (RL).

Résumé

De nos jours, les robots commencent à sortir des milieux industriels et font leur apparition dans la vie de tous les jours. Pour pouvoir simplifier l'utilisation et l'interaction avec ces robots, le besoin de créer de nouvelles interfaces de programmation se fait ressentir. De nouvelles solutions de programmation plus intuitives que les langages de programmation classiques font leur apparition. Ces solutions sont inspirées par les stratégies d'imitation utilisées chez les hommes et chez les animaux pour permettre la transmission du savoir. En robotique, ce domaine est appelé "programmation par démonstration". Le but est de pouvoir donner l'accès à ces technologies à des personnes sans aucune connaissance en informatique, électronique ou mécanique afin de les aider pour de petites tâches dans la vie de tous les jours. Montrer physiquement à un robot comment il doit faire pour réaliser une certaine tâche est certainement la solution la plus simple et la plus accessible pour programmer ce robot. Afin de réaliser cela, le robot ne doit pas seulement imiter, mais il doit aussi adapter sa façon de faire à l'environnement et aux besoins de l'utilisateur.

Lorsqu'un utilisateur veut enseigner au robot une nouvelle tâche, il/elle montre au robot différentes manières de réaliser cette tâche. Le robot doit alors être capable de reproduire la tâche dans différentes situations. Pour cela, il y a deux questions principales qu'il doit résoudre, "*qu'est-ce qu'il faut imiter?*" et "*comment est-ce qu'il faut imiter?*". La première question est reliée à la problématique de déterminer, à partir des démonstrations, quelles sont les contraintes indispensables à l'exécution de la tâche, tandis que la deuxième question est reliée à la problématique de trouver une solution pour reproduire la tâche correctement dans différentes situations et malgré les différences de morphologie tout en respectant ces contraintes.

Le travail effectué dans le cadre de cette thèse consiste à résoudre la seconde question: "*comment est-ce qu'il faut imiter?*". Etant donné les contraintes extraites d'un set de démonstration, différentes techniques sont utilisées afin de créer un algorithme robuste qui soit capable de générer une reproduction satisfaisante de la tâche dans différentes situations.

Le problème défini par la question "*comment est-ce qu'il faut imiter?*" peut-être séparé en deux parties. La première partie consiste à trouver une solution pour résoudre les problèmes posés par la différence de morphologie entre le démonstrateur (l'être humain) et l'imitateur (le robot). Ce problème est connu sous le nom de "problème de correspondance". La première partie de la solution à ce problème est donnée directement par la conception mécanique du robot. En effet, en utilisant des robots humanoïdes dont la cinématique est la plus proche possible de la cinématique de l'être humain, nous réduisons grandement la différence de morphologie et donc nous facilitons le transfert des données du mouvement entre l'être humain et le robot. La seconde partie de la solution est apportée par la combinaison des données enregistrées dans différents référentiels durant les démonstrations. La redondance de ces données est exploitée dans un algorithme de cinématique inverse inspiré par l'algorithme des moindres carrés amortis. Cette algorithme génère une trajectoire en minimisant l'erreur entre la trajectoire désirée et la trajectoire effectivement reproduite par le robot à travers les différents référentiels.

La seconde partie du problème consiste à trouver une manière de généraliser la reproduction de la tâche apprise afin que le robot puisse la reproduire en respectant les contraintes dans différentes situations et même dans des situations qui n'auraient pas été démontrées lors de la phase d'apprentissage. Dans le domaine de la programmation de robots par démonstration, l'idée est que le robot apprenne la tâche avec un instructeur. Mais il existe aussi d'autres techniques, comme par exemple l'apprentissage par renforcement, qui permettent au robot d'apprendre une tâche par lui-même sans interventions extérieures. En utilisant l'apprentissage par renforcement, même si l'apprentissage et un peu plus long, la solution trouvée par le robot est plus adaptée à sa propre morphologie. Dans ce travail, la solution proposée est de combiner les deux méthodes d'apprentissages afin d'aboutir à un algorithme plus robuste. En ajoutant un module constitué d'un algorithme d'apprentissage par renforcement à l'algorithme d'apprentissage par imitation de base, nous donnons la possibilité au robot de réapprendre et d'optimiser son propre modèle de la tâche à effectuer.

Les avantages du système sont: a) le robot a la possibilité de réagir correctement s'il se trouve en face d'une situation qu'il n'a pas observé durant les démonstrations et b) il a aussi la possibilité d'optimiser la reproduction de la tâche en tenant compte des contraintes posées par sa propre morphologie.

Mots clés:

Programmation par démonstration, robots humanoïdes, apprentissage par imitation, apprentissage par renforcement.

Acknowledgments

First of all, I would like to thank prof. Aude Billard for having offered to me the great opportunity to accomplish my PhD in the Learning Algorithms and Systems Laboratory. My time at the LASA was full of enjoyable and rich experiences. I had the opportunity to present my work at various conferences around the world and I have learned a lot about managing research projects through supervision of student projects and through the management of my own project. I have especially appreciated her support, her advices and her guidance for my research work and I also want to thank her for her help in writing my different papers and especially in the redaction of my PhD thesis.

I would like to acknowledge the members of my PhD committee, Prof. Philippe Gaussier, Dr. Tamim Asfour, Dr. Frédéric Kaplan and Prof. Max-Olivier Hongler for their availability and for the time they spent to read my PhD thesis.

Many thanks to my colleagues at the LASA laboratory for their kind help and collaboration during these years of intensive research as well as their friendship during the moments of relaxation.

I also would like to thank my family and friends that always supported me during my PhD even in the most difficult times. I would like especially to thank Carlos and Ursina for their help in checking the English of my PhD thesis manuscript.

Finally, thanks to the Ecole Polytechnique Fédérale de Lausanne for offering me the opportunity to work in a great scientific environment.

The financial support of my work was provided by the EU Integrated Project COGNIRON (”The Cognitive Companion”) and funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

Table of Contents

List of Abbreviations

DOF	Degree Of Freedom
DS	Dynamical System
DTW	Dynamic Time Warping
EM	Expectation-Maximization
GMM	Gaussian Mixture Model
GMR	Gaussian Mixture Regression
HMM	Hidden Markov Model
HRI	Human-Robot Interaction
ICA	Independent Component Analysis
IK	Inverse Kinematics
MDP	Markov Decision Process
NAC	Natural Actor Critic
OA	Orthogonal Arrays
OR	Operations Research
PbD	Programming by Demonstration
PCA	Principal Component Analysis
RbD	Robot Programming by Demonstration
RBF	Radial Basis Function neural network
RL	Reinforcement Learning
S/N	Signal/Noise ratio
SME	Small and Medium Enterprise
SPI	Serial Peripheral Interface
VITE	Vector Integration To Endpoint

List of Symbols

$(\cdot)^{+}$	Transpose
$(\cdot)^{-1}$	Inverse
$(\cdot)^{+}$	Moore-Penrose Pseudo-Inverse
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{J}$	Jacobian matrix
α, β	Proportionality coefficients
$\boldsymbol{x}$	Data of trajectories in the Cartesian referential frame
$\boldsymbol{y}$	Data of trajectories in the object referential frame
$\boldsymbol{\theta}$	Data of trajectories in the robot joint angles referential frame
$\boldsymbol{\xi}$	Data of trajectories in undefined referential frame
m	Dimensionality of the Cartesian referential frame
n	Dimensionality of the joint angle referential frame
t	Time step
T	Number of time steps
q	Iteration step of the self-learning algorithms
d	Dimensionality of the GMM
k	Current state of the GMM
K	Number of states of the GMM
ν	Vector of variable used to train the GMM
π	Weighting factor of the states of the GMM
μ	Mean of the states of the GMM
Σ	Variance of the states of the GMM
p	Probability density function
α^{e}	Parameter that defines the position of the robot elbow in the geometric IK
H	Cost function
w	Weight of a cost function component

$\boldsymbol{w}$	Matrix of weight of a cost function component - also used as approximation of the natural gradient in the NAC algorithm
$\boldsymbol{\theta}^c$	Joint angles trajectory reproduced from the demonstration data of Cartesian space using IK
$\boldsymbol{\theta}^d$	Joint angle trajectory reproduced from the demonstration data of the joint angles space
γ	Coefficient - or function - used to balance data of two different referential frame
τ	Parameter to control the balancing function γ
γ^z	Value of γ for $\widehat{\frac{dH(\gamma)}{d\gamma}} = 0$
Λ	Approximation of the second derivative of $H(\gamma)$
λ	Lagrange multipliers
$\boldsymbol{\xi}^{demo}$	Demonstrated ξ
$\boldsymbol{\xi}^d$	$\boldsymbol{\xi}$ desired
$\boldsymbol{\xi}^r$	$\boldsymbol{\xi}$ reproduced
$\boldsymbol{\xi}^s$	$\boldsymbol{\xi}$ simulated
$\boldsymbol{\xi}^m$	Modulation data
$\boldsymbol{\xi}^g$	Goal position
ρ	Policy used for the RL algorithms
r	Reward function
f	Fitness function
$\ddot{\boldsymbol{\xi}}^{DS}$	Acceleration component brought by the DS
$\ddot{\boldsymbol{\xi}}^{A}$	Acceleration component brought by the modulation trajectory
$\ddot{\boldsymbol{\xi}}^{R}$	Acceleration component brought by the repulsive obstacle

1 Introduction

1.1 Robotics

While the concept of automation has been known for many centuries, the term of robot appeared for the first time at the beginning of the twentieth century in the play R.U.R (Rossum's Universal Robots) by the Czech writer Karel Capek. The word "robot" is derived from the Czech word "Robota" which means serf labor. The play begins in a factory that produces "artificial people", what we would call "Androids" today.

It is difficult to give a definition of what a robot is and which system can be designated as robot or not. The definition can vary from a country to another. It is generally admitted that a robot is an artificial reprogrammable system that is able to interact with its environment through sensors and mechanical actuators. The International Organization for Standardization has introduced the following definition for robots:

"An automatically controlled, reprogrammable, multipurpose, manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications."

It is interesting to see that this definition defines robots as reprogrammable machines used for industrial application only. This definition discriminates the automatons, that are conceived especially for one application, from robots that can be reprogrammed for different uses in industrial fields. In the same way, by using this definition, tele-operating systems are not considered as robots. But what can we say about tele-operating robots that have a part of autonomy like for example the Mars Rover (except the fact that there is no industrial automation applications for the Mars Rover). There is such an evolution in the robotic field that the standard definition seems to be out of date.

The first industrial robots appeared in the U.S market in 1962. These robots were firstly exclusively used in the automobile industry but appeared little by little in other industries like microtechnology, metallurgy, food and many more (*Handbook of Industrial Robotics*, 1999). Some advantages of industrial robots on human operators are that robots are able to accomplish repetitive task quicker, with a greater precision and without any fatigue. Because industrial robots are easily replaceable, they can also be used in environments in which the security is not optimal for the human operator, like for example toxic environments. However, industrial robots also have disadvantages. Usually, the installation and programming is difficult and requires specialized technicians. Moreover, the environment must be completely determined and controlled because industrial robots are usually unable to handle unexpected events.

The last twenty years, the number of domains where robotic systems are used greatly increased. Now, the well established domains are mainly the automobile, mechanics, electronics, food and textile where robots are used for tasks such as painting, soldering, manipulation, assembling, cutting... Robotic systems are currently in development and begin to be used in various other domains such as chemistry, micro-biology or microtechnology for tasks such as manipulation and micro-manipulation (Dafflon, 2008). In the transportation domain we find some mobile and autonomous robots in big harbors like Rotterdam for the transportation of containers. We also find new robots in the domestic domain such as autonomous vacuum-cleaner robots or lawn-mower robots (Sewan, 2004; Smith, Campbell, & Morton, 2005). A lot of other applications are currently studied and tested. Space robots are developed to explore Mars or accomplish some repairing or installing tasks on orbital station (Pedersen et al., 2003; Ambrose et al., 2000). In medicine, some robots are developed to help handicapped people to recover some lost capability (locomotion for example) and to help surgeons during operations or for training (Baumann, 1997; Stauffer et al., 2007).

Another growing domain that is currently explored is called Social Robotics (Dautenhahn & Christaller, 1996; Kuniyoshi, 1994). Social Robotics is a generic name to designate service robots that interact directly with human users. They can be for example robots designed to help handicapped or elderly people, guiding robots or post delivery robots. There are a lot of very challenging problems to solve in order to develop a real social robot prototype. The first problem is security. In industrial robotics, the robots work usually inside secure areas in which human workers are not allowed during the operations. With social robots, there is the idea of direct interaction between the human user and the robot. It means that the robot should not be stopped when a potential user wants to approach, it must continue its operation while taking care not to injure the user. Another problem is that, due to their function, social robots have to work in a dynamic and uncontrolled environment. Unlike industrial robots, social robots will have to work in an environment that is not specifically designed. Thus, they will have to be robust to all unexpected events that can occur in such an environment without creating any dangerous situation for human beings. The last problem is that those robots must be adapted to be used by unexperienced people. It means that everybody should be able to use the robot without any advanced programming skills. The interface of the robots must be as intuitive as possible.

1.1.1 Humanoid Robotics

The idea of creating human-like machines is much older than the current concept of robotics. Famous examples are the knight automaton designed by Leonardo Da Vinci (Rosheim, 2006) around year 1495 and more recently the automatons of Pierre Jacquet-Droz, the Musician, the Drawer and the Writer. The last one was the most complex, it was considered by certain people as the oldest example of a computer. It was able to write any text of 40 characters. The texts were encoded in a wheel that actuated a complex system of cams.

The motivations to develop humanoid robots are three folds. (1) If robots have to work in an environment that has been designed for human beings, humanoid robots - robots that by definition have the shape of human beings - have the most indicated shape for that purpose. (2) If robots have to interact with humans, it is appropriate to use a humanoid robot, because the human user can interpret its motion and infer its intentions much more easily than with other robots. It will thus be much more convenient for a human user to communicate with a robot that has a human shape and the transfer of skill will be much easier. (3) Humanoid robot development is also useful to learn more about the human being itself, as the body is the tool that allow the interaction between a subject and its environment. Thus, building humanoid robots allow scientists to verify theories on human locomotion and control by implementing bio-inspired algorithms in real robots (Pfeifer & Scheier, 1997; Pfeifer, 1998; Tsakarakis et al., 2007).

To be able to compare results, it would be important to have a common platform for research purposes. However, even if a few companies propose commercially available humanoid robots, most research groups build their owns. To illustrate the current situation, it is useful to briefly present humanoid robots around the world.

The first full scale humanoid robot was built in Japan in 1973. The Waseda university developed the WABOT-1 robot (Kato, 1973). WABOT-1 (contraction of WAseda and roBOT) had 6 Degrees of Freedom (DOFs) arms and 6 DOFs legs. Equipped with a vision system and a conversation system, it was then able to communicate in Japanese and to grasp and transport objects by using external sensors. Since then, WABOT-2 was developed (Sugano & Kato, 1987) followed by the robot Hadaly-2 and WABIAN (Hashimoto, 2002; Yamaguchi, Soga, Inoue, & Takanishi, 1999). WABOT-2 was a musician robot able to play music on a keyboard while Hadaly-2 was specifically designed for information interactions with humans and WABIAN was designed to have a human morphology. Currently Waseda is working on the new version WABIAN-2R (Ogura, 2006) and other robots like the Anthropomorphic flutist robot WF-4RIV (Solis, 2007) or the emotion expression Humanoid Robot WE-4RII (Kazuko, 2006).

Another major pole of the humanoid robotics is the Honda Motor CO Ltd.

In 1986, Honda Motor launched a humanoid robot research and development program with the goal of developing a robot with human like capacity. The keywords for this robot are “intelligence” and “mobility”. The idea was to develop a robot that could be used in an environment designed for human beings in order to be used by anyone at home or anywhere else. From 1986 to 1993, Honda developed the locomotion system and built the “E” robot series with “E” standing for the “Experimental Model”. Once the locomotion system was established, Honda continued the development with the “P” robots series (“P” stands for “Prototype Model”) from 1993 to 1997. The “P” series consist in three prototypes of complete humanoid robots. In 2000, Honda released the ASIMO model (Sakagami, Watanabe, Aoyama, Matsunaga, & Higaki, 2002). Since then, most of the development effort has been concentrated on the “artificial intelligence” of ASIMO. However, the hardware development is still continuing and three generations of ASIMO have been released since 2000. ASIMO has become worldwide known especially through shaking famous peoples hands in diverse manifestations.

More recently, other companies developed humanoid robots. After the success of the AIBO, Sony developed the QRIO (Fujita, Kuroki, Ishida, & Doi, 2003), a mini humanoid robot designed to interact with human beings as an entertainment robot. QRIO began its activities as Sony “Corporate Ambassador” in October 2003. QRIO was a promising robot, its last version was able to discriminate voices and faces and thus to recognize a person and associate to this person what he/she likes or dislikes. Although QRIO was close to be commercially available, Sony decided to stop its development (as well as the development of AIBO) in January 2006.

In 2001, Fujitsu Automation released the HOAP 1, a mini humanoid robot of 48cm and 6kg. HOAP stands for “Humanoid for Open Architecture Platform”. Since then, HOAP 2 and HOAP 3 has been developed. The last one, HOAP 3, is a humanoid robot of 60cm for 8.8kg. Unlike the ASIMO or QRIO, HOAP is an open architecture platform that can be purchased by particulars and laboratories for their experiments for approximatively 60’000 USD. It runs with real time Linux and can be controlled by an external Linux station or in an autonomous mode using the integrated CPU, a Pentium M 1.1 GHz.

Another popular full scale humanoid robot in Japan is the HRP-2 robot (HRP stands for “Humanoid Robotics Project”) developed by the Kawada Industries. HRP-2 was developed as a R&D robot for fields like Bipedal Walking and Human-Robot interaction in open space. With 58 Kg for 154 cm, HRP-2 has the size of a small human being. An interesting detail is that the external design of HRP-2 as been done by Mr. Yutaka Izubuchi, a mechanical animation designer famous for his robots that appear in Japanese anime.

Outside Japan, Sarcos is specialized among others in entertainment robotics. The company has developed the humanoid robot DB which is actuated by pneumatic actuators. Compared to electrical actuated robots, the pneumatic actuators provide a dynamic much more similar to the one of a human. The only drawback of the system is the external pneumatic pump used to provide the energy to the robot. The pneumatic system coupled with the powerful external pump allow very big accelerations for the control of the robot. Thus, DB was able to learn to play games like air hockey for example (Bentivegna, Ude, C.G., & Cheng, 2002). An other new mini-humanoid robot is developed by a French enterprise Aldebaran Robotics (Gouaillier & Blazevic, 2006). Nao is design to be a low cost entertainment humanoid robot similar to QRIO. NAO is expected to be available for the public at the end of 2009.

Today, it is still very difficult for Universities that work in humanoid robotics to find a humanoid robot adapted to their needs. The costs are too high, the choice is very reduced and most robots are not for sale but have to be rented. Then, depending on the use of the robot - and knowing that there is not a big choice among the available humanoid robots - it is sometimes easier for a Lab to design its own robots with its own specifications.

Among robots developed by Universities, the best known are the MIT humanoid robot COG (Brooks, Breazeal, Marjanovic, & Scassellati, 1999) developed in collaboration with the NASA for space application, ARMAR, a humanoid torso mounted on a wheel platform and developed at Karlsruhe university for Human-Robot interaction and manipulation tasks, and finally the iCub (Sandini, Metta, & Vernon, 2007), a humanoid robot developed by the European Consortium Robotcub. The goal of iCub is to provide a complete open source humanoid platform dedicated for research on behavioral development cognition.

The great majority of humanoid robots are developed following a classical mechanical design using a series of rotational joints linked rigidly and actuated by electrical DC motors. With such a system, the human kinematic can only be approximated. In order to have a better model of the human kinematics, non conventional designs are necessary.

A good example of such non conventional designs is the work done in the JSK Laboratory at Tokyo University which has a project on designing bio-inspired humanoid robots. Two robots where first designed, CLA and KENTA (Mizuuchi, Inaba, & Inoue, 2001; Mizuuchi et al., 2002), with the particularity of having a flexible spinal cord. The current impressive prototype of robot designed in the JSK Laboratory is the robot Kotaro (Mizuuchi et al., 2006). Kotaro is a small bio inspired humanoid robot of 133cm for 20kg with a Musculoskeletal structure (See Figure 1.1). By Musculoskeletal, we mean a skeleton that imitates in details the skeleton of a human even with respect to the complexity of the shoulder or of the hips. The skeleton is actuated by artificial muscles. The advantages of this system is its modularity since we can try different combinations (hang the tendons at different locations) and its compliance due to the elasticity of the artificial muscle. However, the main drawback of this system is its extreme complexity from the control point of view (about 90 DOFs).

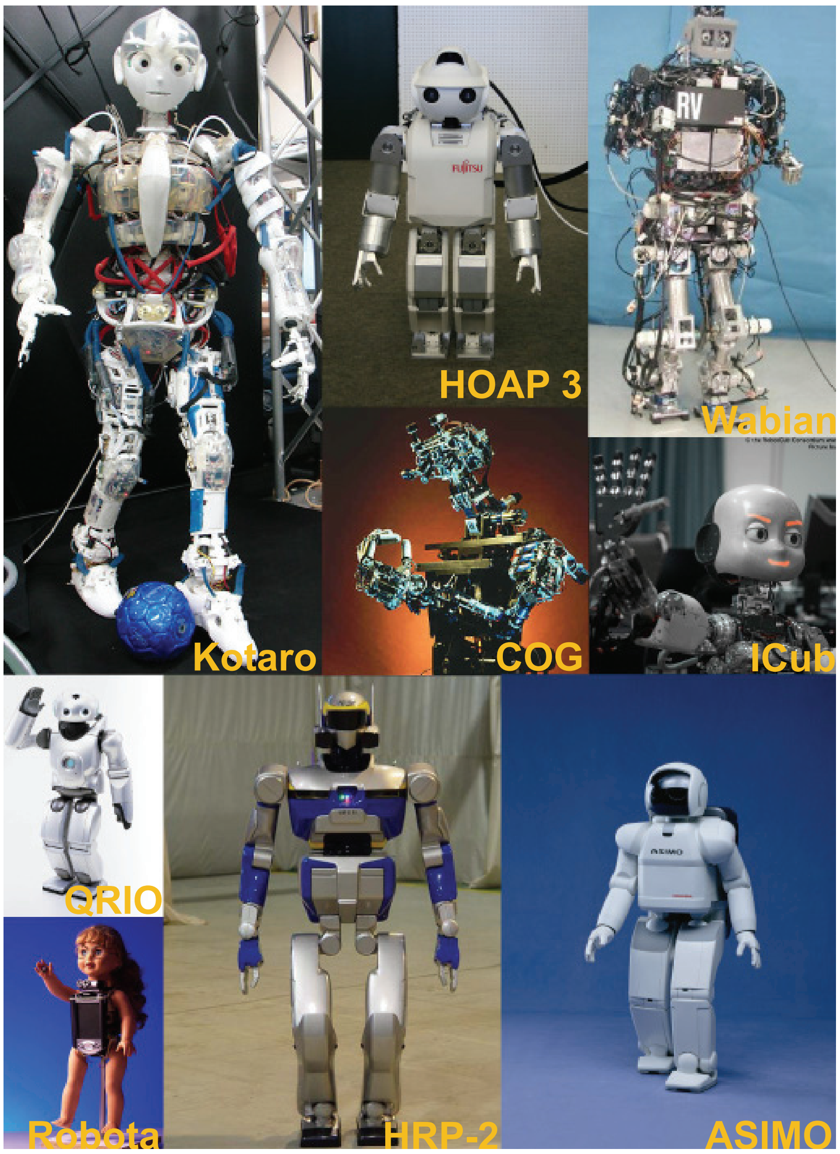

Figure 1.1: Different pictures of humanoid robots presented in Section 1.1.1.

At the Learning Algorithms and Systems Laboratory (LASA), we work mostly on humanoid robots. We have currently a Fujitsu HOAP 3 and as a partner in the European Project Robotcub, we are building a prototype of the ICub robot (Tsakarakis et al., 2007). But we are also developing our own mini humanoid robot, Robota. Robota is a small doll shape humanoid robot developed as an educational toy for children with disabilities. In its first version, Robota has 5 DOFs, one for each limb and one for the head. Robota is then able to imitate very simple movements like putting the hands or the legs up and down and turning the head. The limitations of the first version of Robota reduce greatly the possible interactions during a game. We thus felt the need to develop a new version of the robot with extended capabilities. If it had been possible, it would have been easier to buy a humanoid robot, however, the strict constraints of the Robota project, especially concerning the manipulation capabilities and the aesthetic, discard the few humanoid robots available on the market. As all robots on the market look like Japanese anime, none of them satisfies our aesthetic needs. For the Robota project, the aesthetic aspect is quite important for the interaction between robots and children. That is why the LASA developed its own mini humanoid robot. At the mechanical design level, the goal of the project is to come as close as possible to the kinematics of a human being to reduce the "correspondence problem" (see Section 1.2) to a minimum while keeping the system as simple as possible in order to have a cheap and reliable robot. These constraints have also an impact at the control level. By reducing the differences between human embodiment and the robot kinematic, and keeping the mechanical design as simple as possible, the control is made easier especially to program the robot by demonstration. More details on the design of Robota II are given in Chapter 2.

1.2 Programming by Demonstration (PbD)

Robots can be extremely useful tools for several tasks. Robots are powerful, never get bored, can be very precise and, as long as the energy is provided, never get tired. Today, robots are coming out of their industrial environments and start pervading human environments as guided tour robots, vacuum robots or lawner robots, and tomorrow they will probably do much more for us. The idea of a robot capable of learning to accomplish different tasks in order to help people in their every day life is already fixed in the mind of many researchers. However, there are a couple of challenges to realize this idea. One of the biggest challenges is to create a simple interface allowing interactions between robots and human users. Such an interface should be intuitive enough to allow somebody without any knowledge in computer science, electricity or mechanics to program new tasks on his/her robot.

A promising mechanism to convey new know-hows which is already widely used in the animal kingdom (Dautenhahn & Nehaniv, 2001) is to learn by imitation. “Imitation Learning” is probably the most natural way of teaching a new task. In robotics, this mechanism has also been called Programming by Demonstration (PbD) (Billard & Siegwart, 2004).

Robot programming by demonstration has become a key research topic in robotics. Works in that area tackle the development of robust algorithms for motor control, motor learning, gesture recognition and visuo-motor integration. While the field exists since more than twenty years, recent developments, taking inspiration in biological mechanisms of imitation, have brought a new perspective to the domain (Andry, Gaussier, Moga, Banquet, & Nadel, 2001; Schaal, Ijspeert, & Billard, 2003).

Recent advances PbD have identified a number of key issues to ensure a generic approach to the transfer of skills across various agents and situations. These have been formulated into a set of generic questions, namely *What-to-imitate*, *How-to-imitate*, *When-to-imitate* and *Who-to-imitate* (Nehaniv & Dautenhahn, 1999). These questions were formulated in response to the large body of work on PbD which emphasized ad-hoc solutions to sequence and decompose complex tasks into known sets of actions, performable by both the demonstrator and the imitator (see e.g. (Kaiser & Dillmann, 1996; Skubic & Volz, 2000; Yeasin & Chaudhuri, 2000)). In contrast to these other studies, the four questions above and their solutions aim at being generic by making little or no assumption concerning the type of skills to be transmitted.

In the framework of this thesis, to simplify the problem, we will consider that the learning phase is defined. Thus, the robot does not have to recognize by itself when it has to learn and who is the demonstrator. The beginning and the end of the learning phase are defined by the user and the person providing the demonstrations is known. Thus, the two main problems remain namely, the *What-to-imitate* problem - that refers to the problem of determining which of the features of the demonstration are relevant for the achievement of the task (Billard, Epars, Calinon, Cheng, & Schaal, 2004; Calinon, 2007) - and the *How-to-imitate* problem - also referred to as the *Correspondence problem* (Dautenhahn & Nehaniv, 2001; Alissandrakis, Nehaniv, & Dautenhahn, 2002), that is the problem of transferring an observed task into one’s own capabilities and to adapt it to different situations. Works tackling this issue followed either an approach in which the correspondence is unique and the imitation must produce an exact, but parameterizable, reproduction of the trajectories (Ijspeert, Nakanishi, & Schaal, 2002; Ude, Atkeson, & Riley, 2004; Demiris & Hayes, 2001), or an approach in which only a subset of predefined goals must be reproduced (e.g. (Dillmann, 2004; Zhang & Rössler, 2004; Skubic & Volz, 2000; Billard & Schaal, 2001)).

Concerning the *What-to-imitate* problem, one approach aims at extracting and encoding low-level features - e.g. primitives of motion in joint space (Ijspeert et al., 2002; Ude et al., 2004; Calinon & Billard, 2005) - and makes only weak assumptions as to the form of the primitives or kernels used to encode the motion. By contrast, another body of work stresses the need to introduce prior knowledge in the way information are encoded in order to achieve fast and reusable learning in the imitation of higher-level features - such as complete actions, tasks, and behaviors (Zoellner, Pardowitz, Knoop, & Dillmann, 2005; Zhang & Rössler, 2004; Steil, Roethling, Haschke, & Ritter, 2004).

In the Learning Algorithms and Systems Laboratory, the first problem has been explored by Sylvain Calinon (Calinon, 2007). In his work, he takes aspects from both approaches. Different demonstrations of the same task are performed and a probabilistically based estimation of relevance is used to extract the important aspects of the task. This method provides a continuous representation of the constraints given by a time-dependent covariance matrix that can be used to reconstruct a generalized trajectory.

Here, even if the two problems are highly related, the focus will be on the second problem, the *How-to-imitate* problem. It can be decomposed in a set of sub-problems. In the specific task of programming a robot by demonstration, once the desired informations are extracted from the demonstrations, the first problem encountered for the reproduction is the difference of embodiment between the demonstrator and the robot. As mentioned above, it is also known as the *Correspondence problem.* As discussed in Section 1.1.1 the fact that we work with humanoid robots is already a big part of the solution. The second problem is to be robust in different situations. Knowing the constraint learned during the demonstration, how can we adapt the movement to accomplish the learned task in a setup that has not been demonstrated?

1.2.1 Issues in PbD

In mass industry, where product chains are setup to mount millions of pieces per year, PbD is not a priority. Even if development and maintenance of robot programs is expensive in terms of money and time, the couple of days needed to program the robots of the assembly chain are not so important compared to the series of pieces produced. However, in Small and Medium Enterprise (SME), there might be big advantages in being able to program robots by demonstration. For small series for example, it is not so efficient to rent the services of a specialized technician during a couple of days to program the robots of the assembly chain. Giving the possibility to unexperienced workers to program robots can be very useful in order to setup rapidly a production chain for small series. The European project SME-Robot leaded by the Fraunhoffer-IPA in Germany aims at developing user friendly robots for Small and Medium enterprises in order to provide more adapted solutions for the automation of small series and thus increase the competitiveness of European SME in the world market.

Currently, simple algorithms can be used in industrial robotics. The teaching can be done by simply moving the robot. The data are then recorded and can be replayed. For simple pick and place tasks in a known environment, this system works quite well. It becomes more complicated if pieces that have to be moved are not always at the same place or in the same position, e.g. to take some pieces out of a box. In that case, it is more complicated in terms of teaching and reproduction. We would first need to have a tool to extract the position of the pieces. Then, during the demonstrations, the robot should be able to understand that it exists a direct relation between its hand and the object position (for simple pick and place tasks, it just reproduces the learned trajectory without taking any care of the object). In short, the robot needs much more than just recording the trajectory. It has to make a sort of generalization of the set of demonstrations and to be able to reproduce the task depending on the situation.

In industrial robotics, even if there are specific tasks that need more than a record and play algorithm to be solved, the environment stays relatively predictable. However, if we try to imagine a personal robot that evolves in a human house, the environment is much less predictable and the number of tasks to learn explodes. The robot has to learn new tasks for each new action and to adapt the learned task to a new context at every reproduction while taking care to avoid any dangerous action. Here, more than in the industrial world, error is not permitted. These type of robots would require incredibly adaptive and reliable algorithms.

1.3 Reinforcement Learning

PbD supposes that a teacher is always present to teach the robot a new task and to correct it when the reproduction is not satisfying. There are some other techniques that allow the robot to learn a task by itself without external help. Reinforcement Learning is one of these techniques.

In contrast to PbD which is a learning technique that use the examples given by a teacher, Reinforcement Learning (RL) is a learning technique that does not require the presence of a teacher. The principle of RL is to improve the capacity of a system by applying "trial and error" techniques. Usually, a classical RL algorithm can be described as follows: An agent evolves in an environment defined by a finite number of states. At each state, an action is chosen following a given policy, this action will bring the system to another state with a given probability. A Reward Function is then used to evaluate the action. The goal of the RL algorithm is to maximize the total reward over the task by choosing the appropriate policy. Different tools can be used for that. The value function is used to estimate the reward that can be expected for a given state. The action value function is used to estimate the reward that can be expected for a given state with a given action.

Three main approach can be distinguished in order to find the optimal policy:

a) *Dynamic programming:* The dynamic programming approach considers that a perfect model of the environment as a finite Markov Decision Process

(MDP) is given. The basic idea of this approach can be describe in two step. The first step is known as the *policy evaluation* step and refers to the iterative computation of the value function for each state given the current policy. The second step is known as the *policy improvement* step and refers to the computation of a new improved policy knowing the value function and the current policy. These two step are performed until an optimal policy is found.

b) *Monte Carlo Methods:* The Monte Carlo approach does not assume a perfect model of the world, but is defined only for episodic tasks. Basically, the idea is the same as before, we have a *policy evaluation* step and a *policy improvement* step. The main difference lies in the evaluation of the policy. Here, the idea is to evaluate the policy from experiments. Thus, for each state, we average the rewards obtained after visiting the current state. Thus, after a couple of episodes the average tends to the value function. Once the value function is estimated, the *policy improvement* step is performed using the value function.

b) *Temporal-difference learning:* The temporal difference learning exploits the advantages of the two previous methods. The algorithm does not need a perfect model of the environment and is able to update the value function of each state without waiting for the end of the task. For the update of the value function of the current state, the value function of the next state and the observed reward are used, thus we do not need to wait for the end of the task and we need only two state of the complete environment.

There are several possibilities to combine these different techniques, however, the RL algorithms that we encounter today in the robotic fields are mostly derived from the temporal-difference learning method. Among the successful implementation, we have applications such as swinging up and controlling an inverse pendulum (C. Atkeson & Schaal, 1997) or refining forehand and backhand tennis swings (Schaal, Peters, Nakanishi, & Ijspeert, 2004). In these examples, the dynamic of the robot and the environment were modelized and RL algorithm was used in order to optimize the critical coefficient of these models. Following the same idea, an interesting work was done on acquiring a dynamic stand-up behavior (Morimoto & Doya, 2001; Iida, Kanoh, Kato, & Itoh, 2004) on a 2 DOFs robots with 3 links.

As shown in these examples, RL algorithms perform well in discrete environments of small dimensionality. However, to use RL on real robots in real environment, the algorithms have to be adapted to tackle data of high dimensionality lying in continuous time and space. Bradtke & Duff (Bratke & Duff, 1994) derived algorithms for continuous time semi-Markov Decision Problems. Doya (Doya, 2000) proposed methods such as Q-Learning or Actor-Critic (R. S. Sutton & Barto, 1998; Konda & Tsitsiklis, 2000) for continuous time and space environment.

To deal with the high computing costs of the RL algorithms for higher dimensionality systems, different solutions, based on function approximation and gradient descent algorithms, have been proposed (Nedic & Bertsekas, 2002; Williams, 1992; R. Sutton, McAllester, Singh, & Mansour, 2000; Barto & Mahadevan, 2003; Amari, 2000). Following this trend, Peters et al. (Peters, Vijayakumar, & Schaal, 2003, 2005) proposed the Natural Actor-Critic (NAC) algorithm. This algorithm is based on the Natural Gradient which is more efficient than the more classical solution of the Vanilla Gradient in terms of speed of convergence. RL algorithms can become very complex when applied on real problems in continuous time and space. There are a lot of parameters to fine tune and studies have been conducted on the influence of different parameters on the learning like the reward function (Konidaris & Barto, 2006), the effect of using inaccurate models (Abbeel & Quigley, 2006) or on new efficient exploration process (Simsek & Barto, 2006).

1.4 My contribution

The hardware part of this thesis is focused on the design of a new mini humanoid robot Robota II with extended capabilities. Robota is an innovative robot because of its small size with respect to its big number of DOFs. My contribution to the development of Robota II was the design of the electronic control boards and the supervision of the mechanical design.

The software part of thesis is focused on the *How-to-imitate* problem when programming a robot by demonstration. This problem can be divided in two parts. The first is related to the difference of embodiment between the human demonstrator and the robot imitator. The second is the problem to generate a trajectory in an unknown setup while respecting the constraint extracted during the demonstrations.

This work was complementary to that of Sylvain Calinon (Calinon, 2007). In the first part of the problem, the contribution of my work is the development of a generalized inverse kinematics that allows us to compare the data learned in the joint angle space and in the Cartesian space. This algorithm can be seen as a generalization of the Damped Least Square Method applied on the Jacobian. This type of algorithm as already been introduced by Nakamura (Nakamura & Hanafusa, 1986) and Wampler (Wampler, 1986) as an alternative to the Moore-Penrose pseudo-inverse with the big advantage of being robust to singularity. Under the hypothesis that we work with a humanoid robot, the novelty here is to use this type of algorithm to solve the *Correspondence Problem* by combining the redundant data of the joint angle space and of the Cartesian space from the demonstrator in order to find an optimal solution for the imitator.

For the second problem, the contribution of my work was to add more robustness to the algorithm developed in collaboration with two LASA colleagues, Sylvain Calinon and Micha Hersch (Hersch, Guenter, Calinon, & Billard, 2008). A Reinforcement Learning module is used to relearn the model that the robot has for a given task. The algorithms by themselves are not new, but the combination of the different algorithms allows us to speed up the learning of a new task. Since PbD provided a good starting basis to search the parameter space, by applying RL on a model that was first learned by demonstration, we optimized the time it took to learn a good solution to a given problem compared to learning it when using solely RL. Thus by combining PbD and RL, we obtain a much more robust algorithm. In addition to that, the Reinforcement Learning module can be used only to optimize the reproduction of the task and thus optimize the robots model for its own embodiment. The RL module is useful to tackle big perturbations like new obstacles, but the problem is that we have no explicit reference to those eventual obstacles in the formulation. It means that we are able to learn a new trajectory for a given position of such an obstacle, but the learning must be performed again if the location changes.

The last contribution of this thesis is to adapt the trajectory generation in a new system inspired by the potential fields method. In this system, we have introduced an explicit reference to potential external objects that can play the role of obstacles trough a virtual repulsive force exerted by the object on the end-effector. By using Reinforcement Learning, we should then be able to learn a model of the influence of each object on the trajectory. Because of this explicit reference to external objects, we do not need to relearn the model of the task if the location of the objects changes and we are able to generate a trajectory dynamically.

1.5 Outline

The organization of the thesis follows in chronological order the different parts of the work accomplished during my PhD.

Chapter 2 will present the hardware part of the thesis. The development of the Robota project will be presented and discussed. A short presentation of the HOAP robot will also be done. The HOAP is used in most experiments to validate the learning algorithms.

Chapter 3 will present the different solutions used to resolve the inverse kinematics and control the reproduction of movement performed by the robot. The problem of how we can use this inverse kinematics in order to give a solution to the *How-to-Imitate* problem is approached in this Chapter.

Chapter 4 will present briefly an algorithm developed in collaboration with Sylvain Calinon and Micha Hersch that consists in combining different techniques such as Dynamical Systems and Gaussian Mixture Model in order to build a robust controller. The main part of the Chapter will be dedicated to the description of the Reinforcement Learning module used to optimize the robot's model of the task in this framework.

Chapter 5 will present the evolution of the algorithm presented in Chapter 4 in order to have a more consistent algorithm in which we will be able to relearn the reaction the robot must have in presence of new objects on the setup, like an obstacle for example.

Chapter 6 will discuss specific problems in the domain of humanoid robotics, control and PBS.

Finally, Chapter 7 will present the conclusion of my thesis.

2 Robot Platforms

2.1 Outline

In order to validate the algorithms and programs developed during the project, it is important to work with real robots. In this chapter, two robotic platforms will be presented: Robota II, which is the prototype of the new model of the robot Robota, and HOAP 2 and 3, which are commercial humanoid robots developed and sold by Fujitsu Ltd.

Section 2.2 will present the Robota Project. Robota is a small humanoid robot with 5 DOFs. The capability of Robota was very limited and developing a new robot with extended capabilities became essential in order to use Robota to imitate more complex tasks. That led to the development of Robota II. My work on Robota II was mainly the supervision of the mechanical design done by students in semester or master project.

Section 2.3 will present the HOAP robots used at the LASA. The HOAP robots series has been designed by Fujitsu Ltd. HOAP stands for "Humanoids Open Architecture Platform". HOAP 2 and then HOAP 3 have been used to implement and validate our control algorithm on a real robot.

2.2 Robota Series

The Robota project consists in a series of multiple degrees of freedom doll-shaped humanoid robots, whose physical features resemble those of a baby. The Robota project is part of a current trend in robotics research that develops educational and interactive toys for children with disabilities; see, e.g. (Plaisant et al., 2000). The Robota project started in 1997 with the first prototype. Numerous iterations thereafter followed, leading to the creation of a commercial prototype, sold by DIDEL SA.

2.2.1 Robota

The original Robota robot, sold by DIDEL SA, has 5 degrees of freedom (DOF): 1 DOF for each leg, arm and head. DC motors with a 1:6 gearings drive the 5 DOFs, providing a maximum continuous torque of about 90mNm. Robota's electronic components consist of a Motor Board and a Sensor Board. The Motor board is addressed via a RS232 serial interface from a PC or a PocketPC.

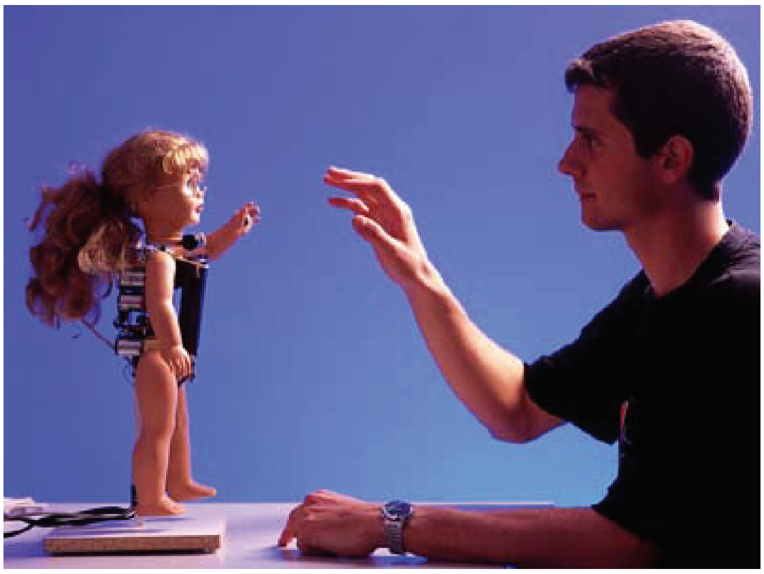

Figure 2.1: A Pocket PC mounted with a FlyCam Camera tracks the motion of the user and sends commands to the micro controllers and servos of Robota, so that it can imitate the motion of the head and arms of a user.

The Sensor Board is addressed from the Motor Board via a Serial Peripheral Interface (SPI). Motor Board and Sensor Board are controlled by two PIC16FA micro-controllers. Schematics of the electronic boards and of the mechanics are available in the Robota User Guide at $http://robota.epfl.ch$.

Robota's behaviors allow multi-modal human-robot interactions. Equipped with vision and speech processing, Robota is able to imitate very simple gestures like moving up and down its arms or its legs and moving left and right its head, see Figure 2.1. A small camera detects the positions of the user's hands and head in the screen and Robota moves its limbs depending on this position. Robota is also able to dialog via speech processing/synthesis, and to learn simple composition of motions and of words (see (Billard, 2003, 1999)).

The Robota robots find an application in two areas:

1) In education, as a programming tool at the University, and as an entertaining interactive toy in museums (Billard, Epars, Cheng, & Schaal, 2003); and 2) In behavioral studies with autistic children, that investigate the potential of an imitator robot to test the ability of autistic children to imitate and to teach these children some simple coordination behaviors (Robins, Dautenhahn, Boekhorst, & Billard, 2004).

2.2.2 Robota II

The first commercial version of Robota is very limited, at least in terms of imitation and corporal interactions. Developing a new version with extended capabilities became essential in order to realize more complex imitation games as e.g. some manipulation tasks. Robota II was built following the same constraints as Robota, i.e. to keep the idea of a doll shape humanoid robot, to have back-drivable DOFs for kinesthetic teaching, to have limited power for security reasons and finally, to keep the size as small as possible.

There are numerous advantages for having small robots, small is cute, small is safer and small is cheaper. The latter is an important aspect of the Robota project, as it opens the doors for global commercialization. However, there are also a number of drawbacks. Miniature components (motors, gears, etc) are not as efficient as larger ones and are not always readily available. Moreover, miniature controllers remain limited in their processing power. As small implies also light, the choice of batteries, as well as the number of sensor components is limited.

The mini-humanoid robots that we find today are optimized more for locomotion than for manipulation (see for example, the Honda ASIMO (Sakagami et al., 2002), the Sony QRIO (Fujita et al., 2003) and the Fujitsu HOAP). But the limited capacity for manipulation is not the only drawback of those platforms. The acquisition of such a robot is quite difficult as QRIO is not commercialized, ASIMO can be rented for approximatively 170'000 USD per year, but is still to big for the Robota project and HOAP 3 is sold for approximatively 60'000 USD.

In order to benefit from the additional capabilities of the prototype of Robota II, more complex algorithms have to be implemented for the control of the robot. The main use of the new Robota will be human interaction and imitation. By constraining the design of the robot to a human like arrangement of the DOFs, it is possible to simplify a lot the complexity of the imitation algorithms. In fact, the design of the robot has a big influence on the resolution of the imitation problem known as the "Correspondence Problem" (Nehaniv & Dautenhahn, 1999).

The new prototype of Robota consists in two 7 DOFs arms, a 3 DOFs torso, a 3 DOFs neck and 3 DOFs eyes. A first prototype has been realized by students in semester project or master project for the right arm, the spine and the neck.

Figure 2.3 shows our current prototype of a 6 DOFs arm with a 1 DOF gripper. The arm is 26 centimeter long for 700gr and a payload of 200 gr. (Guenter, Guignard, Piccardi, Calzascia, & Billard, 2004). Each DOF is back-drivable. In order to obtain human like movements, the different DOFs were placed in the following order[1]: shoulder flexion-extension, shoulder abduction-adduction, shoulder humeral rotation, elbow flexion-extension, wrist rotation, wrist flexion-extension and gripper (see Fig. 2.3). Mechanical stops ensure that each DOF is bounded within the limits of the corresponding DOF of the human arm. Because of cost and space constraints, we had to rule out solutions closer to the human control system (e.g. hydraulic motors, linear actuators) and use

[1]We are aware of the fact that the human arm is not controlled by a serial muscular system. Nevertheless, the order in which the rotation axes have been placed gives the closest equivalent to the human arm motion.

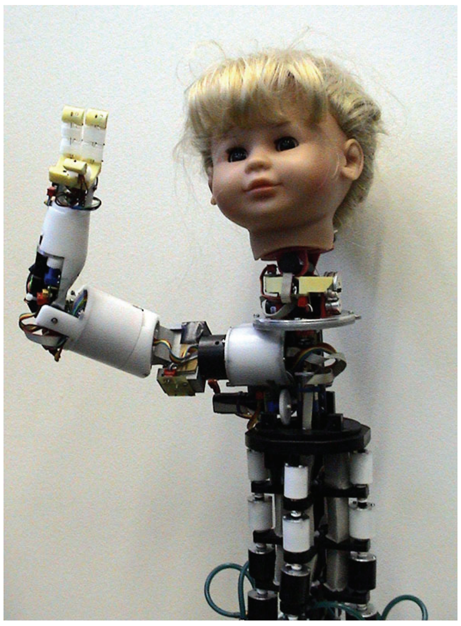

Figure 2.2: The first prototypes of the neck, spine and right arm for Robota II.

electric rotative motors with a serial placement of the joints. The first arm prototype works well, however, there is still some little problems to solve in order to increase the lifetime of the model and reduce the backlash in certain DOFs.

Concerning the design of the neck, the number of DOFs was fixed to 3 (lace, pitch and roll). These were placed in series, starting with lace, and, then pitch and roll. The system has been designed to support a load of 400gr, so that it could still drive the head in the worst cases. Pitch and Roll are controlled by transmission through a set of cables and pulleys, while Lace is controlled by a direct drive from the motor. Using direct drive and cable transmission minimizes the chance of encountering backlash problems. However, some weaknesses in the cable transmission system appeared, the rotation axis were a bit under estimated and the clamping system is not appropriate. So it is difficult to tune the tension of the cables and the system untightens itself with time.

Concerning the torso, the motivations for designing a spine for our robot were two-fold: first, it would provide the robot with a smoother (human-like) bending of its torso; second, it would be a unique realization of this sort. While a few other prototypes of spines for humanoids robots exist, see (Mizuuchi et al., 2001, 2002), none of these were as small as the one we intended to develop. Ensuring that all components remain small (to fit within the doll's body) while supporting an important load (in proportion to the overall size of each limb) is and has always been a tremendous challenge in all the realizations we have described so far. It was even more so for the creation of the spine, and we had to develop home-made hydraulic actuators.

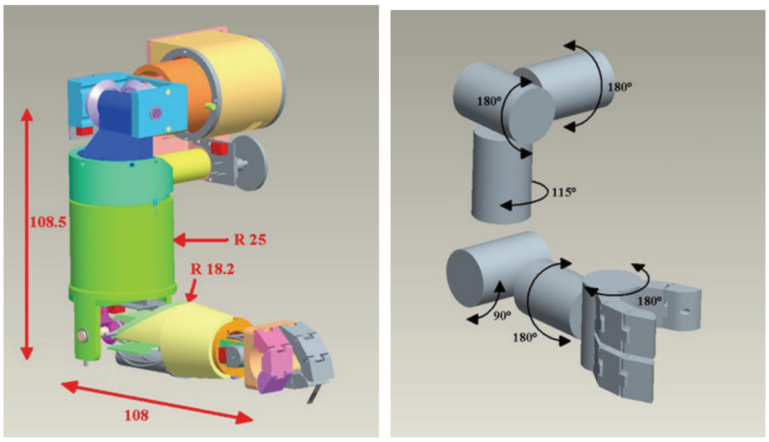

Figure 2.3: Left: Dimensions of Robota II 7 DOFs arm (1 DOF in the gripper); Right: Kinematic chain.

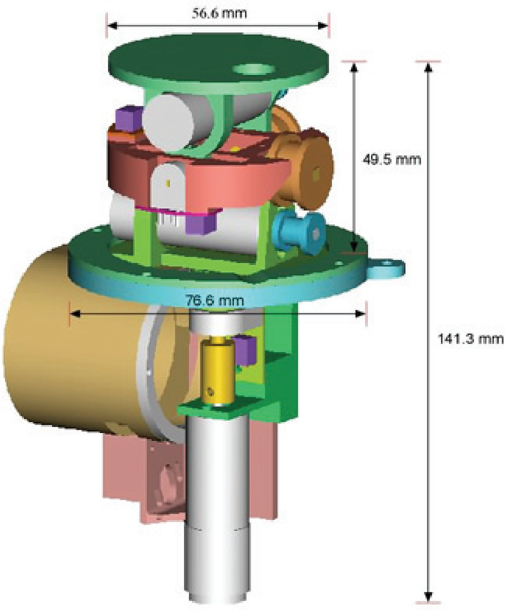

Figure 2.4: Dimensions of Robota II neck.

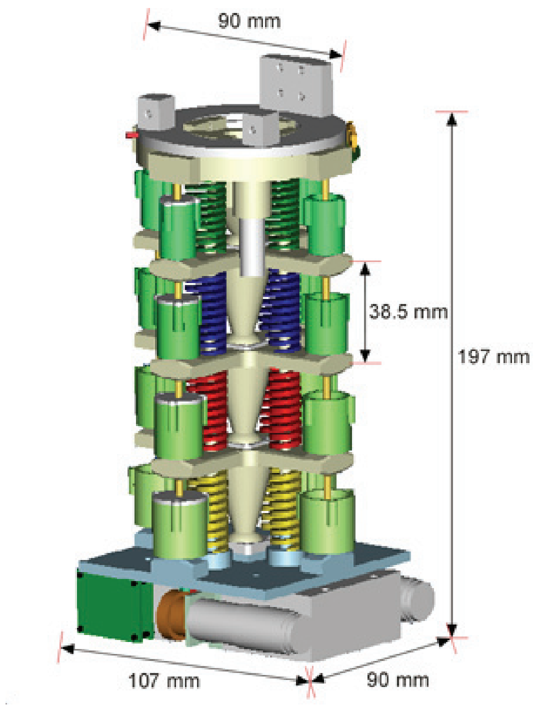

Figure 2.5: Dimensions of Robota II torso.

The current prototype of spine drives two DOFs for front-back and left-right bending respectively. The third DOF of the torso, supported by the spine, drives the horizontal rotation of the shoulders. The spine is about 200mm high for a diameter of 90mm and weights about 1Kg. The principal weakness of this spine is the hydraulic system. The prototype was precisely build to test the feasibility of the system. In this case, the use of a hydraulic system is not appropriate. Due to the miniaturization, problems like liquid loss or problems due to the viscosity of the liquid in the thin hydraulic tube appeared. Because this prototype is not usable as it is, it has to be redesigned by using another movement transmission system. The most suitable system seems now to be a cable driven system.

Currently, the prototypes of neck, spine and arms are redesigned in order to solve the problems found in the first versions. Another new prototype of head containing the 3 DOFs eyes has been designed by an EPFL engineer 2.6. The prototype has three DOFs: one DOF for the vertical movement of the 2 eyes and two separate DOFs to control the horizontal movement. The arrangement of the DOFs has been conceived in order to facilitate the implementation of stereo-vision algorithms in the eyes. Two CCD cameras have been mounted inside the eyes to perform 3D tracking of object, face or hands of the user. Nothing has been done for the design of Robota II legs, as the aim of the lab is to use the upper body only for accomplishing manipulation tasks. Robota II will stay fixed on a chair or on the floor and perform tasks that need the torso only.

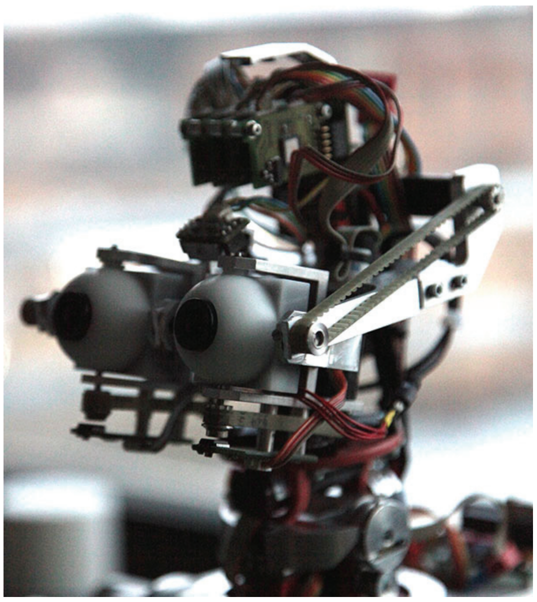

Figure 2.6: Prototype of Eyes for Robota II.

The motivation behind the construction of Robota stems from the need to have a robot with realistic human features, and, enough degrees of freedom to perform objects manipulation. Current commercially available mini humanoid robots have too few degrees of freedom (especially in the wrist) to perform complex manipulation tasks. Moreover, the majority of these robots have only one DOF in the torso, two DOFs in the neck, five DOFs for each arm and none for the eyes. In Robota, we have more DOFs for each of these parts. The drawback of having so many DOFs is that it requires important torques to drive all of the components. Unfortunately, because of the volume limitation, the fact that we have to rely on commercially available DC motors and in order to produce the required torque while using sufficiently small motors, we end up using motors that produce a very low speed for each DOF (much lower than that produced by Q-RIO and HOAP 2). While this is perfectly suitable for the application of the robot with children (in fact, much safer), it may not be optimal for other applications.

2.3 HOAP Series

Robota II is still in development and cannot be used to test the algorithm developed in this project. In order to have a suitable robot to implement and test control algorithms, two robots from the HOAP series from Fujitsu Ltd. have been acquired by the lab. HOAP stands for Humanoids for Open Architecture Platform.

The HOAP 2 has 25 DOFs, 6 DOFs in each leg, 1 DOF in the torso, 5 DOFs in each arm and 2 DOFs for the head. It has also a 3 dimensional accelerometer

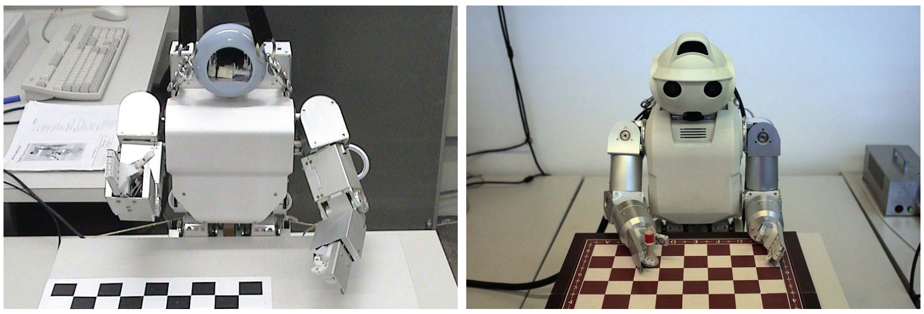
Figure 2.7: Left: HOAP 2 robot (25 DOFs, 50cm and 7kg); Right: HOAP 3 robot (28 DOFs, 60cm and 8kg)

and gyroscope in the torso and four pressure sensors in each feet. We renounced on buying a version with 2 camera in the head, as it was too expensive for the quality of the cameras proposed.

The HOAP 3 has 28 DOFs, 6 DOFs in each leg, 1 DOF in the torso, 6 DOFs in each arm, and 3 DOFs in the head. The additional DOFs are a rotation of the wrist in the arms and a roll angle in the head. In terms of sensors, the HOAP 3 has the same configuration as HOAP 2 but includes some additional sensors: two cameras in the eyes, a distance sensor in the head an two tactile sensors in the hands of the robots. It is then possible to implement stereo vision in the HOAP 3 head in order to track the objects on the scene. However, as the view angle is a bit too narrow and since it is not possible to measure the current position of the head due to the missing encoders on these DOFs, it is not easy to use the stereo vision in real experiments. It is thus very difficult to have a good estimation of the position of the object when moving the head. In most of the experiments, to avoid this problem, we use an external stereo-vision system.

As we are working on manipulation tasks and not on locomotion, we only use the 16 DOFs of the upper body of the HOAP 3 (13 DOFs for the HOAP 2) during the experiments. One can use the robot autonomously with a battery and an on board PC, but the autonomy is limited in terms of energy and computation power. As we do not need to move the robot, the cable for communication and power supply is not a problem, thus, we have no energy problems. The robot is then controlled via a PC running real time Linux. There is no on board computation, the control PC sends orders to each joint angle every millisecond through a memory shared between the main PC and the robot containing the desired trajectory.

We have the possibility to work off-line by writing the whole trajectory in the shared memory and then to run the trajectory on the robot. The off-line mode allows us to easily check the trajectory (smoothness, joint limit and speed profile) before sending it to the robot. If we want to have a feedback during the execution of the trajectory, we can work online and write the trajectory step by step in the shared memory. In that case, we have to be careful to provide new positions to the robot every millisecond. It is then more difficult to check all the parameters of the trajectory before sending it to the robot.

At a practical level, the robot HOAP 3 (like the HOAP 2) is not optimized to accomplish manipulation tasks. The first problem is the limited number of DOFs in the arm. Even if the HOAP 3 has one more DOF (the wrist rotation) than the HOAP 2, it is still very difficult to grasp some objects. The second problem is the weakness of the hand. The arm of the HOAP 3 is very powerful, the shoulder and the elbow are designed to resist to torques up to 1.5Nm (3Nm in peak). However, the wrist rotation and the motor used to close the hand are quite weak in comparison, so the HOAP 3 is not able to grasp objects heavier than 100gr. The last problem is a conception problem. The robot's workspace is very small. In addition to that it is very difficult to accomplish bi-manual operations because the torso of the robot is relatively big and hampers the movement when it has to manipulate something in front of itself.

2.4 Summary

This Chapter presented the hardware part of this thesis. Firstly the developments of prototypes for the upper body of Robota II were presented. The motivation behind the construction of Robota II stems from the need to have a robot with realistic human features, and, enough degrees of freedom to perform efficient object manipulation. Current commercially available mini humanoid robots have too few degrees of freedom (especially in the wrist) to perform complex manipulation tasks. Moreover, the majority of these robots does not reach the desired specifications in terms of aesthetic appearance. During my thesis, my contribution to this work was mainly the supervision of students in Semester and Master project for the mechanical design of Robota II. I also participated to the design of the PCB (printed circuit board) used for the low level motor control.

Secondly, the HOAP robots are presented. To test the algorithm on a usable robot during the development of Robota II, a HOAP 2 robot and then a HOAP 3 robot have been bought by the lab. The first advantage of the HOAP robots is their availability. In fact, there are not many mini-humanoid robots that are commercially available. The SONY Q-RIO was the other possible robot, but it was only available for renting. Another big advantage of the HOAP series is that it is an open source platform. Now, the drawback of these robots is mainly the small workspace that can be used for manipulation. In fact, those robots are designed more for walking than for handling manipulation tasks. On the opposite, Robota II is designed to be much more efficient for manipulation tasks but will not have legs.

3 How to Imitate & Generalized IK

3.1 Outline

In this Chapter, we present the work done in relation with the inverse kinematics and the "How to Imitate" problem in a PbD frame work. This work is complementary to the work of my colleague Sylvain Calinon (Calinon, 2007).

Section 3.2 introduces the problem of the Inverse Kinematics and gives an overview of the different techniques that can be used to solve it.

Section 3.3 presents a geometrical solution developed specifically for the robots used at the LASA (Robota and the two HOAP robots). In contrast to many other techniques that solve the IK problem by finding local solutions for small displacements, this technique allows us to find a solution for any given Cartesian position.

In Section 3.4, we present a solution to solve the IK in the specific case of PbD using humanoid robots. In this specific framework, we based our solution on the idea that the constraints necessary to fulfill a task lie in different referentials like for example the Cartesian referential of the robot's end-effector or the referential of the robot's joint angles. The Inverse Kinematics allow us to compare these two referentials and the redundancies of the data in the two referentials allow us to optimize the IK solution for a given task.

In Section 3.4.1, the main concept is presented. Section 3.4.2 presents briefly the work of Sylvain Calinon that is used here to encode the demonstration data. In Section 3.4.3 a cost function is proposed in order to evaluate the reproduction of a task by the robot. A first algorithm is presented to optimize the reproduction with respect to this cost function. The optimization is achieved by explicitly balancing the importance of the data in the combination of the two frames of reference. Another algorithm based on the same concepts is presented in Section 3.4.4. A new cost function is proposed and by using Lagrange optimization, a solution to the IK problem is derived. This solution can be seen as a generalization of the Damped Least Square Method applied on the Jacobian. It has the big advantage of being robust to singularities that can be encountered in the robot workspace.

3.2 Introduction

The control of humanoid robots operating in human environments presents an enormous challenge for roboticists. A major part of this challenge is related to the fact that, unlike traditional industrial robots, domestic robots must operate in dynamic and uncontrolled environments and interact in security with human beings.

There are different ways to control a robot: position, speed and torque control for example. But all these features have in common that at the lowest level, we have to give an input to the different actuators of the robot. As the robot has to interact in a 3D world where the position of objects are defined by 6 coordinates (usually described as a Cartesian space with three positions and three orientations), in order to find the command to send to the motors, we have to compute a kinematic model that contains the relation between the joint angle space of the robot and the Cartesian space of the world. With serial robots, finding the position of the end-effector used to interact with the external world is quite easy knowing the joint angles of the robot. For a given value of the joint angles, there is only one solution to describe the end-effector position and orientation. However, we usually need to find a solution in the joint angles space in order to reach a certain position and orientation with the end-effector in the Cartesian space. This problem is called the *inverse kinematics* problem. If we define the coordinates of a manipulator as the n-dimensional vector of joint angles $\boldsymbol{\theta}$ and the position of the m-dimensional vector $\boldsymbol{x}$, the forward kinematic function is given by: $\boldsymbol{x} = f(\boldsymbol{\theta})$, while the inverse kinematics is given by $\boldsymbol{\theta} = f^{-1}(\boldsymbol{x})$, where f is a continuous function $\in \mathbb{R}$.

When the problem is under constrained, i.e. when the number of degrees of freedom n exceeds the number of given variables m (for example, when controlling a 4 degrees of freedom robot arm given the 3D position of the target without the orientation), Equation 3.2 may have multiple solutions [1]. In this case, the robot arm is redundant.

When the robot's arm is redundant, for a given point, there will be a infinity of solutions. This gives us the possibility to choose the one optimizing a given criterion (minimization of the energy, minimization of the movement, posture optimization). A common way to solve the inverse kinematics is to use a local method by using the local linearization of the Jacobian $\boldsymbol{J}$:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{3.1}$$

And thus, what we have to do now is to inverse the equation:

[1]at the opposite, if we consider a 4 DOFs robots and a the position and orientation of the object, the problem is over constraint and may have no solution.

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}^{-1}\dot{\boldsymbol{x}} \tag{3.2}$$

Because the joint angle space and the Cartesian space have not the same size, the Jacobian is not a square matrix and thus cannot be inverted. A common solution is to take the Moore-Penrose pseudo-inverse $\boldsymbol{J}^+$, which is defined as follows.

$$\boldsymbol{J}^+ = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1} \tag{3.3}$$

This solution minimizes the displacement $\dot{\boldsymbol{\theta}}$. A more general solution was proposed by Liegois (Liegeois, 1977) which has the advantage of optimizing a explicit function g in the null space of $\boldsymbol{J}$:

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}^+\dot{\boldsymbol{x}} + \alpha(\boldsymbol{I} - \boldsymbol{J}^+\boldsymbol{J})\frac{\partial g}{\partial \boldsymbol{\theta}} \tag{3.4}$$

Where g is a convex function having a single optimum. The second term is thus, the optimization term which projects the function g in the null-space of $\boldsymbol{J}$. The null-space of $\boldsymbol{J}$ is the space of all vectors $\dot{\boldsymbol{\theta}}$ which give the null vector in the Cartesian space. In other words, they do not have any influence on the end-effector. In the case of a typical 4 DOFs arm like the arm of the HOAP 3, the optimization term influences only the position of the elbow. The advantage of this solution is that it is efficient enough to be run in real time for a simple arm as the arm of the HOAP 3. However, a big drawback is the problem encountered by the system around singularities. The term $\dot{\boldsymbol{\theta}}$ can explode for a very small displacement $\dot{\boldsymbol{x}}$, what leads to an uncontrolled behaviour of the robot.

Different ways of solving the inverse kinematics have been explored. One interesting idea is to learn the inverse kinematics model. The pseudo-inverse solution is based on an explicit model of the robot and depends on the precision of the model and on the calibration of the robot arm sensors. Learning the inverse kinematic seems to be a promising solution to solve the calibration problem and to allow the robot to adapt the model in case of joint failure.

Early experiments were conducted on 2 DOFs arms in a bounded space by Kieffer et al. (Kieffer, Morellas, & Donath, 1991). They conducted a series of experiments using a self-organizing mapping neural network. Despite of a big number of learning steps, they obtained very good results in term of precision. They also observed a relation between the number of neurons in the network and the resolution of the system as well as a relation between the accuracy and the number of learning steps. However a 2 DOFs robot arm remains a simple problem.

Other interesting experiments where conducted by Driscoll (Driscoll, 2000) in order to compare different configuration of a Radial Basis Function Neural Network for a more complex robot. He compared three configurations of a RBF neural networks for a six DOFs robot.

The training and tests were made in a first time within a bounded volume and in a second time outside of the volume. The three configurations gave similar results: good precision in the given volume and increasing errors when it goes away from the volume. The same type of tests have been made with multi-layer perception neural networks (Karlik & S.Aydin, 2000). In their paper, they compare two configurations to controll a robotic arm of three DOFs for placement and three DOFs for orientation. The first configuration was one neural network for the whole input and the second configuration was six different networks for the six inputs. When they tried the first configuration all the outputs converged in the same way. But when they tried the second, the first three DOFs converged quicker than the orientation DOF.

All the tests gave good results in terms of precision within the volume, but not outside of the predefined volume. The problem is that to approximate a highly non-linear function with linear systems, the found solution is a local linearization. Oyama (Oyama, Agah, MacDorman, Maeda, & Tachi, 2001) proposed a new architecture for using all the work space of the robot. This architecture was composed of several parallel networks called expert networks and a selection network which was linked with all expert networks. The principle was simple: all experts give an answer to a task from which the selection network takes the best. In other words, all experts approximates local models and the selection network takes the best model for the current location.

A promising method uses memory based neural network and Locally Weighted Regression (C. G. Atkeson & Schaal, 1995) algorithm (LWR). This algorithm is a statistical method to learn inverse kinematics. The principle is to reconstruct a non linear sampled function by fitting a local model on the nearby known points. Thus, we can train the robot by exploring randomly the space while the neural network registers some training points at regular times. It can be compared to a look up table used for finding any point by locally weighted regression. The robot learns quickly to control its movements. In ten minutes, it is able to accomplish a task like drawing an eight (D'Souza, Vijayakumar, & Schaal, 2001). For a more precise execution of the task, it needs three more minutes to learn it. After that, the robot is as precise as a robot using a numerical algorithm with the advantage of the neural network adaptability.

In terms of robot control, the work done by Oussama Khatib et al. gives impressive results (Sentis & Khatib, 2006; Park & Khatib, 2008), a framework for a full humanoid robot control has been developed. In classical inverse kinematics algorithm like the pseudo-inverse algorithm with optimization, the main task is, for example, to accomplish a given movement or to reach a given point while trying to fulfil some additional constraint given by the function g. The problem in defining an algorithm is to define the priorities. In the pseudo-inverse with optimization, the constraints are of second priority. They are projected in the null space of the task space and there is no guaranty of fulfilling these constraints. Now, if we take the example of a full humanoid robot that has to reach a point with one of its end-effector while keeping the equilibrium as a constraint, the constraints are more important than the task, because if the robot falls down, it will not be able to fulfil the task. In the work of Sentis & Khatib (Sentis & Khatib, 2006), the priority is given to the constraints and the task is projected in the null space of the constraint. Thus, in the example above, we can ensure that the robot keeps the equilibrium. In this work a third space is added: the posture space. The specification of the body position is described in this space and projected in the null space of the task space. This space helps the system to optimize the posture of the robot. By integrating the dynamic of the movement in the system, a force control can be applied. As before, we can add to the constraint space a defined contact force that is applied on a object (Park & Khatib, 2008) for example. That can be very useful for object manipulation.

In this thesis, we will use the redundant information in different reference frames extracted from demonstrations provided by a human teacher in order to give an optimal solution for the reproduction of the movement. The different reference frames consist mainly of two categories, the joint angle frame and the Cartesian coordinate frame. Combining these data leads to a solution to the inverse kinematics that can be seen as a generalization of the Damped Least Square Method applied on the Jacobian (see Section 3.4.4). This type of algorithm has already been introduced by Nakamura (Nakamura & Hanafusa, 1986) and Wampler (Wampler, 1986) as an alternative to the Moore-Penrose pseudo-inverse and has the advantage of being robust to singularity. The problem with this type of algorithm is that we work in speed, thus, finding the joint angle given an arbitrary Cartesian position is not possible. To remedy to this problem of the first joint angle position, we used a geometrical algorithm which will be described in Section 3.3.

3.3 Geometrical model to solve the IK

By using the HOAP robots, we are limited in the manipulation possibilities. In order to simplify the problem and to ensure that at least one solution can be found, we consider only the hand position for the control of the robot but not the orientation. We have thus 3 parameters ($m = 3$) as input to the system. For the HOAP's arm, we have 4 parameters to find three angles for the shoulder and 1 angle for the elbow [2]. One way to find a suitable solution to the equation is to find a forth input parameter (in order to have m+1 parameters) to define completely the position of the robot's arm for a given position of the hand.

Thus we defined the angle α^e that is the angle between two plans. One plan is

[2]concerning the HOAP 3, the rotation of the wrist can be set separately in order to help to grasp a object, but has no influence on the Cartesian position of the end-effector.

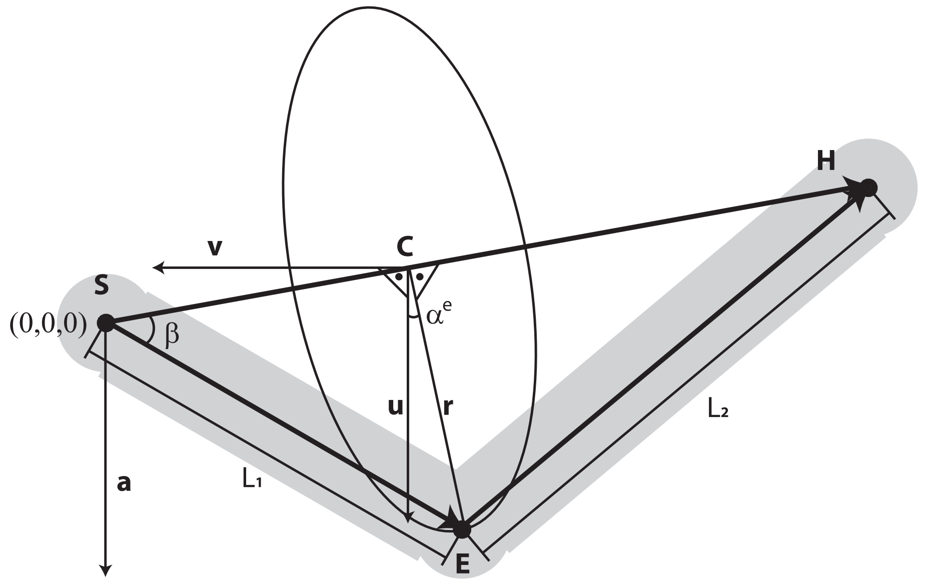

Figure 3.1: The parameter α^e is defined by the angle between a plan passing through the shoulder, the elbow and the hand and a vertical plan passing trough the shoulder and the hand of the robot.

the arm's plan and is defined by the robot's shoulder, elbow and hand. The other plan is a vertical plan passing by the robot's shoulder and hand. A singularity appears with this system when the hand is vertically aligned with the shoulder (see Equation 3.10), but for usual manipulation tasks, this configuration should not appear.

In order to compute the different joint angles given the parameter α^e, we begin by computing the elbow angle. The elbow is the only joint angle that does not depend on α^e.

$$\theta_4 = 180 - acos(\frac{H^2 - L_1^2 - L_2^2}{-2L_1L_2}) \quad \theta_4 \in \{0; \tfrac{2}{\pi}\} \tag{3.5}$$

Where H is the distance between the shoulder and the hand and L_1 and L_2 are respectively the length of the first and second links of the robot's arm. Once we have θ_4, we are able to compute the angle β (see Figure 3.1) and then the vector $\boldsymbol{C}$ that indicates the position of the centre of the circle on which the elbow moves and the radius r of the circle.

$$\beta = \quad asin(\tfrac{L_2 sin(\theta_4)}{H}) \tag{3.6}$$
$$\boldsymbol{C} = \quad cos(\beta)L_1\boldsymbol{n} \qquad \boldsymbol{n} = \frac{\boldsymbol{H}}{H} \tag{3.7}$$
$$r = \quad sin(\beta)L_1 \tag{3.8}$$
$$\tag{3.9}$$

Where $\boldsymbol{n}$ is a normal vector to the elbow circle. We are now able to compute an orthonormal basis $(\boldsymbol{u}, \boldsymbol{v})$ of the circle plan where $\boldsymbol{u}$ is a unity vector defined as the projection of the vertical axes on the circle and $\boldsymbol{v}$ is then a normal vector to the vertical plan.

$$\boldsymbol{u} = \frac{\boldsymbol{a}-(\boldsymbol{a}\cdot\boldsymbol{n})\boldsymbol{n}}{\|\boldsymbol{a}-(\boldsymbol{a}\cdot\boldsymbol{n})\boldsymbol{n}\|} \tag{3.10}$$

$$\boldsymbol{v} = \boldsymbol{n} \times \boldsymbol{u} \tag{3.11}$$

$$\tag{3.12}$$

Where $\boldsymbol{a}$ is a vertical vector with $\|\boldsymbol{a}\| = 1$. If $\boldsymbol{n} \parallel \boldsymbol{a}$, $\boldsymbol{u}$ can not be defined and we have a singularity. If $\boldsymbol{u}$ exists, the position of the elbow can be computed as follows:

$$\boldsymbol{E} = \boldsymbol{C} + r(cos(\alpha^e)\boldsymbol{u} + sin(\alpha^e)\boldsymbol{v}) \tag{3.13}$$

$$\tag{3.14}$$

and the last joint angles are given by:

$$\theta_2 = acos(\frac{E_1}{L_1}) \tag{3.15}$$

$$\theta_1 = acos(\frac{E_3}{-L_1 sin(\theta_2)}) \tag{3.16}$$

$$\theta_3 = asin(\frac{w_1 - L_1 cos(\theta_2) - L_2 cos(\theta_2) cos(\theta_4)}{L_2 sin(\theta_2) sin(\theta_4)}) \tag{3.17}$$

The advantage of this technique is that by defining α^e, it is trivial to find the position of the arm joint angles. Because we have no matrix inversion, it is also quite efficient in terms of computing time. It is then possible to optimize α^e for some criterion like staying away form the joints limits for example. The main drawback of the method is that it works only with this specific configuration. If the robot changes, the model changes, moreover, it can be very difficult to find a suitable model for robot arms with more than 2 links.

3.4 Use human demonstration to find a solution to IK

Human beings have to move their arms all day long without thinking of inverse kinematics solution to perform the task they have in mind. The idea is to benefit from the performance of the human control system by imitation to optimize inverse kinematics and trajectory generation algorithms and to create a controller which will be able to generate human-like movements for manipulation and reaching.

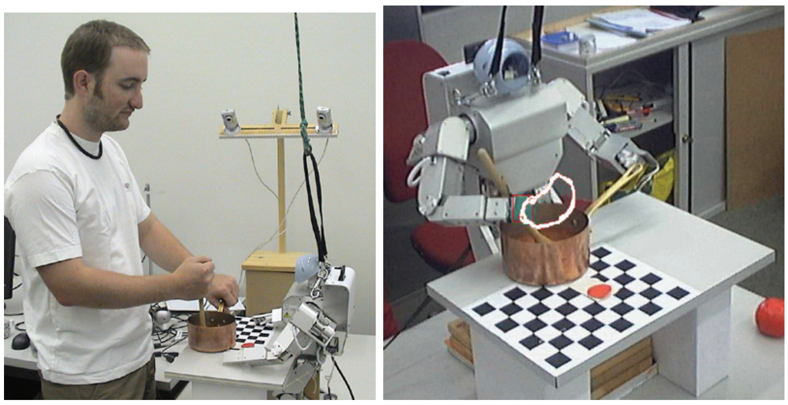

Figure 3.2: Demonstration and reproduction of “stirring the Fondue” with the HOAP 2 robot

In order to do that, we have to solve the “How to Imitate” issue. Even if the differences between the demonstrator and the imitator becomes less important when using a humanoid robot, some problems remain, such as the difference of scale or difference of workspace shape. If we take the HOAP 2 as an example, it is only able to work with objects of small size. If you want to demonstrate movements like “stirring the Fondue”, the robot can reproduce the gesture only using a small saucepan (see Figure 3.2). Even in this case, the space where HOAP 2 is able to reproduce the task is very limited mainly because of its big torso that blocks the movement when the movement comes in the opposite side.

3.4.1 Finding the best balance between the joint angles and the Cartesian coordinates

In order to get the human contribution for solving the IK, the user provides a couple of demonstrations to the robot. These demonstrations can be provided by wearing motion sensors when accomplishing the task or by moving directly the arms of the robot (kinesthetic demonstrations). The data collected during a demonstration are divided into three distinctive sets. Depending on the type of demonstrations, the first set of data are the joint angles of the demonstrator or of the robot. The second set of data are the cartesian coordinates of the robot end-effector or of the demonstrator’s hand. The last data set is the position of the objects on the scene, which is usually used in combination with the second data set in order to compute the position of the end-effector relative to the object.

One of the main idea of the work presented in this chapter is to combine these redundant data in order to find a solution to the “*How to Imitate*” issue and the associated problem of the inverse kinematics. We can give more or less importance to one of the data set relative to the others. For example, for gestures having no relation with the external world as e.g. corporal expression or dancing movements, more importance will be given to the joint angle space.

On the other hand, for movements that have a strong relation with the external world like manipulation tasks, more importance will be given to the cartesian space. The idea is to optimize the balance of the two data spaces to find the best reproduction for the robot.

In order to optimize the trajectory reproduced by the robot, we need to define a cost function which will be used to evaluate the reproduced trajectory. This cost function has to encapsulate the task constraints explicitly as well as to give a measure of their relative importance.

Let us first consider the problem of determining the relevant features of a task and their relative importance. Consider the simple example illustrated in Figure 3.3. There, the demonstrator can reach the two targets (colored dots stamped on either side of the table) with either left or right hand. When the demonstrator reaches for the dot on the left hand-side of the table with his/her left arm, it is said to perform an ipsilateral motion. If conversely the demonstrator reaches for the dot on the right hand-side of the table with his/her left arm, it is said to perform a contralateral motion. In this example, the goals or constraints of the task seem evident. One has to reach for the target while using the proper hand. However, as simple as this might be, such a task might turn to be difficult for a robot to reproduce perfectly. Indeed, consider now the mini-humanoid robot facing the demonstrator in Figure 3.3. When presented with a contralateral motion, the robot is too small to be able to reproduce the exact gesture and reach for the same target (lefthand side). If the robot has to minimize a cost function that gives equal importance to reproduce the gesture as to reach for the same target it might end up producing a silly gesture, i.e. reaching in the air half way toward the target. The "natural" response observed in children as young as 3 years old, is to substitute contralateral for ipsilateral gestures (Bekkering, Wohlschlger, & Gattis, 2000), as illustrated on the righthand side in Figure 3.3.

Note that, interestingly enough, when the dots are removed, i.e. when the actual target on the table is no longer visible, children start reproducing the exact gesture, rather than reaching for the same position in space. Thus, despite the fact that the gesture is the same in both conditions, the presence or absence of a physical object (the dots) affects importantly the imitation strategy. When the object is present, object selection takes the highest priority. Thus, children nearly always direct their imitation to the appropriate target object, at the cost of selecting the "wrong" hand. When removing the dots, the complexity of the task (i.e. the number of constraints to satisfy) is decreased, and hence, constraints of lower importance can be fulfilled (such as producing the same gesture or using the same hand). Similar experiments conducted with adults have corroborated these results by showing that the presence of a physical object affects the reproduction[3].

Such "common" sense is difficult to preprogram in robots. In order to avoid

[3]In that case, the response latency is used instead of the proportion of errors

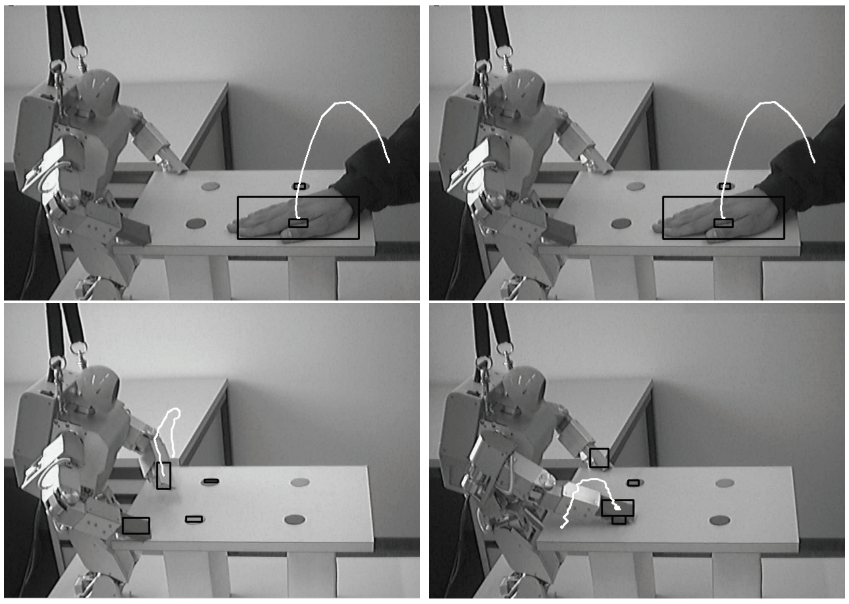
Figure 3.3: In the left side, the importance to reproduce the gesture is equal to the importance to reach the target. In the right side, by adding more importance to the reaching constraint, we obtain a totally different response from HOAP 2.

implementing very heuristic algorithms based on apriori knowledge of the task, the approach here is to generate an optimal reproduction for each arm with respect to a cost function and then to test the different possibilities in simulation (for each arm). By reusing the same cost function the algorithm will be able to find the best solution to reproduce.

3.4.2 Data acquisition and encoding

The first step before trying to resolve the "*How to Imitate*" problem (i.e. what is the optimal way of reproducing the demonstrated task for the robot) is to resolve the "*What to Imitate*" issue (i.e what are the specific constraint that the robot has to extract from the demonstrations). In order to have usable data for the reproduction, we have to acquire demonstration data and to analyze it. For that, probabilistic algorithms developed by Sylvain Calinon have been used (Calinon, 2007).

Let $\boldsymbol{\xi}(t) \in \mathbb{R}^s$ describe the set of demonstrations provided to the robot. Learning of the task's constraints is done in a probabilistic manner, namely by encoding a set of demonstrated trajectories in a Gaussian Mixture Model (GMM) and by retrieving a *generalized* trajectory through Gaussian Mixture Regression (GMR) (Ghahramani & Jordan, 1994). Next, we briefly summarize this procedure (see (Calinon, 2007) for details). The method models the joint distribution of an "input" and an "output" variable as a Gaussian Mixture Model (GMM) with $K \in \mathbb{N}$ states (see Figure 3.4). In our case, the output variables are the positions $\boldsymbol{\xi}$ and the input variable is the time t. If we join those

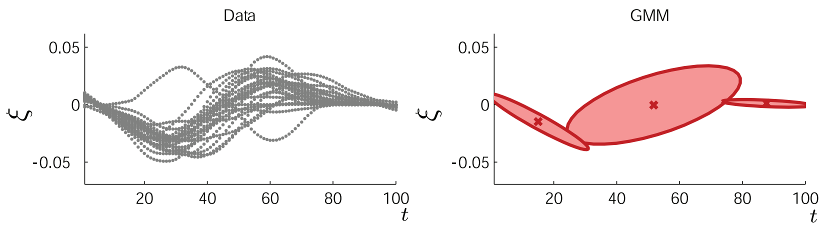

Figure 3.4: This graphic shows a example of how the data are encoded using a GMM with 3 states. The data are the demonstrations for one dimension of the arm joint angles.

variables in a vector $\boldsymbol{\upsilon} = \begin{bmatrix} t \\ \boldsymbol{\xi} \end{bmatrix} \in \mathbb{R}^{s+1}$, it is possible to model its probability density function as:

$$p(\upsilon) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{\upsilon}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3.18}$$

where π_k is a weighting factor and $\mathcal{N}(\boldsymbol{\upsilon}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is a Gaussian function with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$:

$$\mathcal{N}(\boldsymbol{\upsilon}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_k|}} e^{(-\frac{1}{2}(\upsilon - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\upsilon - \boldsymbol{\mu}_k))} \tag{3.19}$$

where d is the dimensionality of the vector υ. The mean vector $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$ can be separated into their respective input and output components:

$$\boldsymbol{\mu}_k = [\boldsymbol{\mu}_{k,t}^T \; \boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}^T]^T \tag{3.20}$$

$$\boldsymbol{\Sigma}_k = \begin{pmatrix} \boldsymbol{\Sigma}_{k,t} & \boldsymbol{\Sigma}_{k,t\dot{\boldsymbol{\xi}}} \\ \boldsymbol{\Sigma}_{k,\dot{\boldsymbol{\xi}}t} & \boldsymbol{\Sigma}_{k,\dot{\boldsymbol{\xi}}} \end{pmatrix} \tag{3.21}$$

This GMM can be trained using a standard E-M algorithm, taking the demonstrations as training data. We thus obtain a joint probability density function for the input and the output. Because it is a GMM, the conditional probability density function, i.e., the probability of the output conditioned on the input is also a GMM. Hence, it is possible, after training, to recover the expected output variable $\boldsymbol{\xi}$, given the observed input variable t (see Figure 3.5).

$$\boldsymbol{\xi}(t) = \sum_{k=1}^{K} h_k(t) \big(\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}} + \boldsymbol{\Sigma}_{k,\dot{\boldsymbol{\xi}}t} \boldsymbol{\Sigma}_{k,t}^{-1} (t - \boldsymbol{\mu}_{k,t}) \big) \tag{3.22}$$

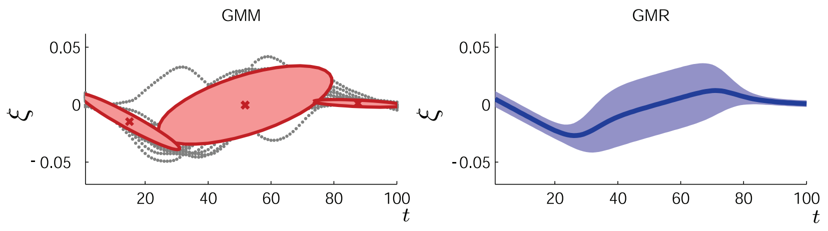

Figure 3.5: This graphic shows how the expected trajectory and its variance are retrieved using GMR. The expected output is for one dimension of the arm joint angles.

where the $h_k(t)$ are given by:

$$h_k(t) = \frac{\pi_k \mathcal{N}(t; \boldsymbol{\mu}_{k,t}, \boldsymbol{\Sigma}_{k,t})}{\sum_{k=1}^{K} \pi_k \mathcal{N}(t; \boldsymbol{\mu}_{k,t}, \boldsymbol{\Sigma}_{k,t})} \tag{3.23}$$

The variance of this conditional probability distribution is given by:

$$\boldsymbol{\Sigma}_{\xi}(t) = \sum_{k=1}^{K} h_k^2(t) \big(\boldsymbol{\Sigma}_{k,\boldsymbol{\xi}} - \boldsymbol{\Sigma}_{k,\boldsymbol{\xi}t} \boldsymbol{\Sigma}_{k,t}^{-1} \boldsymbol{\Sigma}_{k,t\boldsymbol{\xi}} \big) \tag{3.24}$$

Thus, in our application, after training, the GMM can be used to generate a trajectory by taking the expected positions $\boldsymbol{\xi}(t)$ conditioned on time t. This trajectory is taken to be the one to imitate. Moreover, the variances of the GMM can provide an indication about the variability of the observed variables. At any given time, variables with low variability across demonstrations can be interpreted as more relevant to the task than variables with high variability.

3.4.3 Determining the Cost Function

In order to optimize the trajectory reproduced by the robot, we need to define a cost function which will be used to evaluate the reproduced trajectory. This cost function has to encapsulate explicitly the task constraints as well as to measure the relative importance of the task constraints.

Unidimensional case

Let $\xi(t)$ and $\xi^r(t)$ where $t = \{1, 2, \ldots, T\}$ be respectively the generalized trajectory and the reproduction trajectory of a variable ξ. The cost function H is defined by:

$$
\begin{aligned}
H(\xi,\xi^r) &= 1 - f(e) \\
e &= \frac{1}{T}\sum_{t=1}^{T} |\xi_t^r - \xi_t|
\end{aligned}
\tag{3.25}
$$

Where function $f()$ is described in Equation 3.28. $H \in [0,1]$ gives an estimate of the quality of the reproduction. Optimizing the imitation consists of minimizing H. H=0 corresponds to a perfect reproduction. e is a measure of deviation between the generalized data ξ and the reproduced data ξ^r.

Multidimensional case

Let us consider a dataset of K variables. The complete cost function H_{tot} is given by:

$$
H_{tot} = \frac{1}{K}\sum_{i=1}^{K} w_i \; H(\xi,\xi^r) \tag{3.26}
$$

where $w_i \in [0,1]$ is a measure of the relative importance of each cost function. These weights reflect the variance of the data during the demonstration. To evaluate this variability, we use the statistical representation provided by the GMM trained on the demonstration data.

$$
w_i = f\left(\frac{1}{T}\sum_{t=1}^{T}\frac{1}{\Sigma_t^i}\right) \tag{3.27}
$$

Where Σ_t^i is the variance of a given variable i at time t. If this variance is high, i.e. showing no consistency across demonstrations, this suggests that satisfying some particular constraints on this variable will have little bearing on the task. The factors w_i in the cost function equation reflects this assumption: if the standard deviation of a given variable is low, the value taken by the corresponding w_i are close to 1. This way, the corresponding variable will have a strong influence in the reproduction of the task.

A transformation function $f()$ normalizes and bounds each variable within minimal and maximal values (see Equation 3.28). This eliminates the effect of the noise, intrinsic to each variable, so that the relative importance of each variable can be compared.

$$
f(v) = \begin{cases} 1 & \text{if } v \leq v_{min} \\ \frac{v_{max}-v}{v_{max}-v_{min}} & \text{if } v_{min} < v < v_{max} \\ 0 & \text{if } v \geq v_{max} \end{cases} \tag{3.28}
$$

$v_{min} = 0$ in our case corresponds to a derivative equal to zero and v_{max} represents the standard deviation of the distribution of the derivative from its minimum 0 to its maximum.

Relative importance of the constraints

The cost function is thus applied to 3 sets of variables, namely the joint angle trajectories, the hand path and the location of the objects at which actions are directed. $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \boldsymbol{\theta}_4\}$ are the generalized joint angle trajectories over the demonstrations, $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$ the generalized cartesian trajectory of the hand and $\{o_1, o_2, o_3\}$ the 3D location of the goal (object). We compute $y_j = x_j - o_j$ a difference measure for component j between the hand and the goal, along the trajectory.

For $n = 4$ joint angles for each arm and $m = 3$ variables to describe the position of the hands and objects in cartesian space (we make the assumption that only one hand is used at the same time), we define the general cost function H_{tot} as:

$$\begin{aligned} H_{tot} &= \alpha_1 \frac{1}{n} \sum_{i=1}^{n} w_{1,i}\; H_1(\boldsymbol{\theta}_i, \boldsymbol{\theta}_i^r) \\ &+ \alpha_2 \frac{1}{m} \sum_{j=1}^{m} w_{2,j}\; H_2(\boldsymbol{x}_j, \boldsymbol{x}_j^r) \\ &+ \alpha_3 \frac{1}{m} \sum_{j=1}^{m} w_{3,j}\; H_3(\boldsymbol{y}_j, \boldsymbol{y}_j^r) \end{aligned} \tag{3.29}$$

The new factors α_i, with $\sum_i \alpha_i = 1$ and $0 \leq \alpha_i \leq 1$, are fixed by the experimenter, and determine the relative importance of each subset of data, i.e. the importance of each constraint (or goal) in the overall task (α_1 for the joint angle trajectories, α_2 for the hand path and α_3 for the hand-object distance. The cost function $H_{1,2,3}$ are given by Equation 3.26 and the weights $w_{1,2,3}^j$ follow Equation 3.27, and are set to 0 if the corresponding component is missing (e.g. if the object is not detected by the vision system).

Optimizing the imitation

Once the cost function and the relative influence of each constraint have been determined, the goal is to generate an optimal (with respect to the cost function H) trajectory. To compute the cost function, we need to generate a "model" trajectory for the hand path. For that, we use the trajectory generated by the demonstrator and encoded by GMM, like described in Section 3.4.2. By default, we assume that the robot performs a mirror imitation, i.e. moving its left arm for a motion of the demonstrator's right arm and vice-versa. In order to accomplish the mirror imitation, we apply a vertical translation to all the trajectories.

Because of the different embodiments (smaller size, different joint limits) between the demonstrator and the robot, if the demonstration is performed directly by the user, we have to rescale the trajectories, so that the demonstrated trajectory lies within the workspace of the robot. The trajectories are then interpolated from the set of key points, using a 3rd-order spline fit. The new trajectories are, then, considered as candidate hand paths.

To generate the joint angles trajectories corresponding to these hand paths, we have to solve the inverse kinematics equation given by: $\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{\theta}}$, where $\boldsymbol{J}$ is the Jacobian. In this example, the solution to the IK is found using the pseudo-inverse with optimization numerical solution (also described in Section 3.2).

$$\dot{\boldsymbol{\theta}}^c = \boldsymbol{J}^+\dot{\boldsymbol{x}} + \alpha(\boldsymbol{I} - \boldsymbol{J}^+\boldsymbol{J})\frac{\partial g}{\partial \boldsymbol{\theta}} \tag{3.30}$$

Where

$$\mathbf{J}^+ = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1} \tag{3.31}$$

And

$$g(\boldsymbol{\theta}) = |\boldsymbol{\theta}^{rest} - \boldsymbol{\theta}| \tag{3.32}$$

$\mathbf{J}^+$ is the Moore-Penrose inverse and $g(\boldsymbol{\theta})$ is a optimization term which tends to minimize the distance between the arm position and a rest position, determined usually in our work as the middle range of each joint angle. To avoid the singularities, we consider that we work in a bounded workspace.

$\dot{\boldsymbol{\theta}}^c$, as defined in Equation 3.30 returns a joint trajectory satisfying the desired Cartesian trajectories. In order to take into account the observation of the demonstrator's joint trajectories, we add a second constraint to the pseudo-inverse, given by:

$$\dot{\boldsymbol{\theta}} = \gamma\dot{\boldsymbol{\theta}}^c + (1-\gamma)\dot{\boldsymbol{\theta}}^j \tag{3.33}$$

where $\dot{\boldsymbol{\theta}}^j$ is the derivative of the joint angle trajectory of the demonstrator generated by the GMM after training, and γ is a factor used to tune the influence of the two different terms (reproduction of hand path or joint angle trajectories).

Gradient descent

The factor γ in Equation 3.33 determines the relative influence of the cartesian and angular trajectories. Its value affects the general cost function H^{tot} (for simplicity, we will refer to the cost function as H in the remaining part of this chapter). In order to find the optimal imitation strategy, we need to compute the optimal value of γ for a given cost function $H(\gamma)$. Since there is no analytical solution, we have to turn towards numerical solutions.

In the general case, $H(\gamma)$ might have more than one minimum, see the example given in Figure 3.6. In order to find the global minimum of the $H(\gamma)$ curve, we follow a two steps algorithm. First, we estimate roughly the zeros of the derivative $\frac{dH}{d\gamma}$. This gives us an estimate of the number of "hills" formed by the cost function. Second, we search for the minimum of each hill, by gradient descent along the curve across each pair of (zero-derivative) points. Finally, we compute the global minima as the minimum of all minima. Such an algorithm has the advantage to require far less computation time than alternative hill-climbing search algorithm, such as Genetic Algorithm or conjugate gradient descent.

The zeros of the derivative are estimated incrementally. At initialization, we set $\gamma_{t=0} = \delta$ with $\delta << 1$. Then, at each time step, we approximate $\frac{dH}{d\gamma}$ following Equation 3.34.

$$\frac{\widehat{dH(\gamma_t)}}{d\gamma} = \frac{H(\gamma_t + \delta) - H(\gamma_t - \delta)}{2\delta} \tag{3.34}$$

The next iteration point γ_{t+1} is given by:

$$\gamma_{t+1} = \gamma_t + \Delta\gamma \tag{3.35}$$

where

$$\Delta\gamma = \frac{\beta}{\sqrt{\frac{\widehat{dH(\gamma_t)}}{d\gamma}}} \tag{3.36}$$

The denominator in Equation 3.36 reduces the effect of very steep gradient (slowing down the descent). β fixes the balance between minimizing the number of points required for the computation and having a sufficiently high resolution to ensure that one determines all hills. At each iteration, we determine that we have passed a zero (which is a minimum) if the following two conditions are satisfied: $\frac{\widehat{dH(\gamma_{t-1})}}{d\gamma} < 0$ and $\frac{\widehat{dH(\gamma_t)}}{d\gamma} > 0$. When such a zero is passed, we compute two values, the position of the i^{th} zero γ_i^z and the approximation Λ_i of the second derivative to have the curvature in $H(\gamma_i^z)$. Λ will be used to tune the coefficient of the gradient descent. Knowing $\frac{\widehat{dH(\gamma_{t-1})}}{d\gamma}$ and $\frac{\widehat{dH(\gamma_t)}}{d\gamma}$, γ^z is given by:

$$\gamma_i^z = \frac{\frac{\widehat{dH(\gamma_{t-1})}}{d\gamma} * \gamma_t - \frac{\widehat{dH(\gamma_t)}}{d\gamma} * \gamma_{t-1}}{\frac{\widehat{dH(\gamma_{t-1})}}{d\gamma} - \frac{\widehat{dH(\gamma_t)}}{d\gamma}} \tag{3.37}$$

and the curvature Λ is given by:

$$\Lambda_i = \frac{\frac{\widehat{dJ(\gamma_t)}}{d\gamma} - \frac{\widehat{dH(\gamma_{t-1})}}{d\gamma}}{\gamma_t - \gamma_{t-1}} \tag{3.38}$$

The approximation of the third derivative is not very precise. However, the advantage is that we do not need to compute more points than those we already have. This improves the speed of computation, because for each $\frac{\widehat{dH(\gamma_t)}}{d\gamma}$ we have to compute H only twice.

To find the global minimum of the function $H(\gamma)$, we launch a gradient descent algorithm on each zero found for the function. The next iteration γ_{t+1} of the gradient descent is given by:

$$\gamma_{t+1} = \gamma_t + \rho(-\frac{\widehat{dH(\gamma_t)}}{d\gamma}) \tag{3.39}$$

Where $\frac{\widehat{dH(\gamma_t)}}{d\gamma}$ is given by Equation 3.34. For convergence in all cases with a reasonable speed, we tune the parameter ρ with the slope Λ computed during the rough research of the zeros (Equation 3.38).

$$\rho = \frac{\mu}{\Lambda^2} \tag{3.40}$$

where μ is a constant and the squared term is needed to have a decreasing ρ for big slopes. If μ is well tuned, the algorithm converges with a good speed in all cases, narrow hole or flat basin (see Figure 3.6).

Experiments

In order to demonstrate that the algorithm generalizes to different tasks, we conducted two sets of experiments. In the first experiment, the robot has been shown different reaching motions, following the scenario described in Section 3.4.1. There, the task consisted in optimizing different constraints, depending on the situation. In the absence of dots, the task constraint consisted in only reproducing the gesture. In the presence of dots, the task constraints consisted in reproducing both gesture and dots (Calinon, Guenter, & Billard, 2005).

In the second experiment, the robot had to learn how to stir a “fondue” (typical swiss meal), i.e. it had to learn to perform cyclic rotational motions

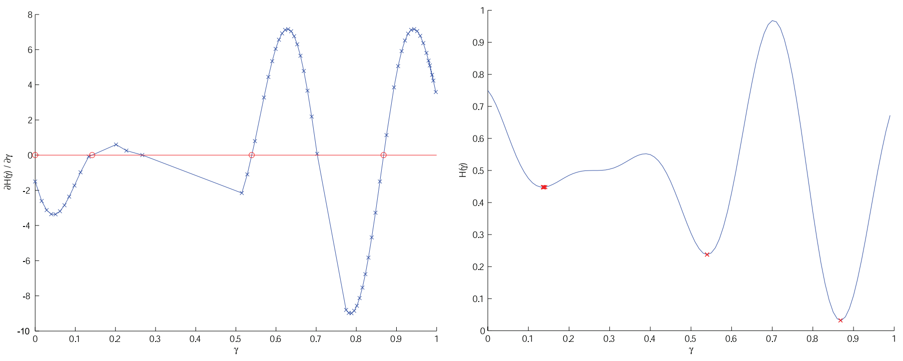

Figure 3.6: In the left side diagram, we first roughly approximate the zeros with $\widehat{\frac{dH(\gamma)}{d\gamma}}$ - cross points are the iteration points and circle points are the zero approximations - and then, in the right side diagram, we launch a gradient descent for each zeros found

while keeping the stick inside the saucepan, see Figure 3.12. After having shown a first example, the robot was required to reproduce stirring the fondue in different locations in space. The change in location of the saucepan puts different constraints on the cost function. The robot has to show that it is capable of generalizing and applying a gesture learned in a first context into a different one.

Experimental setup

The experiments were conducted with a Fujitsu HOAP 2 humanoid robot. In these experiments, trajectory control affected only the two arms (4 DOFs each). The torso and legs were set to a constant position to support the robot's stand-up posture.

The demonstrator's motions were recorded by a set of 5 Xsens motion sensors, attached to the torso and the upper- and lower-arms. Each sensor provides the 3D absolute orientation of each segment, by integrating the 3D rate-of-turn, acceleration and earth-magnetic field, at a rate of 100Hz. The angular trajectories of the shoulder joint (3 DOFs) and the elbow (1 DOF) are reconstructed by taking the torso as referential, with an accuracy of 1.5 degrees.

A color-based stereoscopic vision system tracks the 3D-position of the dots (1st experiment) and the saucepan (2nd experiment), the demonstrator's hands, and the robot's hands at a rate of 15Hz, with an accuracy of 10 mm.

Results

1st Experiment

In this experiment, we set $\alpha_1 = \frac{1}{7}$, $\alpha_2 = \frac{2}{7}$, $\alpha_3 = \frac{4}{7}$. In other words, we set the constraint "reaching the target" as being twice as important as the constraint on the "reproducing the hand path" and four times more important than the

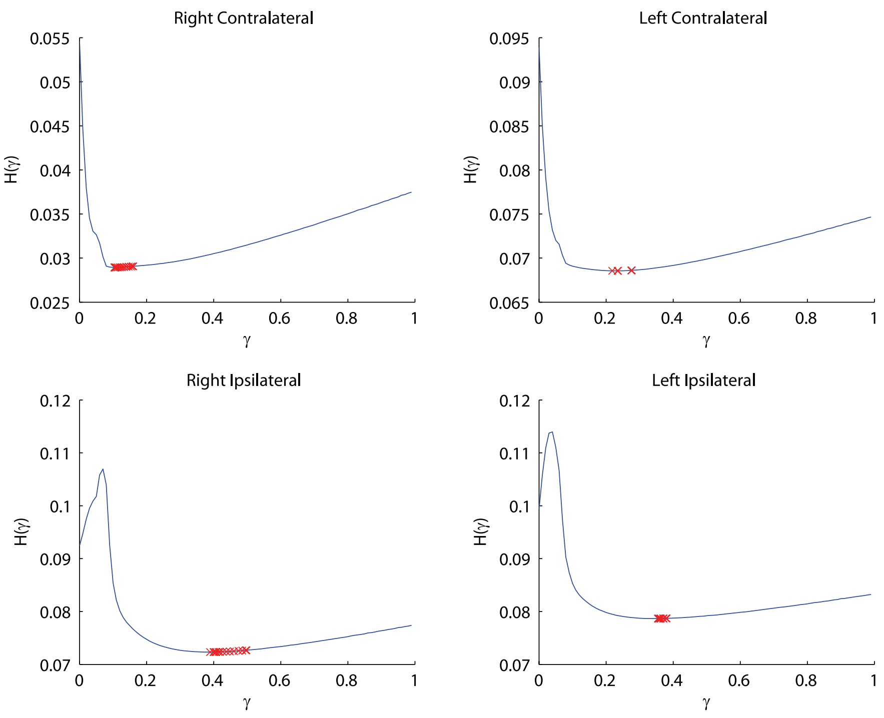

Figure 3.7: Cost function $H(\gamma)$ for a demonstration of a contralateral movement with the left arm. The crosses represent the gradient descent iterations.

constraint on "reproducing the gesture".

Figures 3.7 and 3.8 show the value of the cost function $H(\gamma)$. The cross points represent the iteration points of the gradient descent algorithm along the cost function. In each case, the algorithm converges to the global minimum in a maximum of 30 steps.

In Figure 3.7 we can see that the best response to a contralateral movement of the left arm is a contralateral movement of the right arm. However, when the target (the dot) is present (Figure 3.8), the best response, for the same gesture, is an ipsilateral movement, produced with the left hand. This response is the desired one, i.e. the one corresponding to the "common sense". This decision is due to the fact that the robot cannot reach the target with a right contralateral movement because of its limited joint configurations (i.e. the robot is not able to cross the arms sufficiently Figure 3.3).

2nd Experiment

In this experiment, the importance given to the constraints (gesture versus hand location) were set to be equal, i.e. $\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{3}$. The point of this experiment was to see how the system would react if we would transpose the constraints on the hand position to different location within the robot's workspace, while keeping the constraint on the joint trajectory unchanged. Would the optimal solution change with the cartesian position of the movement?

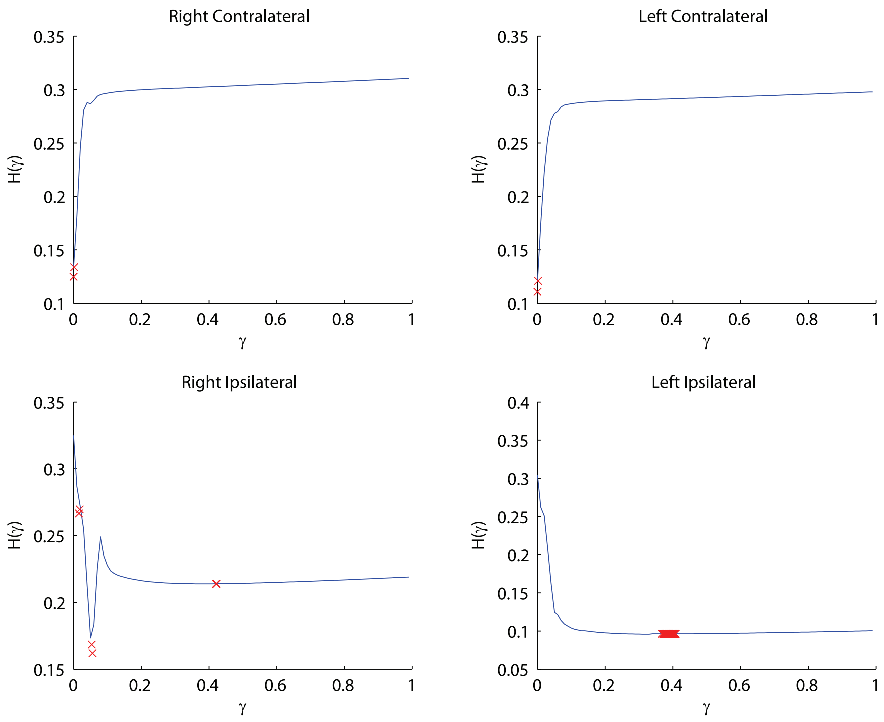

Figure 3.8: Cost function $H(\gamma)$ for a demonstration of a contralateral movement with the left arm and with a goal (crosses represent the gradient descent iterations).

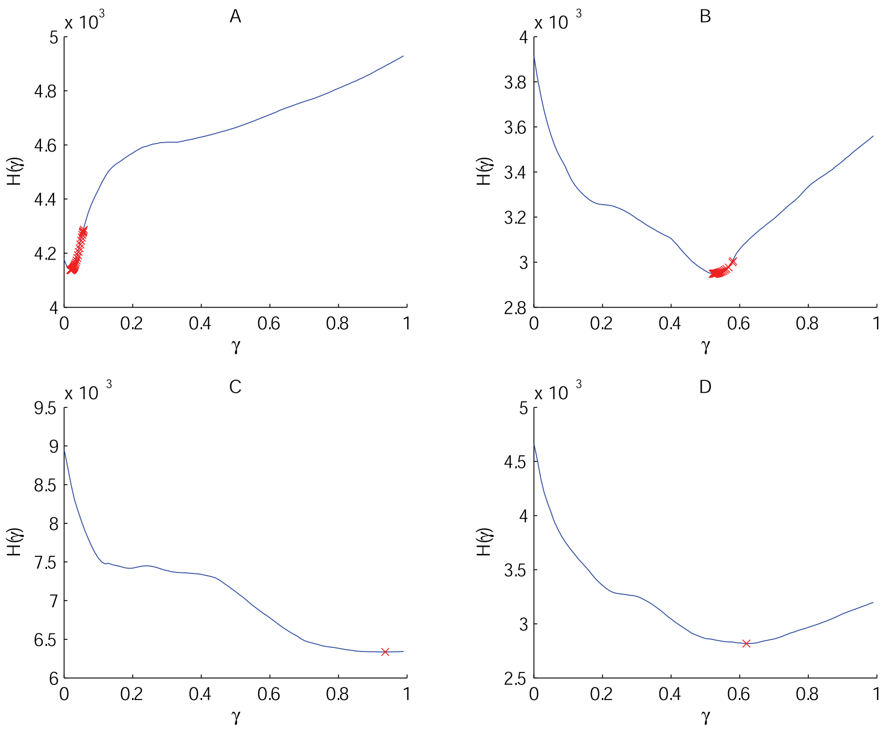

Figure 3.9: Value taken by the cost function $H(\gamma)$ for four locations in the robot's workspace, as illustrated in Figure 3.12. Position A (top-left) corresponds to the location of the saucepan in the original demonstration.

Figure 3.9 shows the value taken by the task's cost function $H(\gamma)$ for four locations in the robot's workspace, as illustrated in Figure 3.12. Position A (top-left) corresponds to the location of the saucepan in the original demonstration. Again, in each case, the algorithm converges to the global minimum in a maximum of $N*30$ steps where N is the number of zeros given by the first part of the algorithm.

As expected, when in location A, the optimum of the cost function corresponds to reproducing the gesture ($\gamma \rightarrow 0$). Indeed, at this location, both constraints (gesture and location of the saucepan/hand) are equivalent (the location can be computed directly from the forward kinematics) and, thus, satisfying one is enough. In contrast, in locations B and D the optimal solution consists in satisfying partly either the hand path or the gesture ($\gamma \rightarrow 0.5$). This is due to the fact that, if we were to reproduce faithfully the joint trajectories only ($\gamma = 0$), we would end up with a large discrepancy along the hand path. This deformation of the trajectory is due to the fact that the starting points of demonstrated and reproduced trajectories differ. Moreover, the translated trajectory might in some case reach the limits of the robot's workspace. Finally, when in location C, satisfying the hand path is optimal. However, the absolute error is larger in this situation (even for the minimal position) than in any of the other 3 situations.

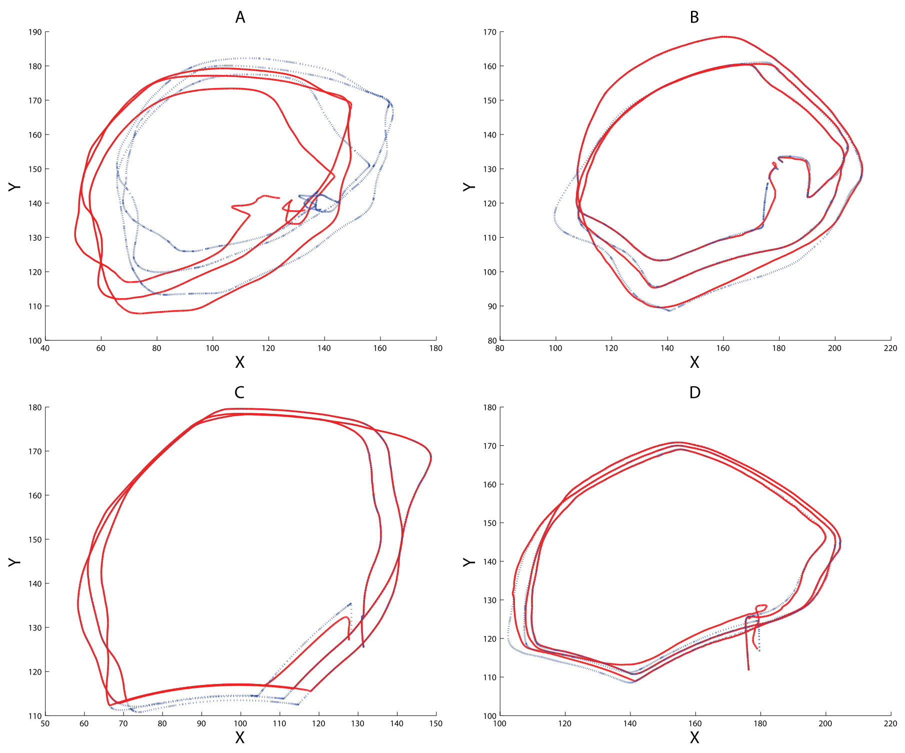

Figure 3.10: Hand paths of the demonstrator (blue dashed line) and of the imitator (red solid line) for the four locations in the experiment "stirring the fondue", see Figure 3.12

As shown in Figure 6, this reflects the fact that the qualitative features of the hand path (the stirring motion) are correctly reproduced, while the whole trajectory is shifted in space, resulting in large errors on the mean square distance. In contrast, the location C is placed higher up in front of the robot, forcing a shift in the oscillatory motion to the elbow and the shoulder adduction-abduction joints, as shown in Figure 3.11.

3.4.4 Constraints-Based Inverse Kinematics

In the Cost Function defined in Section 3.4.3, we take into account the variance of each variable separately, but the covariance information are not used for the evaluation of the reproduced trajectory. We will define here a new generic similarity measure H that takes into account the variations of constraints and the dependencies across the variables which have been learned over time. The metric is continuous, positive, and can be estimated at any point along the trajectory.

As before, let $\{\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}\}$ be, respectively, the generalized joint angle trajectories from the demonstrations, the generalized hand paths from the demonstrations, and the generalized object-hand relationships observed during the demonstrations. Let $\{\boldsymbol{\theta}^r, \boldsymbol{x}^r, \boldsymbol{y}^r\}$ be the candidate trajectories for reproducing

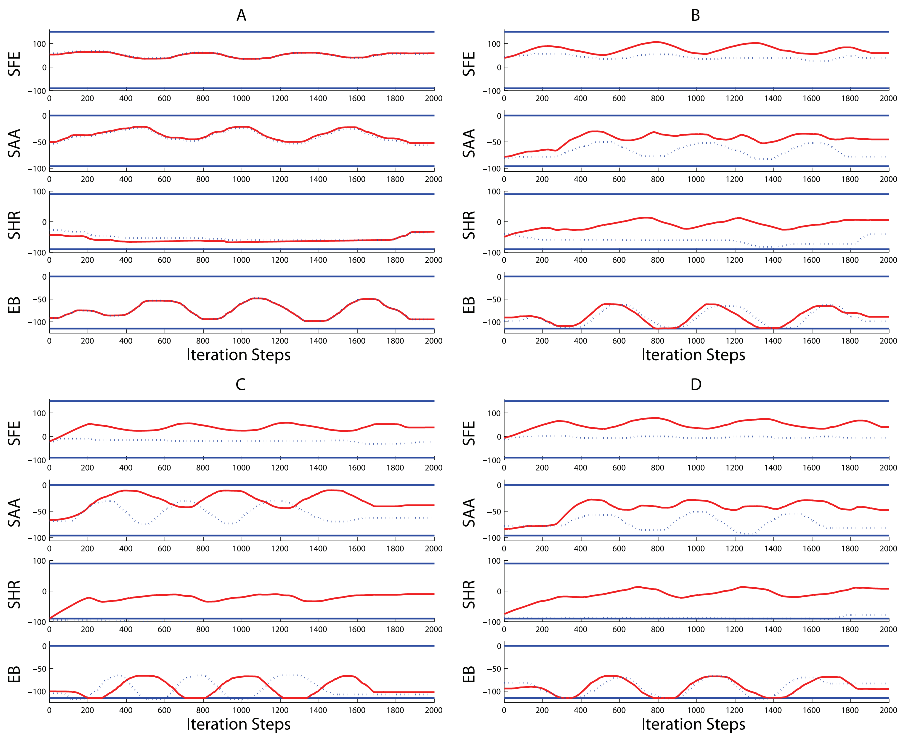

Figure 3.11: Joint trajectory of the demonstrator (blue dashed line) and of the imitator (red solid line) for the four locations in the experiment ”stirring the fondue”, see Figure 3.12

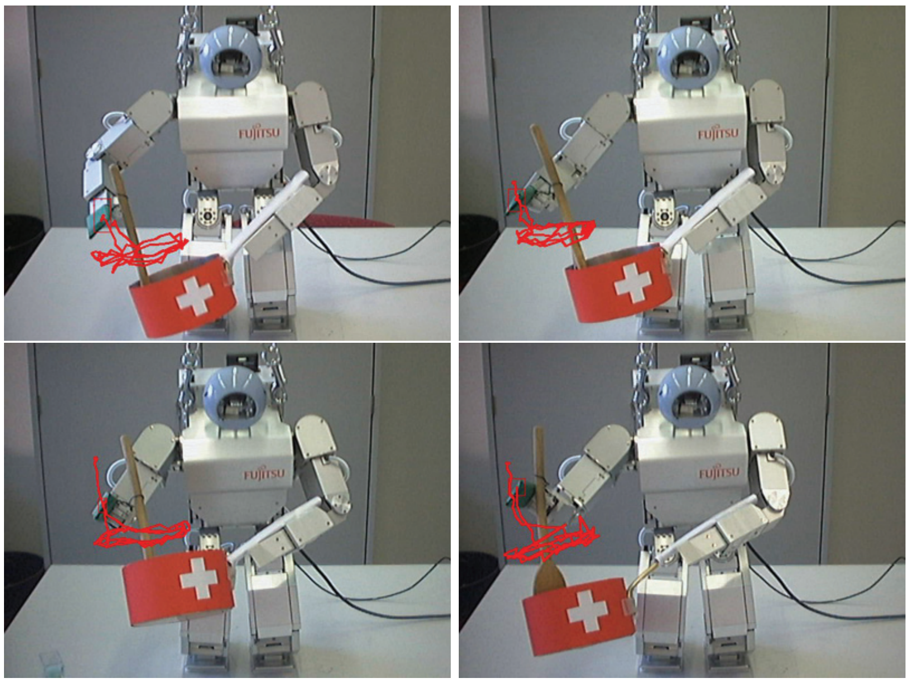

Figure 3.12: The reproduction of the best solution for the four locations A, B, C and D using the joint angle constraints extracted from the demonstrations done at position A

the motion. The new metric of imitation performance (i.e. cost function for the task) H is then given by:

$$\begin{aligned} H &= (\boldsymbol{\theta}^r - \boldsymbol{\theta})^T \, \boldsymbol{W}^\theta \, (\boldsymbol{\theta}^r - \boldsymbol{\theta}) \\ &+ \ (\boldsymbol{x}^r - \boldsymbol{x})^T \, \boldsymbol{W}^x \, (\boldsymbol{x}^r - \boldsymbol{x}) \\ &+ \ (\boldsymbol{y}^r - \boldsymbol{y})^T \, \boldsymbol{W}^y \, (\boldsymbol{y}^r - \boldsymbol{y}) \end{aligned} \tag{3.41}$$

In our model, $\boldsymbol{W} \in \{\boldsymbol{W}^\theta, \boldsymbol{W}^x, \boldsymbol{W}^y\}$ are weighting matrices which represent the time-varying constraints during the task. They can be measured using different cues. The statistical variations and relations across the different variables $\boldsymbol{\Sigma} \in \{\boldsymbol{\Sigma}^\theta, \boldsymbol{\Sigma}^x, \boldsymbol{\Sigma}^y\}$ serve as a basis to represent the constraints (see Equation 3.24). Thus, we define:

$$\boldsymbol{W} = (\Sigma)^{-1} \tag{3.42}$$

We consider the additional variables $\boldsymbol{c}_1$, $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$ defined by:

$$\begin{array}{rcl|l} c_{1,i,k} & = & \theta_{i,k} - \theta^r_{i,k-1} & \forall i \in \{1, \dots, n\} \\ c_{2,j,k} & = & x_{j,k} - x^r_{j,k-1} & \forall j \in \{1, \dots, m\} \\ c_{3,j,k} & = & y_{j,k} - y^r_{j,k-1} & \forall k \in \{2, \dots, T\} \end{array}$$

and rewrite (3.41) as:

$$\begin{aligned} H &= (\dot{\boldsymbol{\theta}}^r - \boldsymbol{c}_1)^T \, \boldsymbol{W}^\theta \, (\dot{\boldsymbol{\theta}}^r - \boldsymbol{c}_1) \\ &+ \ (\dot{\boldsymbol{x}}^r - \boldsymbol{c}_2)^T \, \boldsymbol{W}^x \, (\dot{\boldsymbol{x}}^r - \boldsymbol{c}_2) \\ &+ \ (\dot{\boldsymbol{y}}^r - \boldsymbol{c}_3)^T \, \boldsymbol{W}^y \, (\dot{\boldsymbol{y}}^r - \boldsymbol{c}_3) \end{aligned}$$

3.4.5 Finding an optimal controller

The problem is now reduced to finding a minimum of (3.43), when subjected to the body constraint ($\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{\theta}}$) and environmental constraint ($\dot{\boldsymbol{y}} = \dot{\boldsymbol{x}} - \dot{\boldsymbol{o}}$) where $\dot{\boldsymbol{o}}$ denotes the speed of the object in the robot referential. Since $\boldsymbol{H}$ is a quadratic function, the problem can be solved analytically by *Lagrange* optimization. We define the *Lagrangian* as:

$$\begin{array}{rcl} L(\dot{\boldsymbol{\theta}}^r, \dot{\boldsymbol{x}}^r, \dot{\boldsymbol{y}}^r, \lambda_1, \lambda_2) & = & (\dot{\boldsymbol{\theta}}^r - \boldsymbol{c}_1)^T \ \boldsymbol{W}^\theta \ (\dot{\boldsymbol{\theta}}^r - \boldsymbol{c}_1) \\ & + & (\dot{\boldsymbol{x}}^r - \boldsymbol{c}_2)^T \ \boldsymbol{W}^x \ (\dot{\boldsymbol{x}}^r - \boldsymbol{c}_2) \\ & + & (\dot{\boldsymbol{y}}^r - \boldsymbol{c}_3)^T \ \boldsymbol{W}^y \ (\dot{\boldsymbol{y}}^r - \boldsymbol{c}_3) \\ & + & \lambda_1^T (\dot{\boldsymbol{x}} - \boldsymbol{J} \ \dot{\boldsymbol{\theta}}) \\ & + & \lambda_2^T (\dot{\boldsymbol{y}} - (\dot{\boldsymbol{x}} - \dot{\boldsymbol{o}})) \end{array}$$

where λ_1 and λ_2 are the vectors of associated *Lagrange* multipliers.

We have now to solve $\nabla L(\dot{\boldsymbol{\theta}}^r, \dot{\boldsymbol{x}}^r, \dot{\boldsymbol{y}}^r, \lambda_1, \lambda_2) = 0$.

Considering symmetric matrices $W^T = W$, we use the following second order derivative properties:

$$\boxed{\frac{\partial}{\partial x}(Ax - s)^T W (Ax - s) = -2A^T W (Ax - s)}$$

$$\boxed{\frac{\partial}{\partial x} Ax = A^T}$$

Solving $\frac{\partial L}{\partial \dot{\boldsymbol{\theta}}^r} = 0$, we find:

$$-2 \ \boldsymbol{W}^\theta \ (\dot{\boldsymbol{\theta}}^r - \boldsymbol{c}_1) - \boldsymbol{J}^T \lambda_1 = 0 \tag{3.43}$$

Solving $\frac{\partial L}{\partial \dot{\boldsymbol{x}}^r} = 0$, we find:

$$-2 \ \boldsymbol{W}^x \ (\dot{\boldsymbol{x}}^r - \boldsymbol{c}_2) + \lambda_1 - \lambda_2 = 0 \tag{3.44}$$

Solving $\frac{\partial L}{\partial \dot{\boldsymbol{y}}^r} = 0$, we find:

$$-2 \ \boldsymbol{W}^y \ (\dot{\boldsymbol{y}}^r - \boldsymbol{c}_3) + \lambda_2 = 0 \tag{3.45}$$

Using (3.44) and (3.45), we find:

$$\lambda_1 = 2 \ \boldsymbol{W}^x \ (\dot{\boldsymbol{x}}^r - \boldsymbol{c}_2) + \ 2 \ \boldsymbol{W}^y \ (\dot{\boldsymbol{x}}^r - \boldsymbol{c}_3) \tag{3.46}$$

Using (3.46) and (3.43), we find:

$$\boldsymbol{W}^\theta \ (\dot{\boldsymbol{\theta}}^r - \boldsymbol{c}_1) + \boldsymbol{J}^T \boldsymbol{W}^x \ (\dot{\boldsymbol{x}}^r - \boldsymbol{c}_2) + \boldsymbol{J}^T \ \boldsymbol{W}^y \ (\dot{\boldsymbol{y}}^r - \boldsymbol{c}_3) = 0$$

Solving for $\dot{\boldsymbol{\theta}}^r$, we obtain:

$$\begin{array}{rcl} \dot{\boldsymbol{\theta}}^r & = & \left(W^\theta + \boldsymbol{J}^T \boldsymbol{W}^x \boldsymbol{J} + \boldsymbol{J}^T \boldsymbol{W}^y \boldsymbol{J})\right)^{-1} \\ & \times & \left(\boldsymbol{W}^\theta \ \boldsymbol{c}_1 + \boldsymbol{J}^T \boldsymbol{W}^x \ \boldsymbol{c}_2 + \boldsymbol{J}^T \boldsymbol{W}^y \boldsymbol{c}_4\right) \end{array}$$

with:

$$\boldsymbol{c}_4 \quad = \quad \dot{\boldsymbol{o}}^r + \boldsymbol{c}_3$$

Note that this expression can be generalized to take into account G different objects:

$$\begin{aligned}\dot{\boldsymbol{\theta}}^r \quad &= \quad \left(\boldsymbol{W}^\theta + \boldsymbol{J}^T\boldsymbol{W}^x\boldsymbol{J} + \sum_{i=1}^{G} \boldsymbol{J}^T\boldsymbol{W}_i^y\boldsymbol{J}\right)^{-1} \\ &\times \quad \left(\boldsymbol{W}^\theta\,\boldsymbol{c}_1 + \boldsymbol{J}^T\boldsymbol{W}^x\,\boldsymbol{c}_2 + \sum_{i=1}^{G} \boldsymbol{J}^T\boldsymbol{W}_i^y\boldsymbol{c}_{4i}\right) \qquad (3.47)\end{aligned}$$

We are now able to compute iteratively the joint angle trajectories with:

$$\boldsymbol{\theta}_{i,j}^r = \boldsymbol{\theta}_{i,j-1}^r + \dot{\boldsymbol{\theta}}_{i,j}^r \quad \left| \begin{array}{l} \forall i \in \{1,\ldots,n\} \\ \forall j \in \{2,\ldots,T\} \end{array} \right.$$

Experiments

We conducted three experiments to demonstrate the validity of our model to teach the HOAP 2 robot simple manipulatory tasks. The tasks consisted in moving a chess piece on a chessboard (*"Chess Task"*), moving a bucket using two hands (*"Bucket Task"*), and bringing a piece of sugar to the mouth using either the left or right hand (*"Sugar Task"*), see Figure 3.13. Note that in these experiments, control affected only the eight DOFs of the arms, the one DOF of the torso, and the two binary commands to open and close the robot's hands. The robot was shown the task six to twelve times. Once trained, the robot was required to reproduce each task under different constraints, by placing the object at different locations in the robot's workspace. This procedure aimed at demonstrating the robustness of the system when the constraints were transposed to different locations within the robot's workspace.

The collected data for these three experiments present redundancies for the majority of the tasks. However, the degree and the type of redundancies can differ from one task to another. In these experiments, we were looking for a latent space onto which we project the original dataset to find an optimal representation for the given task. For example, an optimal latent space for a writing task is typically represented as a projection of the 3-dimensional original Cartesian position of the hand onto a 2-dimensional latent space defined by the writing surface, while a waving motion is typically represented as a combination of a single 1-dimensional cyclic pattern. Due to the small number of examples provided, we are only considering a latent space extracted through a linear combination of the data in the original data space. PCA (Principal Component Analysis)

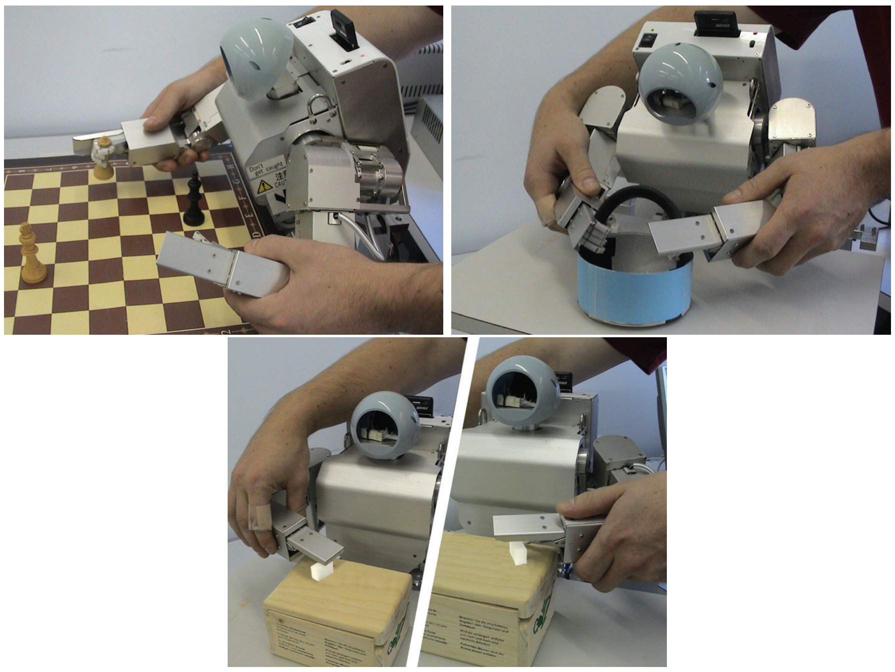

Figure 3.13: Teaching through kinesthetics for the 3 experiments conducted. *Left:* Grabbing and moving a chess piece using the torso. *Middle:* Grabbing and moving a bucket with two arms. *Right:* Grabbing a piece of sugar and bringing it to the mouth, using either the right or left hand.

has been used here as a pre-processing step to reduce the dimensionality of the data. We applied PCA separately to the set of variables $\{\dot{\boldsymbol{\theta}}^d, \dot{\boldsymbol{x}}^d, \dot{\boldsymbol{y}}^d\}$, in order to identify an underlying uncorrelated representation in each dataset. The relation between the original data $\{\dot{\boldsymbol{\theta}}^d, \dot{\boldsymbol{x}}^d, \dot{\boldsymbol{y}}^d\}$ and the projected data $\{\dot{\boldsymbol{\zeta}}_\theta^d, \dot{\boldsymbol{\zeta}}_x^d, \dot{\boldsymbol{\zeta}}_y^d\}$ is defined by the matrices $\{\boldsymbol{A}_\theta, \boldsymbol{A}_x, \boldsymbol{A}_y\}$:

$$\dot{\boldsymbol{\theta}}^d = \boldsymbol{A}_\theta \dot{\boldsymbol{\zeta}}_\theta^d \tag{3.48}$$

$$\dot{\boldsymbol{x}}^d = \boldsymbol{A}_x \dot{\boldsymbol{\zeta}}_x^d \tag{3.49}$$

$$\dot{\boldsymbol{y}}^d = \boldsymbol{A}_y \dot{\boldsymbol{\zeta}}_y^d \tag{3.50}$$

The Inverse Kinematics equation 3.47 is modified as follows:

$$\begin{aligned}\dot{\zeta}^\theta &= \left(\boldsymbol{W}^\theta + \boldsymbol{J}^T\boldsymbol{W}^x\boldsymbol{J} + (\boldsymbol{A}^z\boldsymbol{J})^T\boldsymbol{W}^y(\boldsymbol{A}^z\boldsymbol{J})\right)^{-1} \\ &\times \left(\boldsymbol{W}^\theta\,\boldsymbol{C}_1 + \boldsymbol{J}^T\boldsymbol{W}^x\,\boldsymbol{C}_2 + (\boldsymbol{A}^z\boldsymbol{J})^T\boldsymbol{W}^y\boldsymbol{C}_4\right)\end{aligned} \tag{3.51}$$

Where

Table 3.1: Number of parameters found automatically by the system and used in this experiment.

		Data space			Latent space	
		n	T	$(d-1)$	$(D-1)$	K
Chess Task	θ	7	100	9	4	5
	x	7	100	6	4	5
	y	7	100	6	4	4
	h	7	100	2	(2)	1
Bucket Task	θ	7	100	9	5	4
	x	7	100	6	4	7
	y	7	100	6	4	6
	h	7	100	2	(2)	1
Sugar Task - left	θ	4	100	9	2	5
	x	4	100	6	3	5
	y	4	100	6	3	6
	h	4	100	2	(2)	1
Sugar Task - right	θ	4	100	9	2	6
	x	4	100	6	3	6
	y	4	100	6	3	6
	h	4	100	2	(2)	1

$$\boldsymbol{C}_4 = (\boldsymbol{A}^y)^{-1}\dot{\boldsymbol{o}}_s + \boldsymbol{C}_3 \tag{3.52}$$

$$\boldsymbol{A}^z = (\boldsymbol{A}^y)^{-1}\boldsymbol{A}^x \tag{3.53}$$

and

$$\left.\begin{array}{lll} C_{1,i,k} & = & \hat{\zeta}_{\theta,i,k} - \zeta^r_{\theta,i,k-1} \\ C_{2,j,k} & = & \hat{\zeta}_{x,j,k} - \zeta^r_{x,j,k-1} \\ C_{3,j,k} & = & \hat{\zeta}_{y,j,k} - \zeta^r_{y,j,k-1} \end{array}\right| \begin{array}{l} \forall i \in \{1,\ldots,n\} \\ \forall j \in \{1,\ldots,m\} \\ \forall k \in \{2,\ldots,T\} \end{array}$$

For more details on the use of PCA as pre-processing step in this type systems and the techniques used to determine the optimal reduction of dimensionality, please refer to (Calinon & Billard, 2005; Calinon, Guenter, & Billard, 2007).

Results

Table 3.1 gives the dimensionality of the dataset, in the original data space and after projection by PCA, for the different tasks. In Figure 3.14, by observing the continuous description of the variation along the trajectories, we see that the object-hand relationships are highly constrained from time step 80, when grabbing and holding the bucket (i.e relative constraint), while the hands positions are highly constrained at the end of the motion. The essential features of the task have thus been extracted successfully in a continuous representation.

Figures 3.16-3.24 are using the same legend as presented in Figure 3.15. Figure 3.16, 3.17 and 3.18 show the reproduced trajectories for the *"Chess Task"*, depending on the initial position of the chess piece. Knowing that the right shoulder position is $\{x_1, x_2\} = \{100, -35\}$, we see on the map of Figure 3.17 that the best location to reproduce the motion is to initially place the chess piece in front of the right arm, see inset (1). In inset (2) and (3), the chess piece is placed initially to the left and to the right of the generalized hand path. We then observe the changes of the constraints along the trajectory during each experiment. The robot follows the demonstrated hand path as much as possible, still grasping the object.

Figures 3.19, 3.20 and 3.21 show the reproduced trajectories for the *"Bucket Task"*, depending on the initial position of the bucket. The optimal trajectory which satisfies the learned constraints follows globally the demonstrated hand path, still using the demonstrated object-hand trajectory when approaching the bucket.

Figures 3.22, 3.23 and 3.24 show the reproduced trajectories for the *"Sugar Task"*, depending on the initial position of the piece of sugar. In this task, a box is centered in front of the robot and two different gestures are taught to the robot. Firstly the robot is taught how to grasp a piece of sugar on the right side of the box with its right hand. Then, it is taught to grasp a piece of sugar on the left side of the box with its left hand. We compute an optimal controller for both the left and right arms, evaluate each controller with its respective metric, and select the best controller to reproduce the task. We see in the bottom-right inset (Figure 3.23) that the limit between the left/right part is clearly situated at $x_1 = 0$ (i.e. on the symmetry axis of the robot). Insets (3) and (4) of Figure 3.23 correspond to an initial position of the piece of sugar which differs from the initial positions used during the demonstrations.

3.5 Summary

In this chapter, we presented different methods, related to the resolution of the IK problem, used to control the HOAP's arm. The geometric IK was developed in order to solve the problem of finding the initial position with a algorithm such as the Moore-Penrose pseudo-inverse of the Jacobian which gives only a small displacement $\Delta\theta$. If we have an initial point defined only through Cartesian coordinates, by using the pseudo-inverse method, we have to interpol a trajectory from a known point to the initial point. Moreover, the position of the arm at the initial point will depend on the known starting position we use.

On the other hand, the geometric IK give us a position for a given angle α^e. If we want to generate a trajectory using this method, we have to take care to vary α^e smoothly along the trajectory in order to be able to follow the desired trajectory even in position where the possible α^e are restrained.

In the second method, the system uses all the data of the demonstrator in order to generate a trajectory. Here the IK algorithm used is related to the Moore-Penrose Pseudo-Inverse method. But for the trajectory generation, we combine the joint angle retrieved by GMR from the demonstration and the solution given by the IK applied on the Cartesian trajectory retrieved by GMR. The method has been validated on the HOAP 2 humanoid platform to perform different types of arm movements (reaching and stirring). However, the resulting strategy depends importantly on the hierarchy we fixed for the constraints (i.e. the value we set for the α_i factors in the cost function). Another important limitation of the system lies in its heavy computational requirement. At this stage, the computation is too slow to be run on-line. It takes about thirty seconds on a high speed processor (Pentium IV, 2.4 GHz) to compute a trajectory. While this is acceptable for robots working in protected areas, it would be impossible at this stage to run the system in an on-line fashion. The main problem comes from the gradient descent algorithm used to optimized the parameter γ that is used to balance the joint angles and the Cartesian data.

By extending the cost function and using Lagrange to find directly the optimum controller given the IK constraint and the constraint of the relation between the end-effector and a given object, we developed a more robust system that appears to be a generalization of the Damped Least Square Method applied on the Jacobian. This algorithm has the advantage of being robust to singularity.

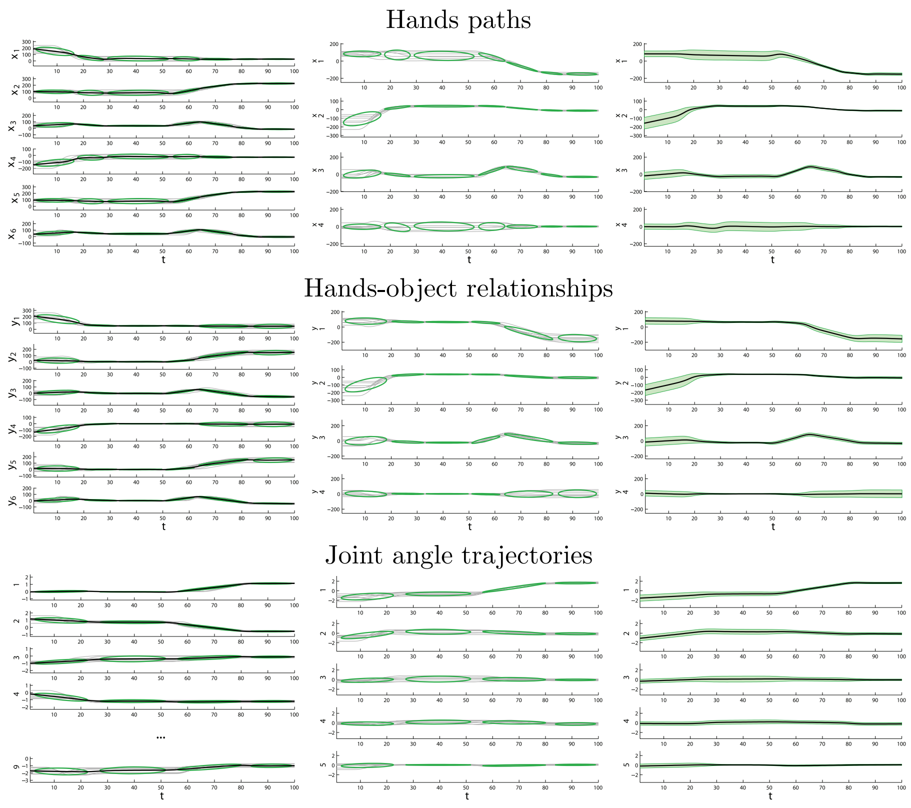

Figure 3.14: Probabilistic encoding for the *"Bucket Task"*. *Left:* Generalization of the hand paths x ($\{x_1, x_2, x_3\}$ and $\{x_4, x_5, x_6\}$ represent respectively the right/left hand paths). The six demonstrations are shown in the data space as dotted lines and the generalized signal reconstructed from the latent space is shown as a solid line. *Middle:* Reduction of dimensionality and temporal alignment. x are projected onto a latent space of lower dimensionality, and processed by Dynamical Time Warping (DTW). The resulting signals ξ^x are encoded in a GMM of 5 components, whose covariance matrix is represented by ellipses. We see that the positions are highly constrained at the end of the motion (represented as narrow ellipses for each dimension), since the ending-positions do not vary too much across the multiple demonstrations (the bucket is always placed at the same location after being grabbed). *Right:* Extraction of the constraints. Using the GMM representation, regression is used to retrieve a generalized version of the signals in the latent space $\hat{\xi}^x$ (black line), and to retrieve continuous covariance information along the signal $\hat{\Sigma}^x_s$ (envelope around the signal).

——	Hand path reproduced by the robot, reconstructed from θ'
- - -	Generalized hand path $\hat{x}$
······	Generalized hand path reconstructed from $\hat{\theta}$
○	Initial position of the object o_s
×	Beginning of the trajectories

Figure 3.15: Legend for Figure 3.16-3.24.

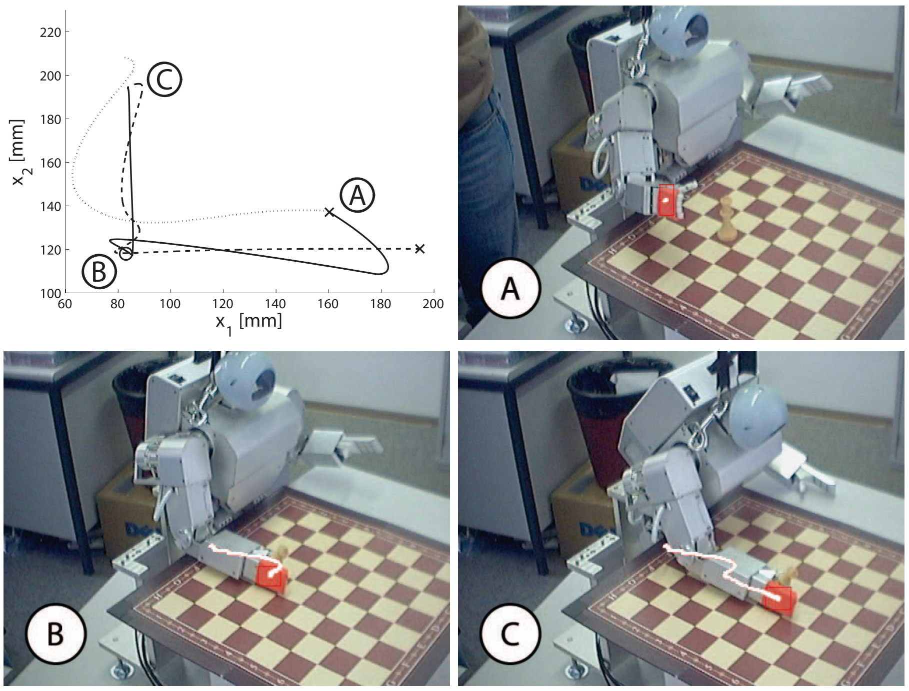

Figure 3.16: Decomposition of the *"Chess Task"*, when reproducing the task with an initial position of the object which is close to the generalized trajectories.

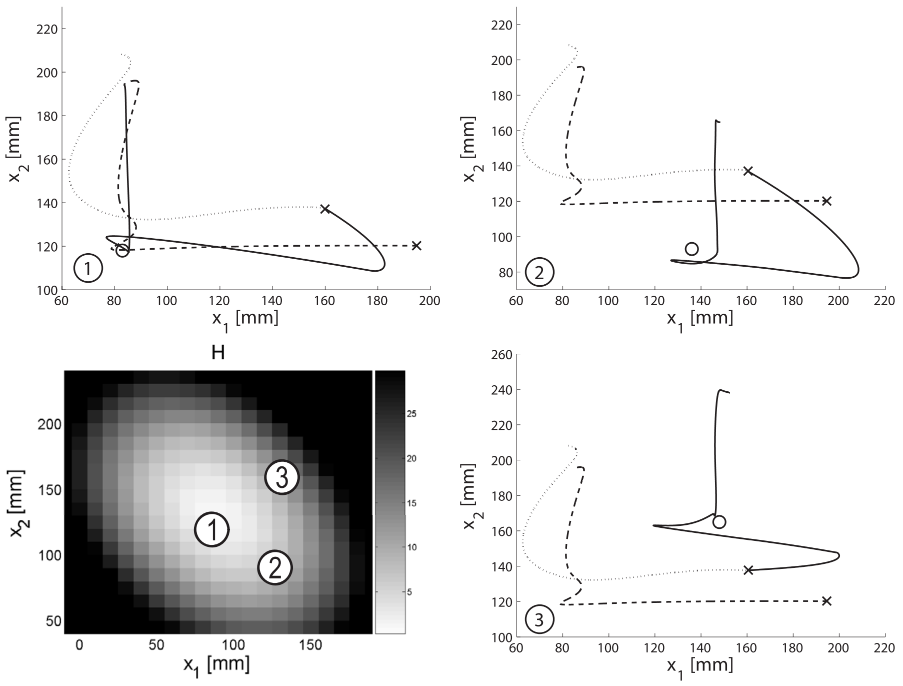

Figure 3.17: *Bottom left:* Mean values taken by the cost function H for the *"Chess Task"*, with a varying initial position of the chess piece on the board. *1,2,3:* Reproduction for the corresponding three locations on the map.

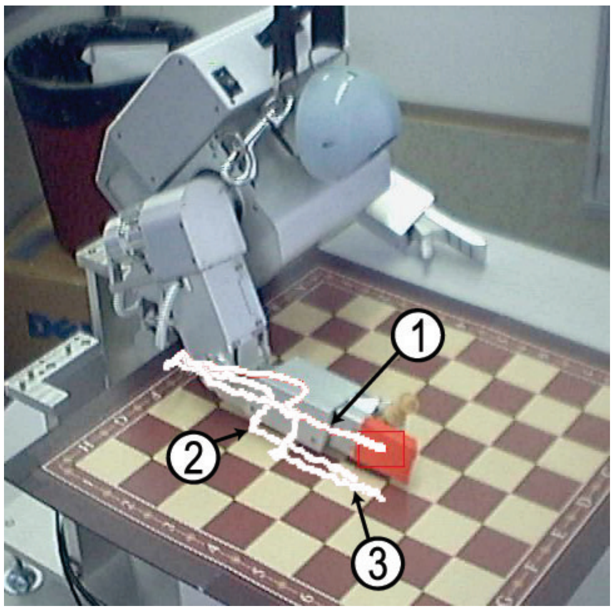

Figure 3.18: Reproduction of the *"Chess Task"*, for the 3 different locations presented in Figure 3.17. Here the hand paths have been tracked by a stereoscopic vision system.

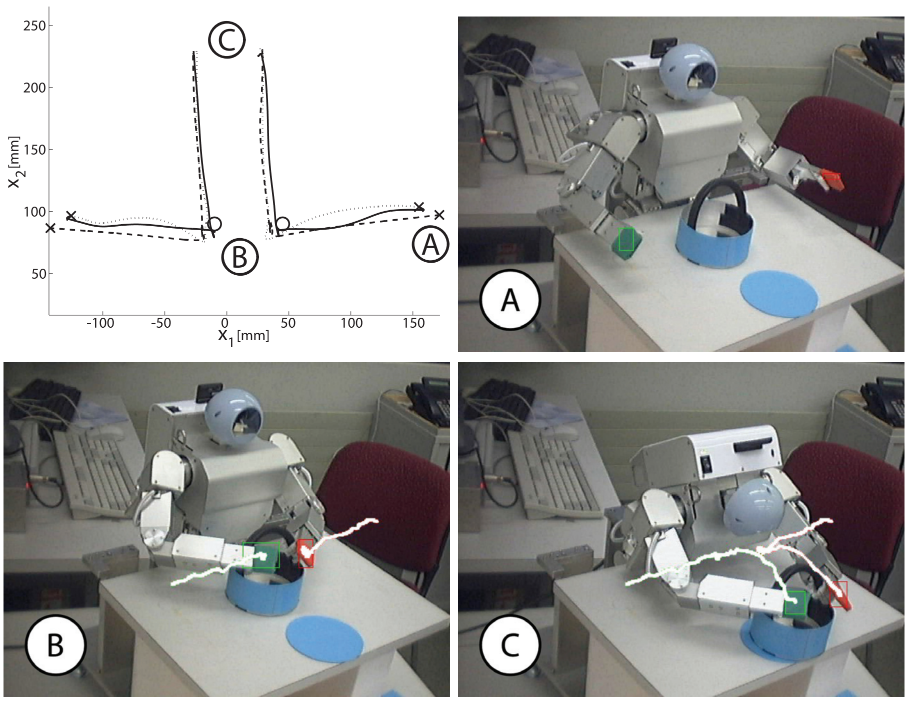

Figure 3.19: Decomposition of the *"Bucket Task"*, when reproducing the task with an initial position of the object close to the generalized trajectories.

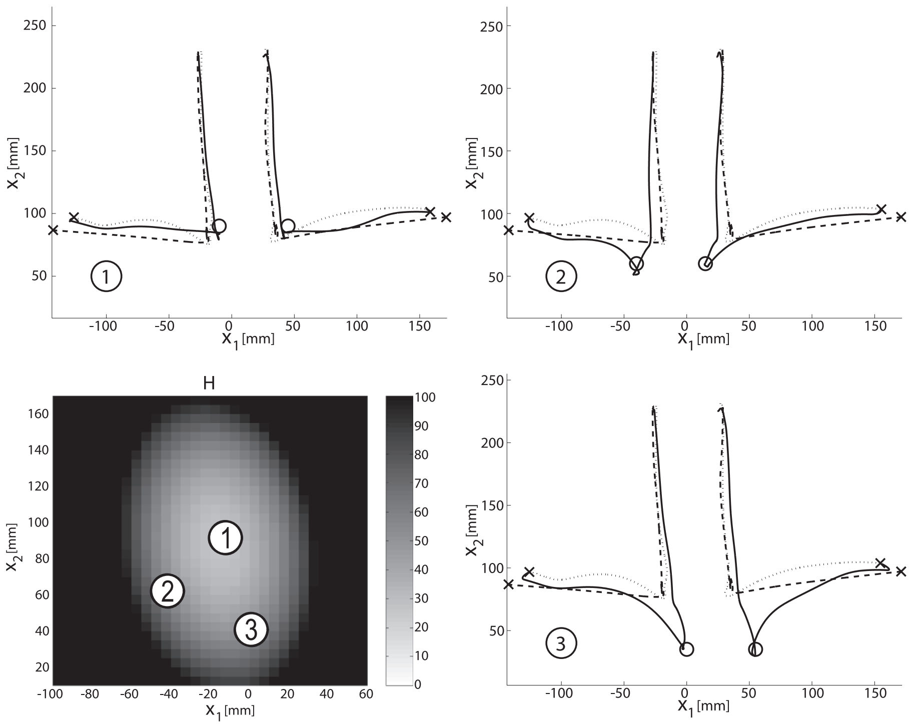

Figure 3.20: *Bottom left:* Mean values taken by the cost function H for the *"Bucket Task"* with a varying initial position of the bucket on the table. *1,2,3:* Reproduction for the corresponding three locations on the map.

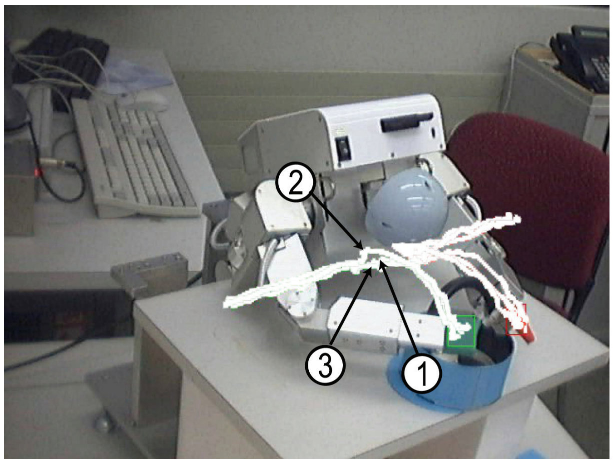

Figure 3.21: Reproduction of the *"Bucket Task"*, for the 3 different locations presented in Figure 3.20. The hands paths have been tracked by a stereoscopic vision system.

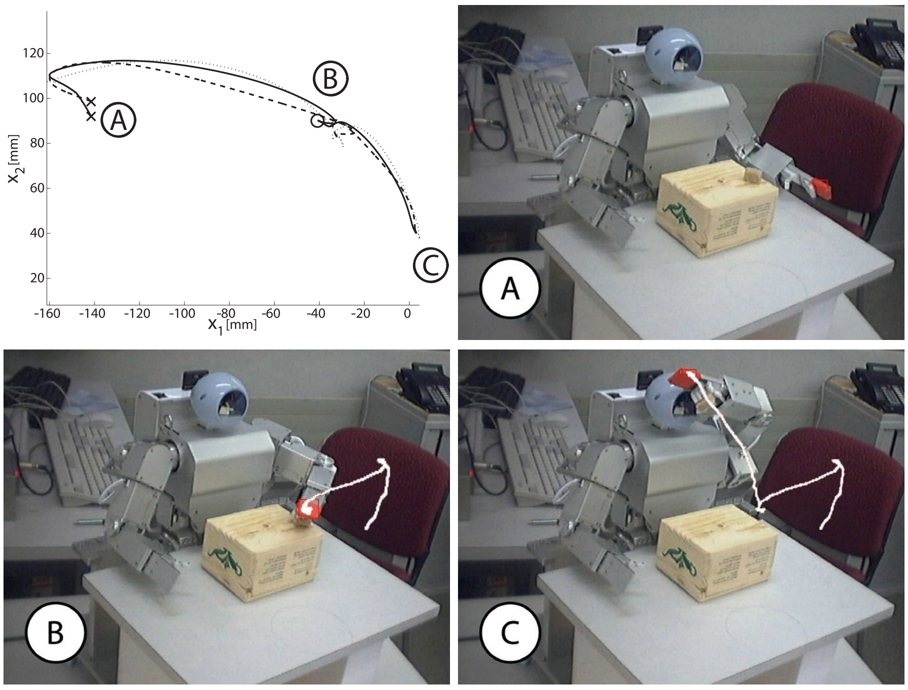

Figure 3.22: Decomposition of the *"Sugar Task"* for the left hand, when reproducing the task with an initial object position close to the generalized trajectories.

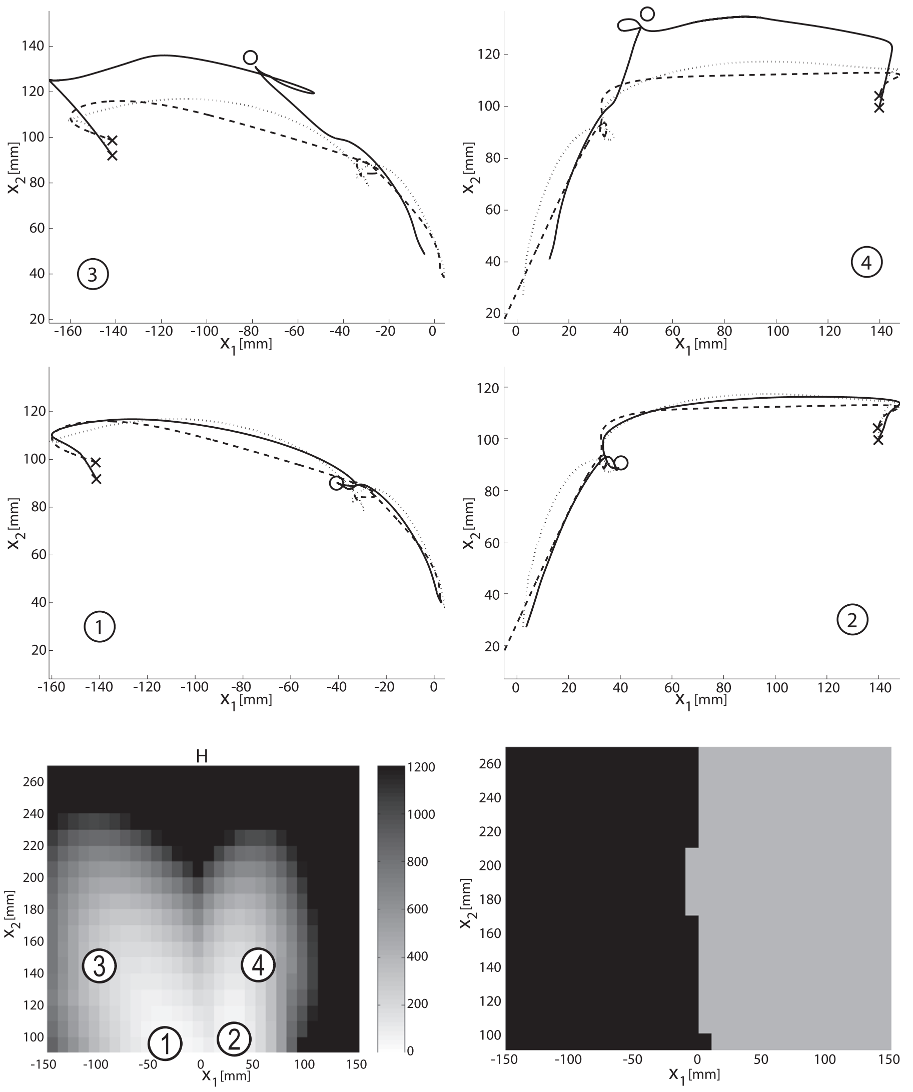

Figure 3.23: *Bottom left:* Mean values taken by the cost function H for the *"Sugar Task"*, with a variation in the initial position of the piece of sugar on the box. We distinguish two different areas for the left and right arm. The difference in size is due to the different variations used for the left and right part. *Bottom right:* Selection of a left/right arm controller depending on the value of H (black areas correspond to $H_{left} < H_{right}$) *1,2,3,4:* Reproduction for the corresponding four locations on the map.

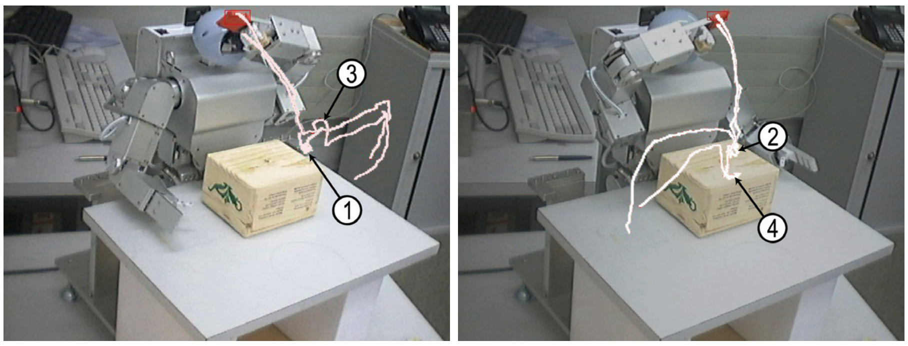

Figure 3.24: Reproduction of the *"Sugar Task"*, for the 4 different locations presented in Figure 3.23. The hands paths have been tracked by a stereoscopic vision system.

4 Combining Self-Learning and Imitation Learning

4.1 Outline

In the work presented in Chapter 3, we used different methods to combine the joint angles data and the Cartesian data extracted from a couple of demonstrations. The main drawback of this system is its poor robustness to perturbations. We are able to generate offline trajectories easily for working with static objects, but if the objects are moving too far from the demonstrations location or if an obstacle appears, the system is quickly unable to find a solution.

This Chapter will present a system developed in collaboration with two other PhD students at the Learning Algorithms and Systems Laboratory (LASA), Micha Hersch and Sylvain Calinon, in order to provide more robustness to the reproduction of tasks in a PbD framework. For that, we started with the hypothesis that each manipulation task can be decomposed in a series of reaching movement primitives. This hypothesis allows us to use a Dynamical System (DS) which consists in a spring and damper system with an attractor on the target of the reaching movement. The DS component of our system brings much more robustness, but does not resolve all the problems that can be encountered. My contribution was to add a trial-and-error learning component to the system that gives the robot the possibility to update its own model of the task in order to handle new situations or to optimize its reproduction taking into account its own body schema. A Reinforcement Learning (RL) algorithm has been implemented. In order to be able to compare the efficiency of the RL algorithm with different techniques, an algorithm from operations research has also been implemented. The global structure of the Chapter follows the chronological developments done on the system.

Section 4.2 gives an overview of the developed algorithm and its different modules. Data encoding in GMM and Dynamical System modelling developed respectively by Sylvain Calinon and Micha Hersch are briefly presented in Section 4.3 and 4.4.

In Section 4.5, I will especially focus on my contribution and present in details the Natural Actor Critic (NAC) algorithm that allows the robot to relearn and optimize its own model of the task through Reinforcement Learning. Different Subsections present the different parts of the algorithm. Section 4.5.1 presents the policy chosen, Section 4.5.2 presents the Reward Function used to evaluate the reproductions, Section 4.5.3 presents the algorithm itself, Section

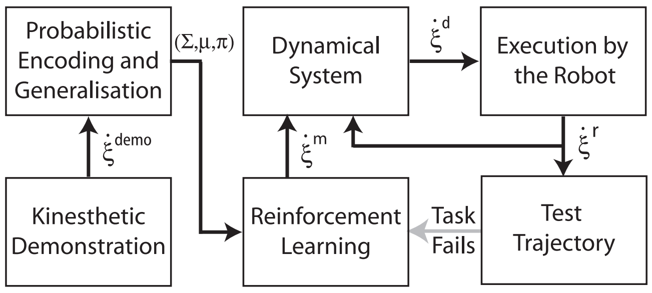

Figure 4.1: Schematic overview of the complete system. The speed profiles $\dot{\boldsymbol{\xi}}^{demo}$ provided by the set of demonstrations are fed into a learning system that trains a GMM model for the data. The value π, μ, Σ are used in a Reinforcement Learning module for generating a trajectory $\boldsymbol{\xi}^{m}$ that modulates a DS with attractor $\boldsymbol{\xi}^{g}$ and generate an output speed $\dot{\boldsymbol{\xi}}^{d}$ executed by the robot. The effective trajectory measured on the robot is given by $\dot{\boldsymbol{\xi}}^{r}$ and is tested using a reward function to determine if the task have failed or not. In our experiments, when the reward is higher than a certain threshold, the task is considered as fulfilled. If the task fails, we use a reinforcement learning algorithm to update the GMM and produce a new trajectory $\dot{\boldsymbol{\xi}}^{m}$ in order to fulfill the task.

4.5.4 presents the different solutions used to update the parameters we want to optimize and finally, Section 4.5.5 presents the different solutions that can be used in order to detect the convergence of the RL algorithm.

Section 4.6 is dedicated to the presentation of the results obtained using the NAC algorithm. A comparison between the first version of the NAC and a second improved version is presented here.

Section 4.7 presents a different algorithm used for the trial-and-error learning module. The extended Taguchi method is an algorithm from Operations Research (OR) that can be used here in order to optimize the reproduction of the learned task. The algorithm is presented in Section 4.7.1 while the results are presented in Section 4.8. A comparison between the performance of the NAC and the extended Taguchi method is also presented in this Section.

4.2 PbD Framework: System Overview

Let $\boldsymbol{\xi(t)} \in \mathbb{R}^s$ describe the complete state of the robot at each time step.

The information flow of the algorithm is illustrated in Figure 4.1. After having been exposed to several demonstrations $\dot{\boldsymbol{\xi}}^{d}(t)$ of the task, the algorithm extracts a generalized form of the original demonstration $\dot{\boldsymbol{\xi}}^{m}(t)$ using a probabilistic model. The generalized trajectory is then used to modulate a DS which produces a trajectory used to reach the target $\boldsymbol{\xi}^{g}$.

When facing important perturbations, such as an obstacle blocking the arm of the robot, it may happen that the DS alone does not find a satisfying solution to reach the goal. To avoid that type of problem, we have implemented a RL module which allows the robot to learn how to avoid the obstacles or other perturbations properly. The RL module acts directly on the mean of the GMM in order to update the modulation of the DS. This way, the convergence properties of the DS are preserved. Note that the system described below does not make any assumption on the type of data and, thus, ξ could be composed of other variables, such as, for instance, the position of the robot's end effector or the data projected in a latent space as done in (Calinon et al., 2007).

4.3 Probabilistic Encoding and Generalization

This part of the work has already been presented in Section 3.4.2. We use a GMM in order to encode the demonstrated data and a GMR in order to retrieve the expected trajectory $\dot{\boldsymbol{\xi}}^m$ learned from the demonstration. $\dot{\boldsymbol{\xi}}^m$ is then used to modulate the DS.

4.4 Dynamical System

As introduced at the beginning of this chapter, the GMR has the inconvenient that the retrieved trajectory is fixed and does not handle the perturbations. An effective solution to generate trajectories that are robust to perturbations is to use a Dynamical System (DS). DS are bio-inspired control system that have already been successfully used in robotics (Haken, Kelso, Fuchs, & Pandya, 1990; Schoner, Dose, & Engels, 1995; Iossifidis & Schoner, 2004; Righetti & Ijspeert, 2006). Here, we take inspiration on the Vector Integration To Endpoint (VITE) model introduced in (Bullock & Grossberg, 1988). This model can be associated to a proportional derivative controller and allows the robot to bring its arm smoothly to the target, following a straight trajectory (in the absence of perturbations).

$$\ddot{\boldsymbol{\xi}}(t) = \alpha(-\dot{\boldsymbol{\xi}}(t) + \beta(\boldsymbol{\xi}^g - \boldsymbol{\xi}^r(t))) \tag{4.1}$$

$$\dot{\boldsymbol{\xi}}(t) = \dot{\boldsymbol{\xi}}^r(t) + \ddot{\boldsymbol{\xi}}(t) \tag{4.2}$$

The constants $\alpha, \beta \in \mathbb{R}_{[0,1]}$ are control parameters, $\dot{\boldsymbol{\xi}}$ and $\ddot{\boldsymbol{\xi}}$ are the current speed and acceleration of the DS, $\boldsymbol{\xi}^r$ is the current position of the robot and $\boldsymbol{\xi}^g$ represents the target position (the goal). α and β are similar to the proportional gain and derivative term in a PD controller. If $t \to \infty$, the system will then converge to the target.

4.4.1 Modulation of the Dynamical System

Because there are some tasks for which the robot must follow a specific trajectory to reach the goal (for example to put an object into a box, you have to come from above) while being robust to perturbation, we will combine the GMR

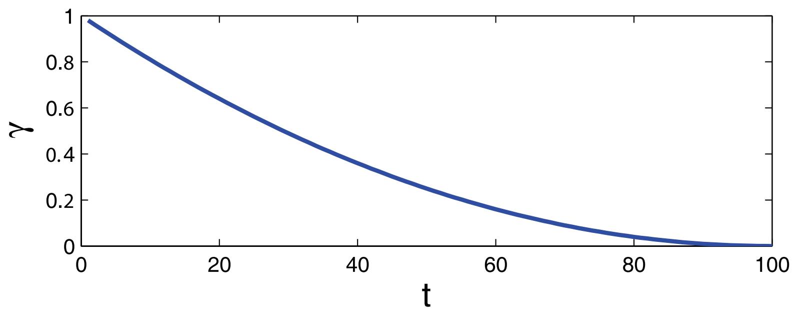

Figure 4.2: Evolution of the parameter γ along time.

and the DS in order to take the advantages of both methods. To realize that, we modulate the DS with the output of the GMR. In other words, we give a weighted average between the output of the DS and a desired speed given by the GMR to the robot.

$$\dot{\xi}^d(t) = (1 - \gamma(t))\dot{\xi}(t) + \gamma(t)\dot{\xi}^m(t) \tag{4.3}$$

where $\dot{\boldsymbol{\xi}}^m$ is the desired speed used to modulate the DS and $\dot{\xi}^d$ is the speed command for the robot.

One inconvenience of the system is that the modulation speed $\dot{\xi}^m(t)$ is given for a fixed number of time steps T while the DS speed $\dot{\xi}(t)$ brings the arm to the goal in a unknown number of time steps T^g.

In a first version, the time T^g was set equal to T and the function $\gamma(t)$ was define by the function:

$$\gamma(t) = ((T - t)/T)^2 \tag{4.4}$$

Where $\gamma(t) \in \mathbb{R}_{[0,1]}$. Equation 4.4 ensures a smooth decay to zero for the modulation term $\dot{\xi}^m(t)$ (see Figure 4.2) and then ensure the convergence of the system to the target.

The two main drawbacks of this system are: (1) The fixed time to reach the goal. If the starting position is too far from the target compared to the demonstration, the system will not be able to ensure the convergence of the end-effector to the target. (2) The fixed function $\gamma(t)$. The system is entirely constrained by this function that is fixed by hand. Ideally, we would prefer to have a function that allows tuning the modulation of the DS.

In order to palliate at these two problems, a second version of the modulation has been developed. The system is now allowed to go beyond time T. In other word, now, in the case where $T^g > T$, $\dot{\xi}^d(t) = \dot{\xi}(t)$, we have thus the contribution

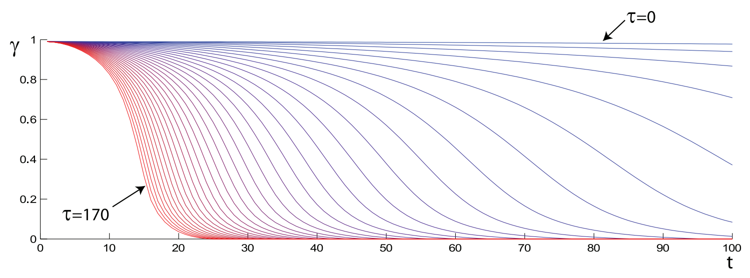

Figure 4.3: This graphic shows the evolution of the function γ when τ is varying between 0 and 170.

of the DS alone for $T^g > T$. In order to ensure a smooth convergence of the system on the target, we need to force the second term to zero at time T and have a function $\gamma(t)$ that can variate in function of the system needs. To realize this, we modulate $\dot{\boldsymbol{\xi}}^m$ with the new function $\gamma(t)$ which is defined by:

$$\begin{aligned}\ddot{\gamma}(t) &= \alpha_\gamma(-\dot{\gamma}(t) + \beta_\gamma(0 - \gamma(t))) &(4.5)\\ \dot{\gamma}(t) &= \dot{\gamma}(t-1) + \ddot{\gamma}(t) &(4.6)\\ \gamma(t) &= \gamma(t-1) + \dot{\gamma}(t) &(4.7)\\ \gamma(t=0) &= 1 &(4.8)\end{aligned}$$

Where $\alpha_\gamma = \tau(\epsilon + \gamma(t)^2)$ and $\beta_\gamma = \frac{\tau}{5}$. In α and β, $\epsilon = 0.05$ is a constant used to avoid a proportional gain $\alpha_\gamma = 0$ and τ is a parameter that allows us to control the shape of the function $\gamma(t)$. In fact, this is a variation of the DS, we introduced in 4.4. By varying the parameter τ, we act on the proportional and derivative term of the system and thus act on the time needed to the function $\gamma(t)$ to reach a null value.

With this system we can choose during which proportion of the time we want to have a contribution of the desired speed given by the GMR. Figure 4.3 represents the function $\gamma(t)$ for different values of τ. We also see in Figure 4.3 that if we have a small τ, γ will not reach 0 at the end of the trajectory. In this case, the modulation still perturbs the DS at time T. The consequence is a discontinuity in the speed profile when passing from time T to time $T+1$ as shown by the blue trajectories in Figure 4.4. In the other extreme, when τ becomes too big, the function $\gamma(t)$ becomes unstable. For these two reasons, we have to bound the parameter $\tau \in \{35; 170\}$.

The default value used for τ is 35, which is the value that allows the longest influence of the desired speed while ensuring that γ is equal to 0 for $t = T$. But we have now the possibility to include τ in the parameters learned by the RL module.

For security reason, we have implemented a system that avoids reaching the joint limits of the robot HOAP3. For that, we use a parameter L in order to

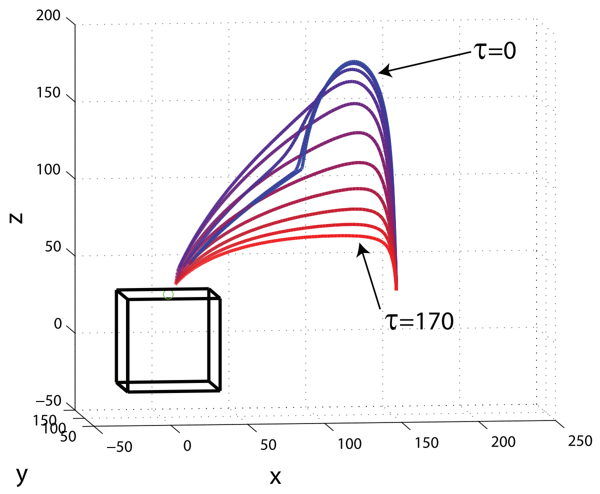

Figure 4.4: This graphic shows the different trajectories generated by the combination of the DS and the GMR for different values of τ. τ is varying between 0 (Trajectory in blue) and 170 (Trajectory in red) as in Figure 4.3.

decrease the speed amplitude when we approach of the joint limits. The position update is done as follow:

$$\xi^d(t+1) = \xi^d(t) + L\dot{\xi}^d(t)\Delta t \tag{4.9}$$

where

$$L = \begin{cases} F(\xi^d_t) & \text{if } \xi^d(t) \in [\xi^d_{min}, \xi^d_{max}] \\ 0 & \text{if } \xi^d(t) \notin [\xi^d_{min}, \xi^d_{max}] \end{cases} \tag{4.10}$$

and

$$F(\xi^d) = 1- \mid arctanh(1.52 * (\frac{\xi^d(t) - \xi_{mean}}{\xi^d_{max} - \xi^d_{min}}))^{10} \mid \tag{4.11}$$

The shape of the L parameter within the joint limits $[\xi^d_{min}, \xi^d_{max}]$ is shown in Figure 4.5.

4.5 Reinforcement Learning: Natural Actor Critic (NAC)

In order to allow the robot finding a new solution even if the system fails to reproduce the learned task, we have added a Reinforcement Learning module to our system. This module acts on the dynamical system through the modulation

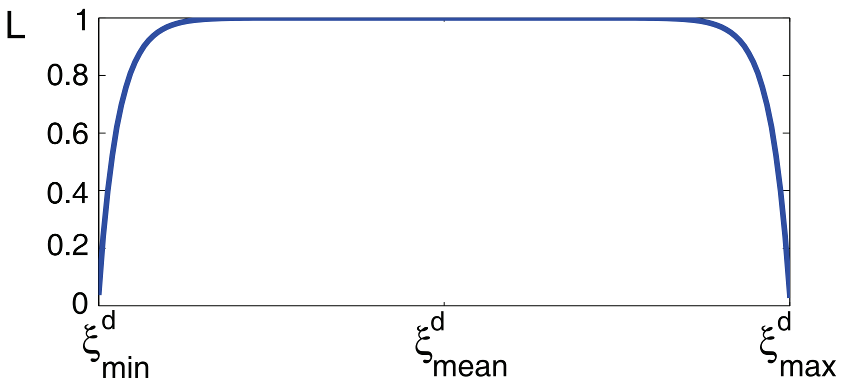

Figure 4.5: Evolution of the parameter L in function of ξ^d.

variable $\dot{\xi}^m(t)$ by optimizing the means $\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}$ of Equation 3.18.

By default, for the first trial $q = 1$ (where q denotes the trial number), the means $\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}$ are given by the GMM model. Thus, we avoid interferences of the reinforcement learning module if the dynamical system is sufficient to ensure the reproduction of the task. If the system fails to reach the target by using the initial GMM, the RL is applied to learn a new set of means $\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}$ for the GMM. The whole learning process is done in simulation even though we will implement the system on the robot. instead of making 100 or 200 trials with the robot in a real environment, it is much more efficient to simulate the process as it allows finding a solution within a few seconds. The advantage of Reinforcement Learning techniques over other direct path planning techniques is that it allows the robot to handle different types of situations. The robot is able to learn by exploring the solution space and discovering new solutions.

The algorithm we have used for the RL module of our system is based on the episodic natural actor-critic (NAC) algorithm presented in (Peters et al., 2003, 2005). The NAC is a variant of the actor-critic method which is classified as a temporal-difference learning method. This algorithm combine the advantage of dynamic programming techniques and Monte Carlo method. Moreover, the NAC is a efficient algorithm that well represents the last developement in the RL field in robotics. In the critic part of the NAC, the policy is evaluated by approximating the state-value function. For the approximation, an adaptation of a $LSTD(\lambda)$ (Least Square Temporal Difference) algorithm (Nedic & Bertsekas, 2002; Boyan, 2002) called $LSTD-Q(\lambda)$ (Peters et al., 2003) is used. In the actor part, the policy is improved by using the "natural" gradient descent (Amari, 2000) which is a steepest gradient descent algorithm with respect to the Fisher information matrix.

4.5.1 The Policy

The first step is to define the policy of our system. In order to explore the space of the parameters, we introduce a stochastic policy defined by a Gaussian control policy:

$$\rho(\dot{\boldsymbol{\xi}}^r, \dot{\boldsymbol{\xi}}^m, \boldsymbol{\Sigma}_{\dot{\xi}}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_{\dot{\xi}}|}} e^{(-\frac{1}{2}(\dot{\boldsymbol{\xi}}^r - \dot{\boldsymbol{\xi}}^m)^T \boldsymbol{\Sigma}_{\dot{\xi}}^{-1} (\dot{\boldsymbol{\xi}}^r - \dot{\boldsymbol{\xi}}^m))} \tag{4.12}$$

where $\dot{\boldsymbol{\xi}}^m$ is the modulation speed of the dynamical system retrieved by GMR (see Equation 3.22), $\boldsymbol{\Sigma}_{\dot{\xi}}$ is the covariance matrix (see Equation 3.24) of the Gaussian control policy and $\dot{\boldsymbol{\xi}}^r$ is the noisy speed profile generated by $\rho(\dot{\boldsymbol{\xi}}^r, \dot{\boldsymbol{\xi}}^m, \boldsymbol{\Sigma}_{\dot{\xi}})$ and used to explore the parameters space. We consider here that the demonstrations performed by the user are sufficiently informative to allow the robot reproducing the task in standard conditions. We thus choose to keep the covariance matrix $\boldsymbol{\Sigma}_{\dot{\xi}}$ in order to respect the constraints taught by the demonstrator during the exploration process of the RL module. We then use $\dot{\boldsymbol{\xi}}^r$ to modulate the DS instead of $\dot{\boldsymbol{\xi}}^m$. Thus we obtain a noisy trajectory that we evaluate using the reward function r_q.

4.5.2 The Reward Function

The first version of the reward function r_q was very simple and contained only two terms:

$$r_q(\dot{\boldsymbol{\xi}}^r, \boldsymbol{\xi}^r) = \sum_{t=0}^{T} -c_1 |\dot{\boldsymbol{\xi}}^r_t - \dot{\boldsymbol{\xi}}^m_{t,q=1}| - c_2 |\boldsymbol{\xi}^r_T - \boldsymbol{\xi}^g|, \tag{4.13}$$

where $c_1 > 0, c_2 > 0 \in \mathbb{R}$ are weighting constants, $\boldsymbol{\xi}^r$ is the simulated noisy command used to explore the solution space (see Section 4.5.1), $\boldsymbol{\xi}^m_{t,q=1}$ is the modulation speed of the first trial (see Equation (3.24)) and $\boldsymbol{\xi}^g$ is the target position. Thus the reward function is determined by a term that represents the similarity between the current trajectory and the original modulation given by the GMM and a term that represents the distance between the target and the last point of the tested trajectory.

The effect of the first term on the reward function is that r_q tends to a maximum $r_q < 0$ in most of the cases. Because the trajectory $\boldsymbol{\xi}^m_{t,q=1}$ is the expected output of the GMM and does not depend on the target, it does rarely reach the target. Thus, after the learning, if the target is reached, there is still a difference between the current trajectory $\boldsymbol{\xi}^m_{t,q}$ and the initial trajectory $\boldsymbol{\xi}^m_{t,q=1}$ due to the deformation needed to reach the target.

The effect of the second term on the reward function is that r_q tends to a maximum when $\boldsymbol{\xi}^m_{t,q}$ reaches the target. The obstacle is implicitly represented in this term. During the learning, when the arm is blocked by an obstacle, the last point of the trajectory $\boldsymbol{\xi}^s_T$ does not reach the target, thus the second term of r_q does not reach zero with an obstacle on the trajectory.

The point here is that the reward function depend only on the noisy modulation trajectory. It does not depend on the combination of the DS and the modulation trajectory. The reason to evaluate only the modulation was to avoid the instability of the system due to the DS component as it will be presented in Section 4.5.4. But there are some drawbacks with this solution. The main problem is that by evaluating only the modulation, we have no information on the trajectory generated by combining the DS and $\dot{\boldsymbol{\xi}}^r$. It is then possible to find some cases where the modulation trajectory pass the obstacle but not the final trajectory.

To new version of the reward function depends now on $\boldsymbol{\xi}^s$ that is the simulated states of the robot, in other word, the combination of the DS and the noisy speed profile $\dot{\boldsymbol{\xi}}^r$. The new reward function r_q is defined by:

$$r_q(\dot{\boldsymbol{\xi}}^s, \boldsymbol{\xi}^s) = \big(\sum_{t=0}^{T} -c_1|\dot{\boldsymbol{\xi}}^s_t - \dot{\boldsymbol{\xi}}^m_{t,q=1}| - c_2|\dot{\boldsymbol{\xi}}^s_t - \dot{\boldsymbol{\xi}}^s_{t-1}|\big) \tag{4.14}$$

$$-c_3|\boldsymbol{\xi}^s_T - \boldsymbol{\xi}^g| - c_4 T^g \tag{4.15}$$

where $c_1, c_2, c_3, c_4 \in \mathbb{R}$ are weighting constants, $\boldsymbol{\xi}^s$ is the simulated state of the robot, $\dot{\boldsymbol{\xi}}^m_{t,q=1}$ is the modulation speed of the first trial (see Equation 3.24), $\boldsymbol{\xi}^g$ is the target position and T^g is the time needed to reach the goal.

Thus, as before, we retrieve the two terms that represent the similarity between the current trajectory and the original modulation given by the GMM and the distance between the goal and the last point of the tested trajectory (respectively the first and third terms of the new reward function). The second term is new and ensures a smooth trajectory by minimizing acceleration along the path. Finally the last term comes from the modification done in the DS modulation system. As the time T^g needed to reach the target varies and depends now on the DS, we can use it in the reward function to penalize long trajectories in terms of time.

4.5.3 the NAC

This exploration process allows us approximating the gradient of the expected returns. We then use this approximated gradient to update the means $\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}$ of the GMM and thus update the policy at the same time. By optimizing $\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}$, we are able to generate new trajectories $\dot{\boldsymbol{\xi}}^m_q$ that will help the dynamical system to avoid the obstacle smoothly (see Figure 4.6). The exploration process has to be conducted in simulation in order to avoid breaking the robot with the noisy trajectories. Once the gradient has been approximated and the means updated, we generate a smooth trajectory that can be tested on the robot using the new

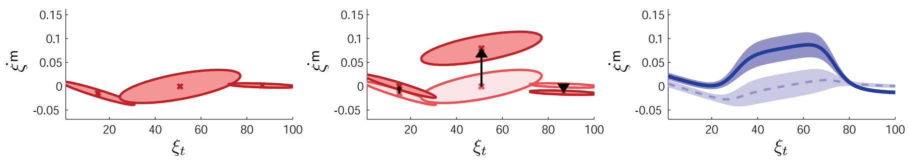

Figure 4.6: This graphic shows how the RL module acts on the GMM and then on the expected output given by GMR. This is a example done on one dimension of the arm joint angles of the HOAP3

$\dot{\boldsymbol{\xi}}^m_q$.

The following algorithm is applied separately on each $\mu_{k,\dot{\boldsymbol{\xi}}}$. For ease of reading, we will thus omit the subscript $_k$ in the following description. In the "critic" part of the algorithm, for the policy evaluation, like in most RL algorithms, we evaluate the expected reward of a command $\dot{\boldsymbol{\xi}}^r$ for a state $\dot{\boldsymbol{\xi}}^s$. That is generally done by the evaluation of the Action-Value function defined as:

$$Q^\rho(\dot{\boldsymbol{\xi}}^r, \dot{\boldsymbol{\xi}}^s) = E\{\sum_{t=0}^{\infty} \gamma^t r_t | \dot{\boldsymbol{\xi}}^r, \dot{\boldsymbol{\xi}}^s\} \tag{4.16}$$

where $\gamma \in \mathbb{R}_{[0,1]}$ is a discount factor and r_t is the immediate reward. Note that this model is for a non episodic task. In the same way, we can define the Value function as:

$$V^\rho(\dot{\boldsymbol{\xi}}^s) = E\{\sum_{t=0}^{\infty} \gamma^t r_t | \dot{\boldsymbol{\xi}}^s\} \tag{4.17}$$

As shown in (Peters et al., 2003), these two expressions allow us to define an advantage function:

$$A^\rho(\dot{\boldsymbol{\xi}}^r, \dot{\boldsymbol{\xi}}^s) = Q^\rho(\dot{\boldsymbol{\xi}}^r, \dot{\boldsymbol{\xi}}^s) - V^\rho(\dot{\boldsymbol{\xi}}^s), \tag{4.18}$$

where $A^\rho(\dot{\boldsymbol{\xi}}^r, \dot{\boldsymbol{\xi}}^s)$ represents the advantage of action $\dot{\boldsymbol{\xi}}^r$ over the state $\dot{\boldsymbol{\xi}}^s$. To adapt the NAC to an episodic task, we formulate the discounted sum of the advantage function along one trial as:

$$\begin{aligned}\sum_{t=0}^{T} \gamma^t A^\rho(\dot{\boldsymbol{\xi}}^r, \dot{\boldsymbol{\xi}}^s) &= r_q(\dot{\boldsymbol{\xi}}^r, \boldsymbol{\xi}^r) + \sum_{t=0}^{T} \gamma^{t+1} V^\rho(\dot{\boldsymbol{\xi}}^s_{t+1}) - V^\rho(\dot{\boldsymbol{\xi}}^s_t) &(4.19)\\ &= r_q(\dot{\boldsymbol{\xi}}^r, \boldsymbol{\xi}^r) + \gamma^{T+1} V^\rho(\dot{\boldsymbol{\xi}}^s_{T+1}) - V^\rho(\dot{\boldsymbol{\xi}}^s_0) &(4.20)\end{aligned}$$

Note that for a episodic task, $\gamma^{T+1} V^\rho(\dot{\boldsymbol{\xi}}^s_{t+1}) = 0$. In order to have an efficient algorithm we use a linear approximation function for $V^\rho(\dot{\boldsymbol{\xi}}^s)$:

$$V^\rho(\dot{\boldsymbol{\xi}}^s) \approx \boldsymbol{\phi}(\dot{\boldsymbol{\xi}}^s)'\boldsymbol{v}_q, \tag{4.21}$$

where $\boldsymbol{v}_q \in \mathbb{R}^T$ is a vector of weights and $\phi(\dot{\boldsymbol{\xi}}^s)' \in \mathbb{R}^T$ is the transpose of a feature vector. In our case, since only $V^\rho(\dot{\boldsymbol{\xi}}^s_0)$ in Equation (4.20) has to be evaluated, we need only one basis function to approximate $V^\rho(\dot{\boldsymbol{\xi}}^s)$. Thus, we can arbitrarily set $\phi(\dot{\boldsymbol{\xi}}^s) = 1$ and approximate the value function at time $t = 0$ by using only the weight v_q.

Following (Peters et al., 2003), we then approximate the advantage function with:

$$A^\rho(\dot{\boldsymbol{\xi}}^r, \dot{\boldsymbol{\xi}}^s) \approx \nabla_\pi ln\rho(\dot{\boldsymbol{\xi}}^r, \dot{\boldsymbol{\xi}}^m, \boldsymbol{\Sigma}_{\dot{\xi}})'\boldsymbol{w}_q \tag{4.22}$$

where $\boldsymbol{w}_q$ is a vector of weight that is equal to the approximation of the "natural" gradient of the expected return (see (Peters et al., 2003)). By using this approximation, we are then able to evaluate of the reward on one trial:

$$\sum_{t=0}^{T} \gamma^t \nabla_\pi ln\rho(\dot{\boldsymbol{\xi}}^r, \dot{\boldsymbol{\xi}}^m, \boldsymbol{\Sigma}_{\dot{\xi}})'\boldsymbol{w}_q + \boldsymbol{v}_q \approx r_q \tag{4.23}$$

Where $\boldsymbol{v}_q$ represents the approximation of the value function at the initial state of each trial. To find the natural gradient $\boldsymbol{w}$, we will use a $LSTD - Q(\lambda)$. For that, we have to define a new basis function:

$$\widehat{\phi} = \sum_{t=0}^{N} [1, \gamma^t \nabla_\Theta ln\rho(\dot{\boldsymbol{\xi}}^r, \dot{\boldsymbol{\xi}}^m, \boldsymbol{\Sigma}_{\dot{\xi}})']' \tag{4.24}$$

With this new basis function, we can rewrite Equation 4.23 in a vectorial form:

$$\widehat{\phi} \cdot [\boldsymbol{v}_q, \boldsymbol{w}_q']' \approx r_q \tag{4.25}$$

For each trial q, we compute the basis function $\widehat{\phi}$ and the immediate reward r_q to update the sufficient statistics $\boldsymbol{A}_q$ and $\boldsymbol{b}_q$ (see (Nedic & Bertsekas, 2002)).

$$\begin{aligned} \boldsymbol{A}_q &= \boldsymbol{A}_{q-1} + \widehat{\phi}_q \widehat{\phi}_q' && (4.26) \\ \boldsymbol{b}_q &= \boldsymbol{b}_{q-1} + \widehat{\phi}_q r_q && (4.27) \end{aligned}$$

These sufficient statistics are then used to update the critic parameters v_q

and $\boldsymbol{w}_q$:

$$[\boldsymbol{v}_q, \boldsymbol{w}_q^T]' = \boldsymbol{A}_q^{-1}\boldsymbol{b}_q \tag{4.28}$$

After multiple trials, $\boldsymbol{w_q}$ converges to the natural gradient of the expected return. Thus, once w has converged over a window h, i.e. $\forall\tau \in [0, ..., h]$, the angle between w_{e+1} and $w_{e-\tau}) \leq \epsilon$, we are able to update the parameters $\boldsymbol{\mu}_{\dot{\boldsymbol{\xi}}}$ in order to optimize the expected return.

4.5.4 Parameters Update

In the first version, the update was done using a classical gradient descent technique that consists in updating the means of the GMM by using a value proportional to the gradient of the expected return $\boldsymbol{w}_q$:

$$\boldsymbol{\mu}_{q,\dot{\boldsymbol{\xi}}} = \boldsymbol{\mu}_{q-1,\dot{\boldsymbol{\xi}}} + \alpha_{learn}\boldsymbol{w}_q \tag{4.29}$$

where $\alpha_{learn} \in \mathbb{R}_{[0,1]}$ is the learning rate.

This system has the advantage to be fast, but if the gradient is too steep, the update term might explode. If the update term explodes, the parameters $\boldsymbol{\mu}_{q,\dot{\boldsymbol{\xi}}}$ become inconsistent and the system does not find a solution. Because of the DS, when the system is just at the limit to pass the obstacle, some of the noisy trajectories used to explore the space around the current solution pass the obstacle and reach the goal while others get stuck on the obstacle. At this point, the gradient becomes very steep and it is difficult to tune the parameters to keep the system stable.

As explained in the Section 4.5.2, in the first version this was not a problem as the reward was used to evaluate the noisy trajectory produced by the policy and not the trajectory produced by the modulated DS. In the new version, as the evaluated trajectory is produced by the modulated DS, we need a new solution to avoid instability problems. We use then the Rprop solution (Riedmiller, 1994). This solution consists in updating the means $\boldsymbol{\mu}_{q,\dot{\boldsymbol{\xi}}}$ with a bounded value dependent on the sign of the gradient:

$$\boldsymbol{\mu}_{q,\dot{\boldsymbol{\xi}}} = \begin{cases} \boldsymbol{\mu}_{q-1,\dot{\boldsymbol{\xi}}} + \boldsymbol{\Delta}_q^u & \text{if } \boldsymbol{w}_q > 0 \\ \boldsymbol{\mu}_{q-1,\dot{\boldsymbol{\xi}}} - \boldsymbol{\Delta}_q^u & \text{if } \boldsymbol{w}_q < 0 \\ 0 & \text{else} \end{cases} \tag{4.30}$$

Where Δ^u is a bounded value computed as follows:

$$\boldsymbol{\Delta}_q^u = \begin{cases} \eta^+ \boldsymbol{\Delta}_{q-1}^u & \text{if } \boldsymbol{w}_q \boldsymbol{w}_{q-1} > 0 \quad \& \quad \boldsymbol{\Delta}_{q-1}^u < \epsilon_s \\ \eta^- \boldsymbol{\Delta}_{q-1}^u & \text{if } \boldsymbol{w}_q \boldsymbol{w}_{q-1} < 0 \quad \& \quad \boldsymbol{\Delta}_{q-1}^u > \epsilon_i \\ \boldsymbol{\Delta}_{q-1}^u & \text{else} \end{cases} \tag{4.31}$$

Where ϵ_s and ϵ_i are respectively the superior and inferior born of Δ^u and $0 < \eta^- < 1 < \eta^+$ are factors used to increase or decrease the update value Δ^u. In other terms, as long as the gradient keeps the same sign, Δ^u increases until reaching the maximum value of ϵ_s, but if the gradient changes its sign, it means that Δ^u was too big and that we have passed an optimum. We then reduce the value of Δ^u before continuing the optimization. This system has more parameters to tune and converges more slowly to a solution, but it is robust to an eventual explosion of the update term. Even in the vicinity of the obstacle limits, if the gradient becomes to steep, the update will not be bigger than ϵ_s. Thus, the system is much more stable.

Once the parameters $\boldsymbol{\mu}_{\dot{\boldsymbol{\xi}}}$ are updated, we have to forget a part of the sufficient statistics $\boldsymbol{A}$ and $\boldsymbol{b}$ in order to approximate the new gradient $\boldsymbol{w}$.

$$\boldsymbol{A}_q \leftarrow \beta \boldsymbol{A}_q \tag{4.32}$$

$$\boldsymbol{b}_q \leftarrow \beta \boldsymbol{b}_q \tag{4.33}$$

where $\beta \in \mathbb{R}_{[0,1]}$. The algorithm converges then to an optimum with respect to the reward.

4.5.5 Stopping Conditions

The algorithm presented in 4.5 converges to an optimum, thus, a convergence criterion has been chosen to stop the algorithm. A window h_r is defined in which one we test the convergence of the algorithm. For that, we fit a straight line L_r on the reward function in the window h_r, and determine a threshold for the slope of L_r under which one L_r is considered has horizontal. At this point we consider that a complete convergence has been reached:

$$\frac{\partial L_r}{\partial q} < \epsilon_r \tag{4.34}$$

Where $\frac{\partial L_r}{\partial q}$ is the slope of the line L_r and ϵ_r is a threshold determined empirically.

However, depending on the trajectory tested using the reward function, we do not need to wait for a complete convergence of the system. E.g. in the first version of the system, the evaluation was done on the noisy modulation trajectory $\dot{\boldsymbol{\xi}}^r$. Thus, waiting for the complete convergence means waiting for the modulation trajectory to reach the goal. The point is that it is not useful in

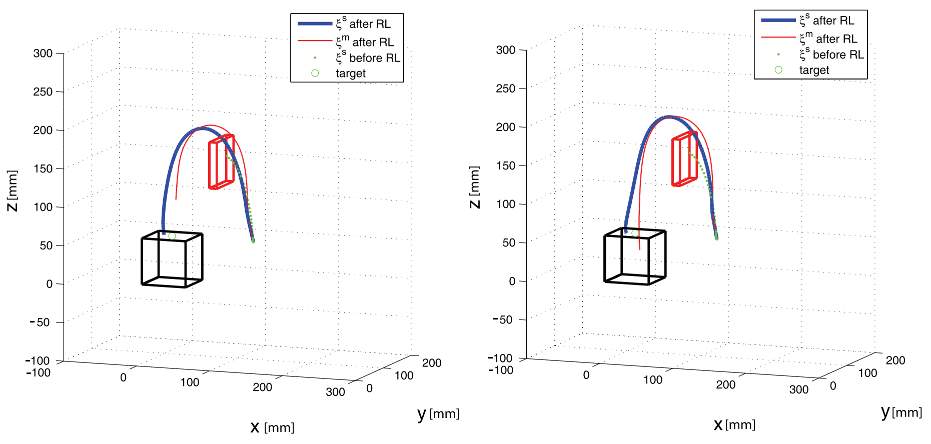

Figure 4.7: **Left:** The DS alone would have produced the trajectory represented by the dashed line. By using the reinforcement learning module, the first version of the algorithm is able to find a different trajectory that avoids the obstacle (in thick line). The trajectory produced by the modulation speed $\dot{\boldsymbol{\xi}}^m$ (in thin line) does not reach the goal. The learning has been interrupted before the convergence because a satisfying solution has been found for ξ^d. **Right:** Here, we can see that the trajectory produced by $\dot{\boldsymbol{\xi}}^m$ reaches the goal. In this example, we use a convergence criterion to stop the learning, in order to show the convergence of the reinforcement learning module.

our case because we have the component of the DS that guarantee us to reach the goal as soon as the modulation allows it. In addition to that, if we choose to let the system converge to a optimum solution, the RL module will be used even if there are no obstacles. It means that it will cost a lot of time even if the GMM model and the DS system are able to find a solution.

Thus, during the learning phase, at each update of the means $\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}$, the new trajectory $\boldsymbol{\xi}^s$ produced by the dynamical system was tested in order to be able to stop the learning as soon as a satisfying solution has been found. Because we are only interested in goal-directed reaching movements, only the distance between the last point of the trajectory $\boldsymbol{\xi}^s_T$ and the target was tested. The task were considered as fulfilled if $|\boldsymbol{\xi}^s_T - \boldsymbol{\xi}^g| < d$ where $d \in \mathbb{R}$ represents the maximal distance we want to obtain.

Figure 4.7 shows the two versions of trajectories obtained. On the left, the learning has been stopped by the distance criterion. We can observe that the trajectory produced by the modulation speed $\dot{\boldsymbol{\xi}}^m$ (in thin line) does not reach the goal. On the right, the learning is stopped when the algorithm has converged to a optimum solution for $\dot{\boldsymbol{\xi}}^m$, we can observe that $\dot{\boldsymbol{\xi}}^m$ does reach the target. Figure 4.8 represents the time gain in term of learning step for the two cases presented in Figure 4.7

One problem of this solution is that we do not use the cost function to determine the stopping condition, but another metric. For consistency, we thus have tested a third stopping criterion with the second version of the algorithm.

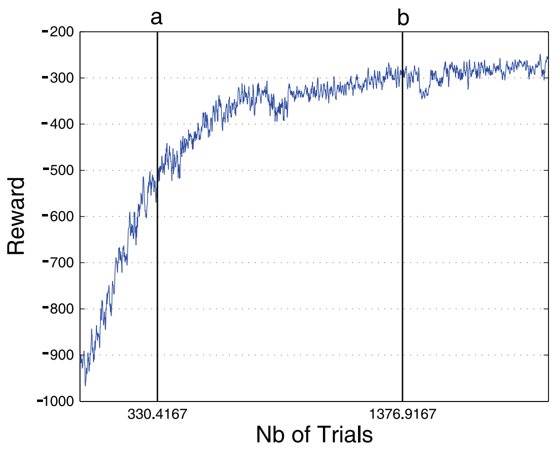

Figure 4.8: This diagram represents the evolution of the reward function during the learning phase of the two examples shown in Figure 4.7. This curve is a mean over 12 runs of the system in identical situations. The first vertical line (a) represents the average number of trials needed when the learning is interrupted as soon as a satisfying solution is found. The second vertical line (b) represents the average number of trials needed when the learning is interrupted after the complete convergence (when the modulation trajectory reaches the goal).

For that, as before, we test only the smooth trajectory $\xi^d(t)$ generated at each update of the means $\boldsymbol{\mu}_{\dot{\boldsymbol{\xi}}}$. However, instead of the distance to the goal, we use the reward function $r_q(\xi^d)$ that was defined for the learning and the stopping criterion is determined by a threshold R_t defined empirically:

$$r_q(\xi^d) > R_t \tag{4.35}$$

Here, as in the second stopping criterion, the main advantage is that when the first trajectory is tested and the task is fulfilled successfully, we do not run the RL algorithm.

4.6 Experiments with NAC

The task chosen for testing the algorithm consists in putting an object into a box. We use the humanoid robot HOAP3 from Fujitsu. It has a total of 25 DOFs, but here for the manipulation tasks , we use only one 4 DOFs arm. In a first phase, the robot has been trained by demonstrating kinesthetically the task 26 times, starting in each case with a different initial arm posture and reaching for a different target location.

During each demonstration, the robot recorded the trajectories followed passively by its 4 DOFs using the motor encoders. What the robot was expected to learn throughout the demonstration was that, no matter where the box was located, what mattered (i.e. what remained consistent throughout the demon-

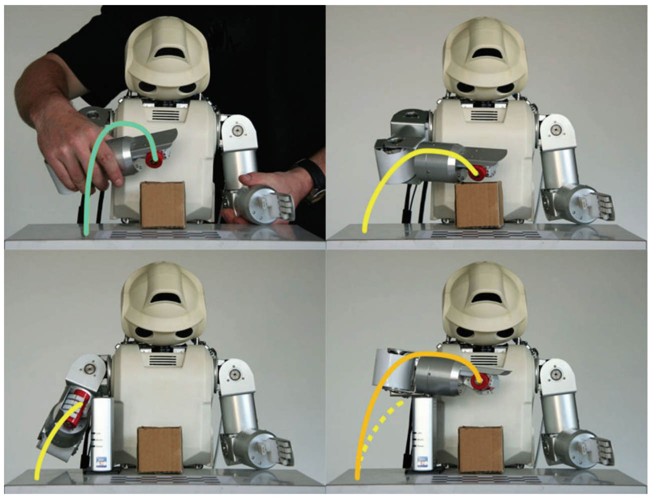

Figure 4.9: Programming a robot by demonstration means that a user demonstrates a task and the robot has to extract the important features of the task in order to be able to reproduce it in different situations. In special cases, it might happen that the robot encounters a new situation where following closely the demonstration does not help to fulfill the task. In this case, there are two possibilities, the first one is to teach the robot how to handle the new situation and the second one is to let the robot learn by itself how to fulfill the task. In this paper we focus on the second possibility.

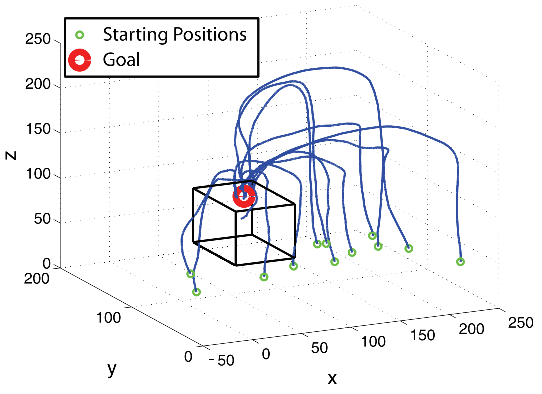

Figure 4.10: In this example, the Gaussian Mixture Model is trained with 12 kinesthetic demonstrations of the task shown in Figure 4.9. Here, for easiness of reading, the demonstrations have been done with a fixed position for the box, but for the experiments presented in this chapter, we have changed the position of the box for each demonstration.

strations) was that the box should be reached from above (i.e. by looming over the box).

Once the robot has extracted the generalized speed profile $\dot{\boldsymbol{\xi}}^m$ of the task, the robot is able to reproduce the task with various starting positions and for various locations of the box, by using only the DS modulation (see (Hersch, Guenter, Calinon, & Billard, 2006)). In order to test the RL module of our algorithm, we have introduced an obstacle lying in the trajectory of the robot's end effector. In this case, the DS modulation fails to bring the robot's hand to the target properly. The RL module helps then the robot reach the goal by avoiding the obstacle while trying to still satisfy the constraints of the movement shown during the demonstrations.

4.6.1 Early results

This section will presents results obtained with the first version of the algorithm. These results are presented to emphasize the great amelioration obtained with small modifications of the system.

In order to study the convergence property of the RL algorithm, we have first conducted a series of tests to have statistics on the number of trials that are needed to converge to a solution. The box and the obstacle have the same position for all the experiments, only the starting position varies. We have defined 23 starting positions which are equally distributed along a vertical line. For each of these positions, we have made 22 runs of the RL algorithm. Here, only the modulation trajectory $\boldsymbol{\xi}^m$ is optimized. In Figure 4.12, $\boldsymbol{\xi}^m$ is represented alone without the component of the dynamical system. To limit the computation time of the simulation, we have limited the maximum number of steps to $10'000$ steps.

Figure 4.11 represents the statistics (mean and standard deviation) for the 23 starting positions. Position number 1 is the lowest position on the vertical line while position 23 is the highest (see sample trajectories in Figure 4.12). We can see on Figure 4.11 that the RL algorithm has more difficulties finding a solution for the lower starting positions. This is certainly due to the fact that it is more difficult to keep the bell shape of the demonstrated trajectory while finding a way between the obstacle and the box than satisfying the same bell shape by passing above the obstacle.

This phenomenon can also be observed in Figure 4.12. These diagrams represent the different trajectories for 6 chosen starting positions [1]. The large standard deviations we observe in Figure 4.11 for the lower positions are due to the failure of the RL algorithm.

The same type of experiments has been conducted using the second stopping criterion as described in Section 4.5.5. Here, we stop the learning as soon

[1]The trajectories for which the RL algorithm has not found a satisfying solution within the $10'000$ trials are not shown in those graphics.

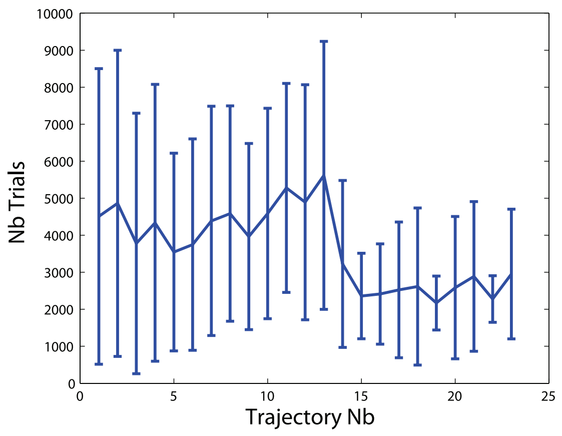

Figure 4.11: This diagram represents the statistics of the number of trials needed to converge to a solution. As abscissa, we have the starting position (23 starting positions equally distributed along a vertical line, the position number 1 is the lowest and the number 23 is the highest position). As ordinate, we have the mean and standard deviation of the number of trials for each run of the RL algorithm. We observe here that the solution seems to be more difficult to obtain with the lower starting points.

as a satisfying condition has been found for the trajectory generated by the modulated DS. In other words, the learning is stopped when $|\boldsymbol{\xi}^s_T - \boldsymbol{\xi}^g| < d$ where d is a distance threshold.

Figure 4.14 represents the statistics (mean and standard deviation) for the 23 starting positions. In Figure 4.14, we can clearly observe the positions where it is more difficult for the system to find a solution. In the lower positions, the initial trajectory is blocked near the lower bound of the obstacle. Since the exploration of the solution space by the system is limited, a solution is quickly found. In the middle position, it is more difficult for the system to find a solution, the initial trajectory is blocked far from the bound of the obstacle. For position 17 and above, in this example, the initial trajectory reaches the goal, there is thus no need to use the RL module. Compared to the results obtained with the convergence criterion, we have here a more efficient algorithm in terms of convergence time.

In Figure 4.12, we observe the different trajectories for 6 chosen starting positions. An interesting point is that for position 16 (see Figure4.12), the algorithm finds a better way by passing aside the obstacle and not above or below. We can also observe that, in the last example, there is no variance in the trajectories reproduced by the system. This is due the the fact that we do not use the RL in this case.

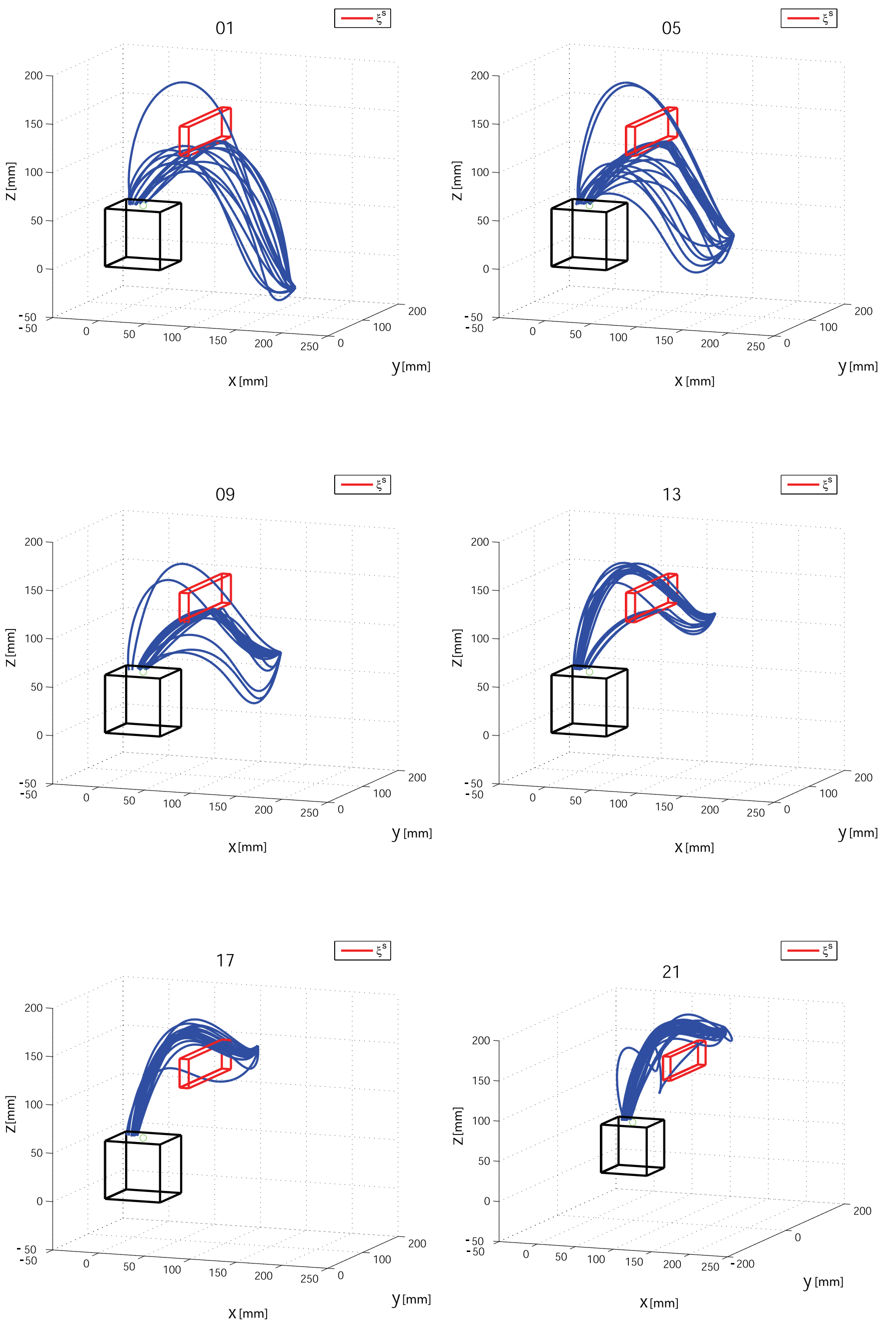

Figure 4.12: We observe here the different trajectories for 6 selected starting positions. The number for each diagram corresponds to a position represented in Figure 4.11. The number 01 represents the lowest starting point and the 23 the highest. We have not represented the trajectories for which the RL module haven't found a solution within the 10′000 trials.

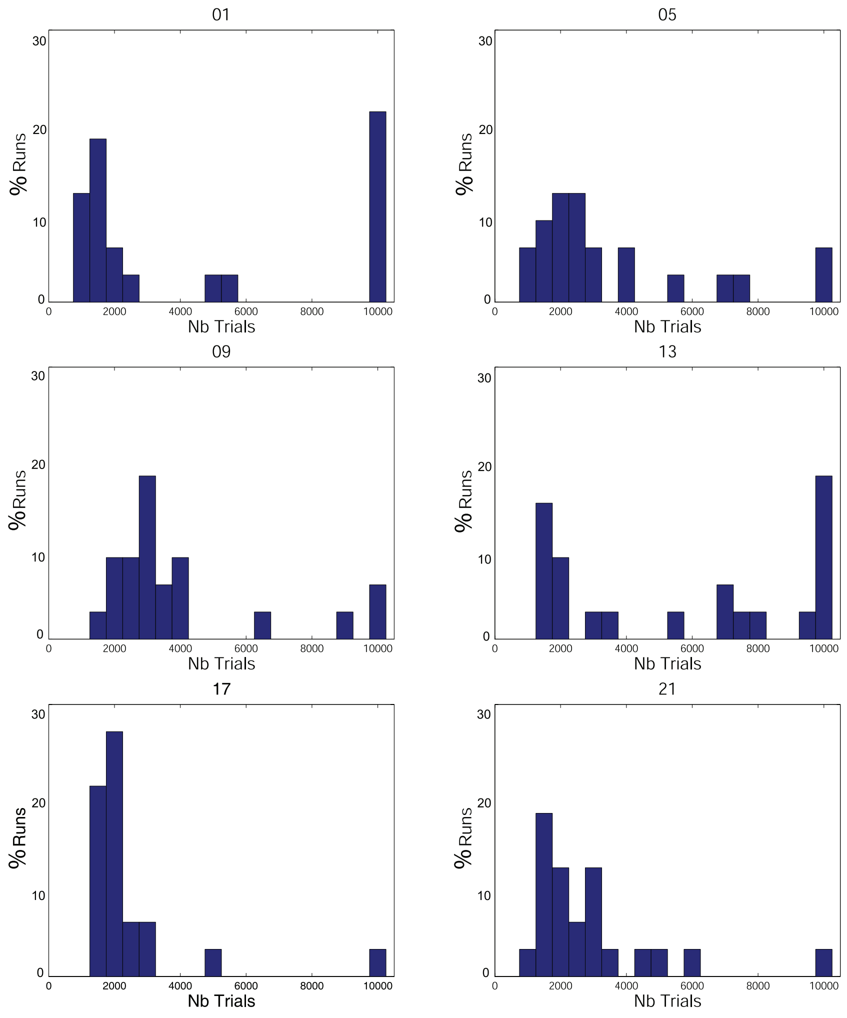

Figure 4.13: These diagrams represents the histogram of the number of trials needed to converge to a solution for the 6 starting position shown in Figure 4.12. The number of trials was limited to 10′000. We observe here that, for each starting point, there is at least one run where the RL algorithm has not found a satisfying solution. As in Figure 4.12, we see that this problem is more important in the lower starting positions.

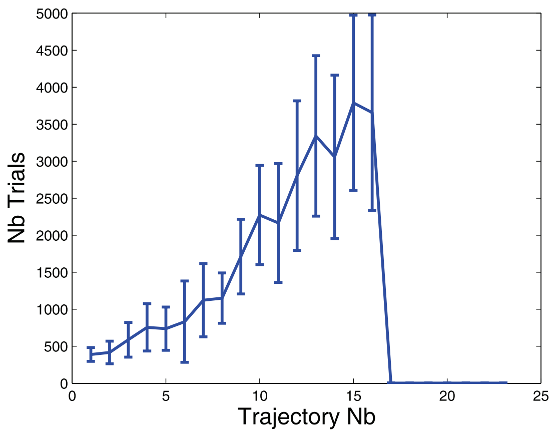

Figure 4.14: This diagram represents the statistics of the number of trials needed to converge to a solution. As abscissa, we have the starting position (23 starting positions equally distributed along a vertical line, the position number 1 is the lowest and the number 23 is the highest position). As ordinate, we have the mean and standard deviation of the number of trials for each run of the RL algorithm.

4.6.2 Results for the new system

As in Section 4.6.1, the whole system will be tested using two stopping criterion. The first one is a convergence criterion and the second one a threshold on the reward function as described in Section 4.5.5. The experiment is the same as before, the box and the obstacle are fixed and the starting positions are distributed along a vertical line.

Here we test the convergence criterion. In the new system, during the learning, the trajectory generated by the DS modulated by the noisy speed profile is evaluated with the reward function. It is then more consistent to let the system converge to a optimum.

Figure 4.17 represents the statistics for the 23 starting positions we tested. As for the first version, we can see on Figure 4.17 that the RL algorithm has more difficulties to find a solution for the middle starting positions. Another thing to observe is that for the 3 lowest and 8 highest starting positions, even if the first solution is not blocked by the obstacle, the system needs several trials to satisfy the convergence criterion. It means that even if the trajectory reaches the goal, it is not a optimum trajectory with respect to the reward function. This is mainly due to the first term of the new reward function presented in Section 4.5.2. This term penalizes trajectories that are different for the original modulation trajectory. Now, by using the DS, we add a component that deforms the original trajectory, it is then possible to optimize this trajectory in order to decrease this deformation.

It is interesting to see in Figure 4.18 the dispersion of the different trajectories for each starting position. For the first starting position, even if the first

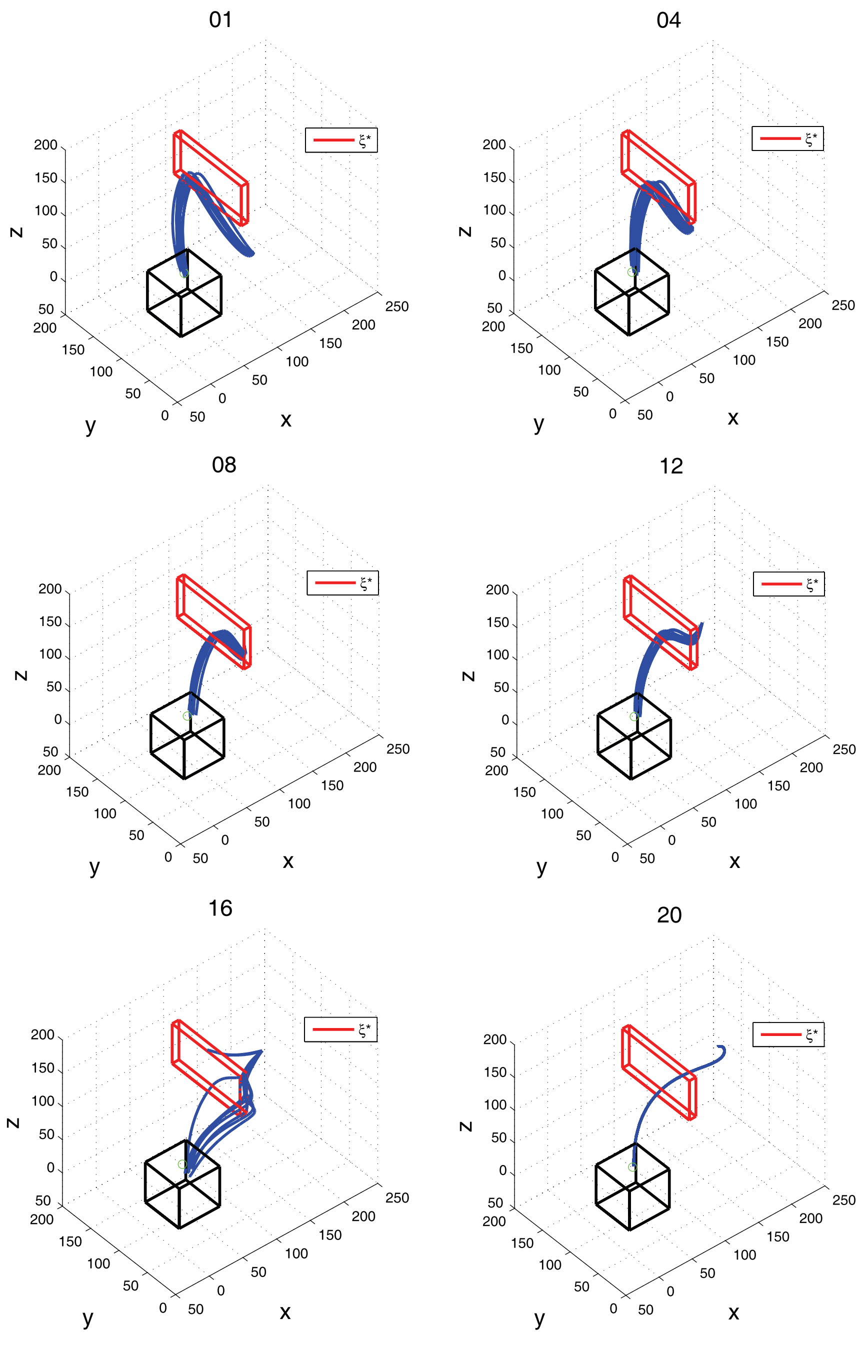

Figure 4.15: We observe here the different trajectories for 6 selected starting positions. The number of each diagram corresponds to a position represented in Figure 4.11. The number 01 represents the lowest starting point and the 23 the highest. We have not represented the trajectories for which the RL module haven't found a solution within the $10'000$ trials.

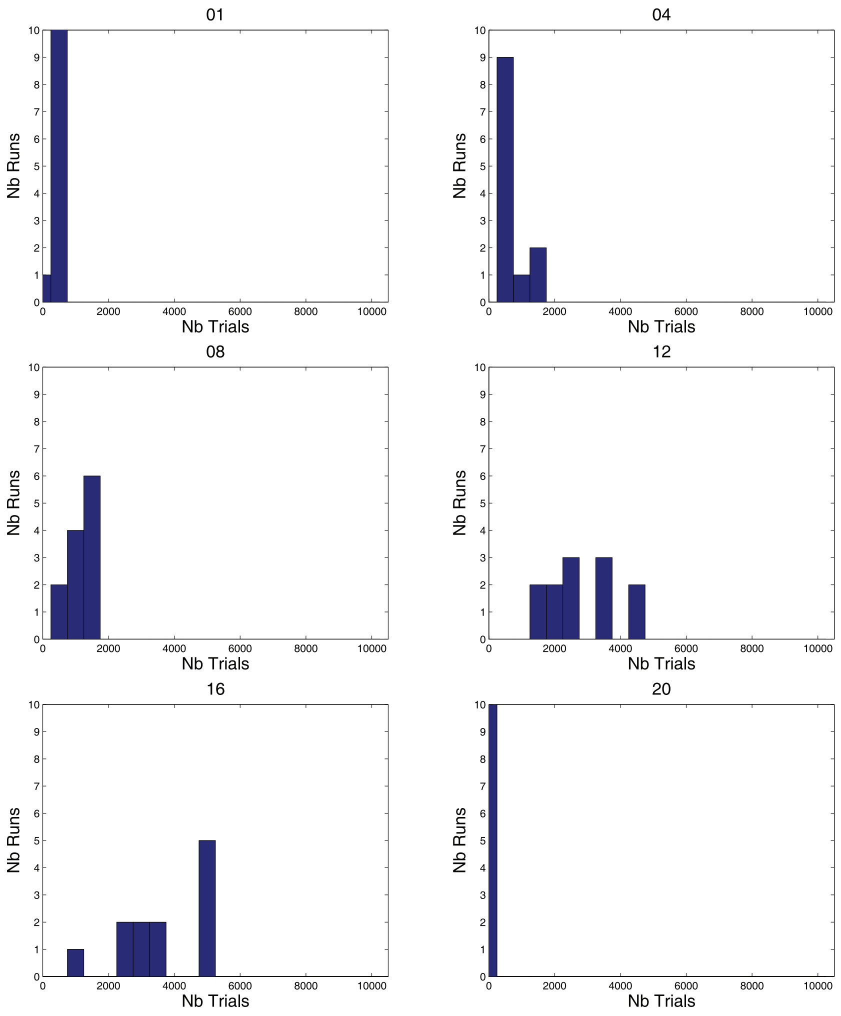

Figure 4.16: These diagrams represents the histogram of the number of trials needed to converge to a solution for the 6 starting position shown in Figure 4.12. During the experiment, the number of trials where limited to $10'000$, we observe here that there are for each starting points at least one runs where the RL algorithm has not found a satisfying solution. As in Figure 4.12, we see that this problem is more important in the lower starting positions.

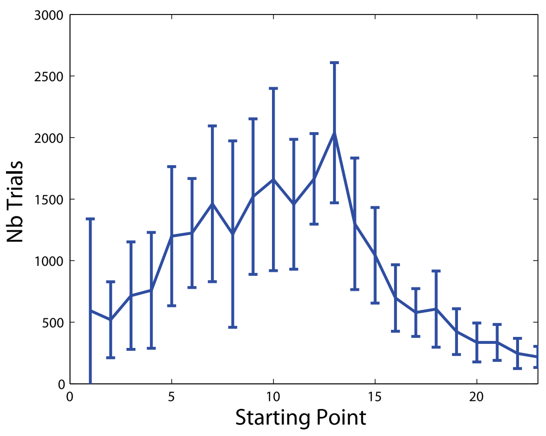

Figure 4.17: This diagrm represents the statistics of the number of trials needed to converge to a solution. As abscissa, we have the starting position (23 starting positions equally distributed along a vertical line, the position number 1 is the lowest and the number 23 is the highest position). As ordinate, we have the median and interquartile range of the number of trials needed to find a satisfying solution for each starting point of the experiment.

trial reaches the goal, the RL module is used in order to satisfy the convergence criterion. It seems that some solutions tend to shorten the path for the first starting position. This is certainly due to the fourth term of the new reward function that penalizes long paths.

We can already observe a big amelioration in terms of convergence time compared to the result of the first version of the system. In addition, in these experiments, a solution has been found for each run of the system and the instability problem disappeared.

However, it still take some times to find a solution when the modulated DS is sufficient to reach the target. To avoid this, a second series of experiment has been realized with a threshold criterion on the reward function as defined in Section 4.5.5.

Figure 4.20 represents the statistics for the threshold stopping criterion. Compared to Figure 4.17, we observe now very clearly for which starting positions the obstacle crosses the path of the robot's arm. Here, the system needs much less trials to find a satisfying solution. Even if the obstacle interferes with the trajectory generated by the modulated DS, the number of steps needed to find a satisfying solution is smaller. And compared with the best result of the first system version, we need three times less steps on average to find a solution with an increased reliability.

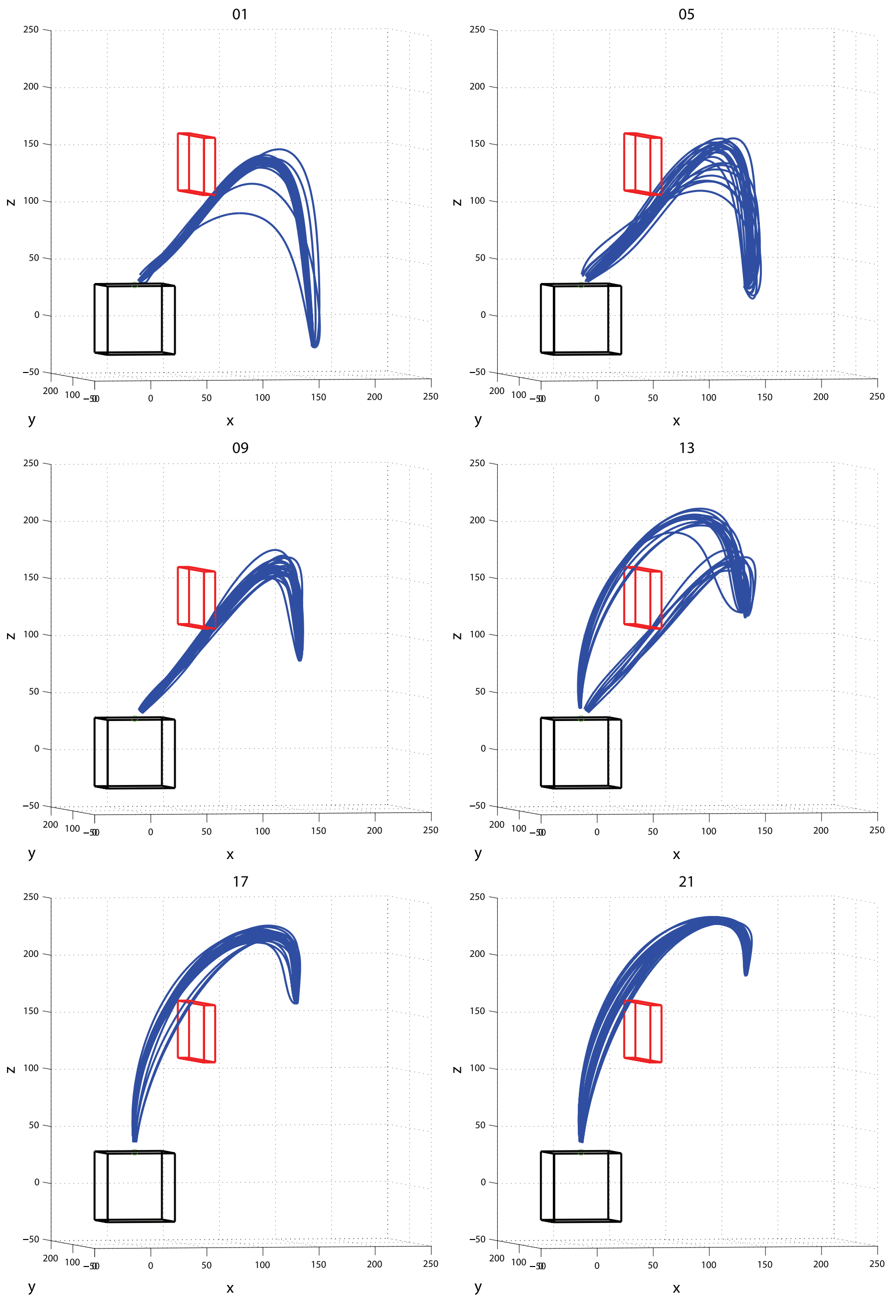

Figure 4.18: We observe here the different trajectories for 6 selected starting positions. The number of each diagram corresponds to a position represented in Figure 4.17. The number 01 represents the lowest starting point and the 23 the highest. We can observe here that the algorithm find always a solution to reach the goal.

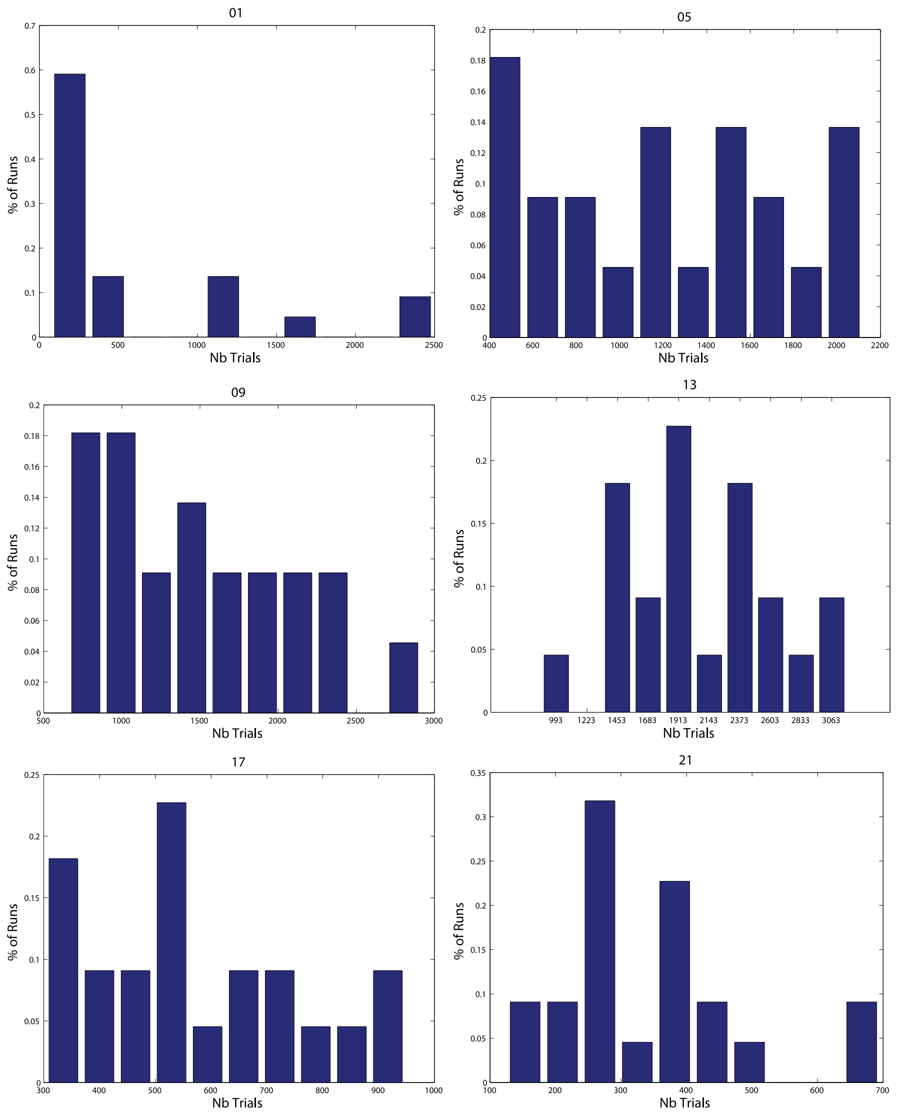

Figure 4.19: These diagrams represent the histogram of the number of trials needed to converge to a solution for the 6 starting position shown in Figure 4.18. During the experiment, the number of trials where limited to 10′000, we observe here that we are far from reaching this limit. The starting position number 13 has the highest mean, but nevertheless, the algorithm never needed more than 3100 trials to find a solution.

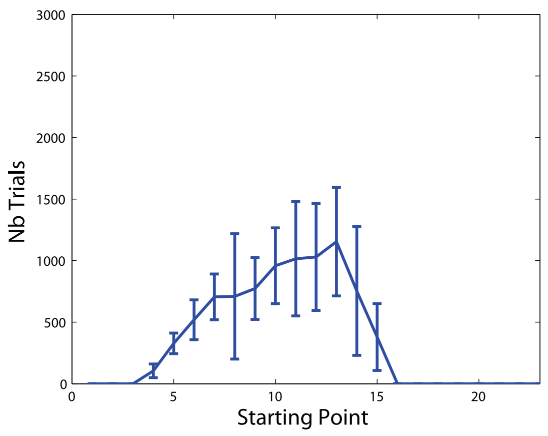

Figure 4.20: This diagram represents the statistics of the number of trials needed to converge to a solution. As abscissa, we have the 23 starting position. As ordinate, we have the median and interquartile range of the number of trials needed to find a satisfying solution for each starting point of the experiment.

4.7 The extended Taguchi method

By using the NAC (Natural Actor Critic, see Section 4.5) for the trial-and-error module, there are a lot of parameters to tune in order to have a stable system, especially when it is used in combination with the DS (see Section 4.6). Moreover, the problem presented here can be viewed as an optimization problem. Thus, in order to have a idea of the efficiency of the RL algorithm to solve this problem compared to other optimization algorithms, we have implemented an algorithm from operations research inspired by the Taguchi method for fractional factorial design of experiment[2]. This type of algorithm is able to deal with several parameters while ensuring a smooth convergence to an optimal solution in a bounded workspace. The NAC represents a stochastic exploration of an unbounded solution space to find an optimal solution while the extended Taguchi method represent an ordered and systematic exploration of a bounded workspace.

The main idea of the Taguchi method is to define a couple of bounded parameters that are supposed to influence the output of the system. These parameters are then divided in two or three levels that will be tested in a set of experiments. To setup this set of experiments, we use different permutations of the parameters level. Ideally, the best set of experiments would be the one in which we test every possible permutation of these levels for each parameter. The problem is that it becomes quickly too expensive in terms of computation time. In fact, by choosing three levels by parameters and considering k factors, we will have to perform 3^k experiments to test all possible permutations. This technique is commonly called full factorial design. A more efficient alternative

[2]from now this algorithm will be called the extended Taguchi method

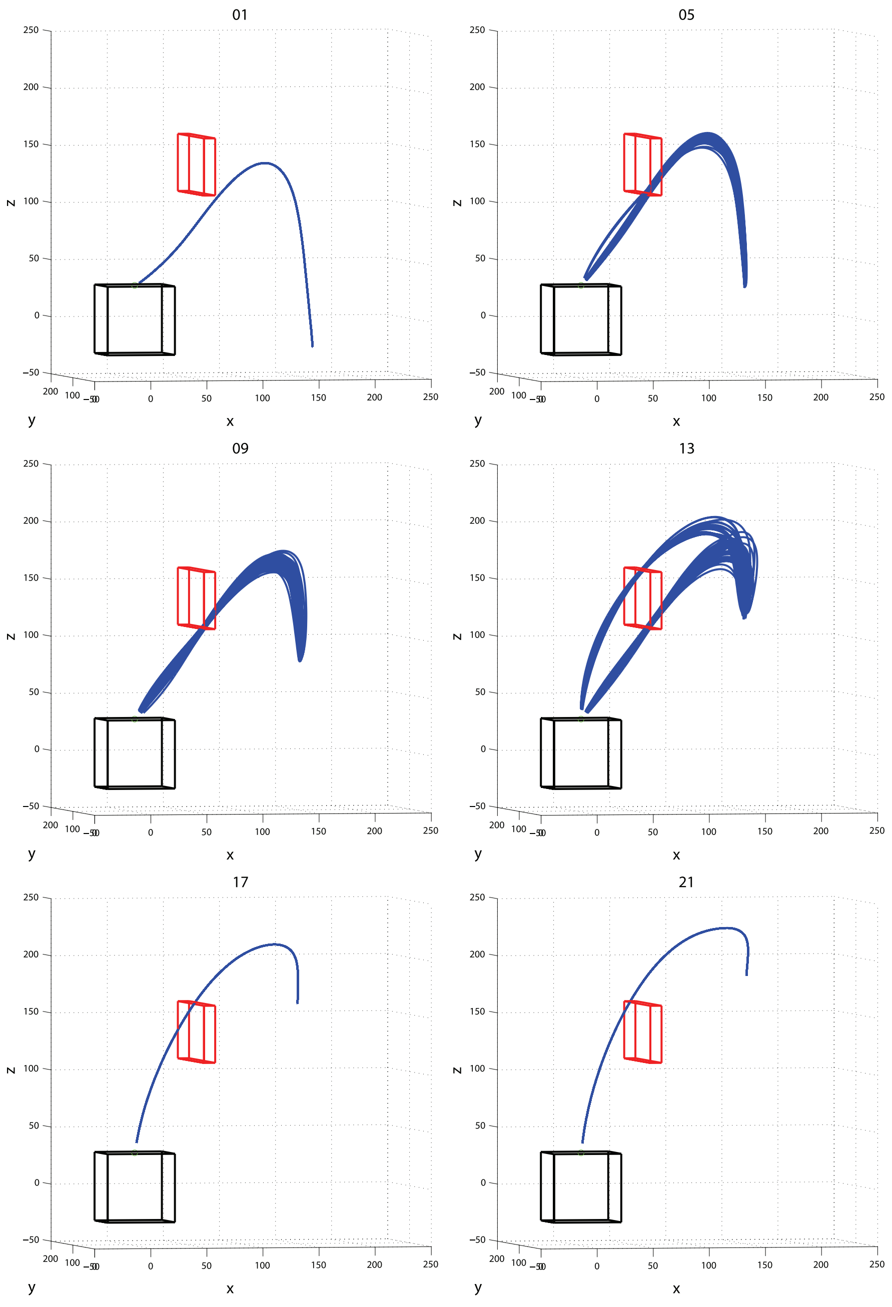

Figure 4.21: We observe here the different trajectories for 6 selected starting positions. The number of each diagram corresponds to a position represented in Figure 4.20. In comparison with the diagrams shown in Figure 4.18, we do not observe any dispersion for diagrams Nb 1, 17 and 21. Here the system tests the first trajectory using the reward function. If the reward is bigger than the threshold, there is no need to use the RL.

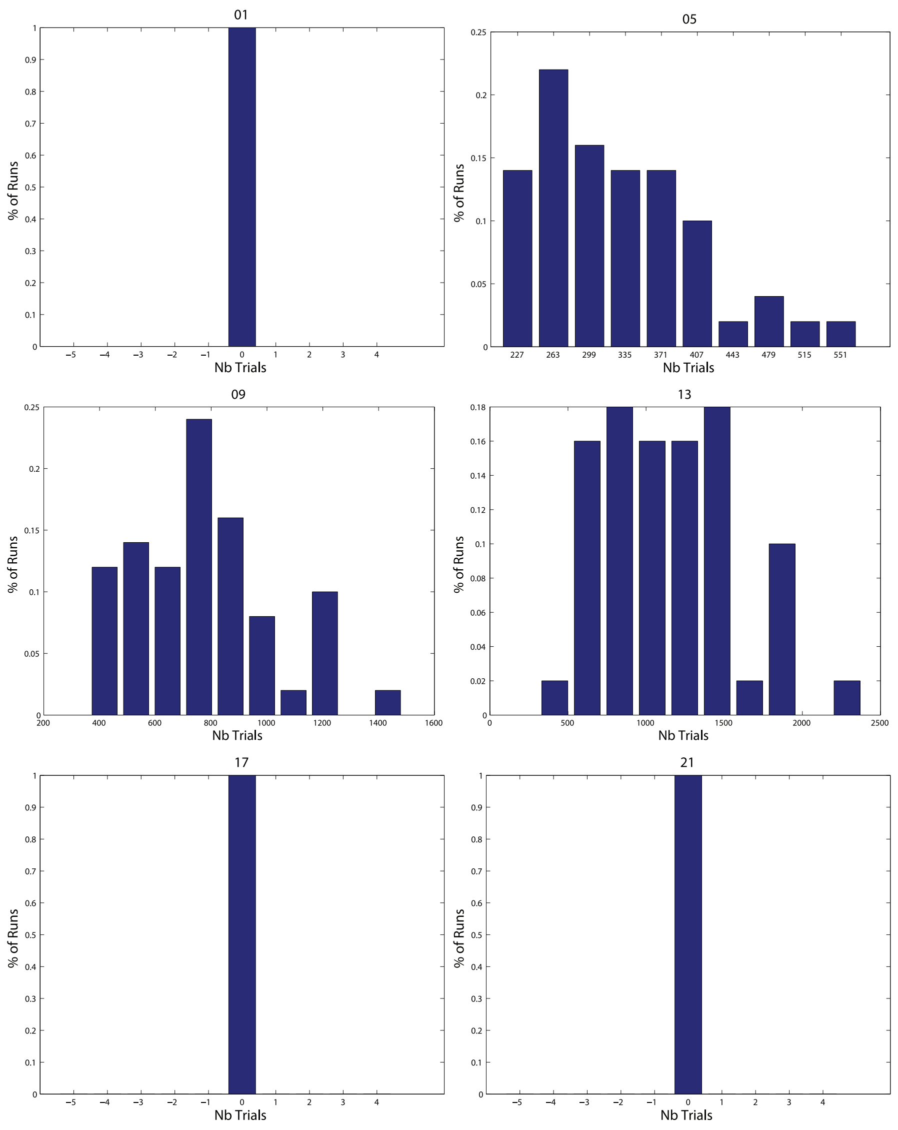

Figure 4.22: These diagrams represent the histogram of the number of trials needed for convergence to a solution for the 6 starting positions shown in Figure 4.21. As the RL module has not been used at all in diagrams Nb 1, 17 and 21, the respective histograms are useless.

is to use a fractional factorial design. It consists in taking only a part of the complete set of experiments. Following this idea, Genichi Taguchi (Roy, 1990) proposed a special set of Orthogonal Arrays (OA) to lay out a set of experiments originally for the optimization of the quality of manufacturing processes. A complete study on OA can be found in (Hedayat, 1999).

4.7.1 Description of the algorithm

In a first step, we define the parameters we want to optimize for the execution of the task and their bounds. Here, as before, we will use the gaussian centers $\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}$ of the GMM (see Section 4.5). Here, k stands for the k^{th} Gaussian and $\dot{\boldsymbol{\xi}}$ stands for the parameter we want to modelize with the GMM, i.e. the joint angle speed profiles. For each vector of parameters, we define a lower bound and an upper bound vector, respectively $\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}^{min}$ and $\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}^{max}$ of dimension n, the number of DOFs of the robot's arm. As the GMM modelizes a speed profile for the joint angles of the HOAP 3 robot, we can choose the lower and upper bounds following the specifics given by Fujitsu. From now on, we will work exclusively with three levels per parameter and we will omit the subscript $_{\dot{\boldsymbol{\xi}}}$ for easiness of reading. In order to define the parameters levels, we begin with the level 2. At the first iteration, $\boldsymbol{\mu}_{k,1}^{2}$ is defined by:

$$\boldsymbol{\mu}_{k,1}^{2} = \frac{\boldsymbol{\mu}_{k}^{min} + \boldsymbol{\mu}_{k}^{max}}{2} \tag{4.36}$$

Where $\boldsymbol{\mu}_{k}^{min}$ and $\boldsymbol{\mu}_{k}^{max}$ are respectively the lower and upper bounds of parameter $\boldsymbol{\mu}_{k}$, in $\boldsymbol{\mu}_{k,1}^{2}$ the subscript k stands for k^{th} Gaussian center, the subscript 1 stands for the first iteration and the superscript 2 stands for level 2. $\boldsymbol{\mu}_{k,1}^{2}$ is then defined by the middle range of $\boldsymbol{\mu}_{k}$.

The value $\boldsymbol{\mu}_{k,1}^{1}$ and $\boldsymbol{\mu}_{k,1}^{3}$ are then computed by respectively subtracting and adding to $\boldsymbol{\mu}_{k,1}^{2}$ a range $R_{k,1}$ defined by:

$$R_{k,1} = \frac{\boldsymbol{\mu}_{k}^{max} - \boldsymbol{\mu}_{k}^{min}}{4} \tag{4.37}$$

Thus,

$$\boldsymbol{\mu}_{k,1}^{1} = \boldsymbol{\mu}_{k,1}^{2} - R_{k,1} \tag{4.38}$$

$$\boldsymbol{\mu}_{k,1}^{3} = \boldsymbol{\mu}_{k,1}^{2} + R_{k,1} \tag{4.39}$$

Once the three levels are defined for each parameter, by using a OA, we will conduct a set of experiments to determine the best combination of levels for the iteration q. In the OA, each column corresponds to a parameter and each row to an experiment. The value for the j^{th} element of the i^{th} raw defines then the level (1, 2 or 3) to use for the j^{th} parameter in the i^{th} experiment of the iteration q (an example of OA is shown in A.2).

For each experiment i of the iteration q (i.e. for each row of the OA), we compute a fitness using a fitness function. In contrast to the reward function used in the NAC, the fitness function must be here positive definite. In our case, the fitness function is defined by the absolute value of the reward function defined in Section 4.5.2:

$$f_{q,i} = abs((\sum_{t=0}^{T} -c_1|\dot{\boldsymbol{\xi}}_t^s - \dot{\boldsymbol{\xi}}_{t,q=1}^m| - c_2|\dot{\boldsymbol{\xi}}_t^s - \dot{\boldsymbol{\xi}}_{t-1}^s|) \qquad (4.40)$$

$$-c_3|\boldsymbol{\xi}_T^s - \boldsymbol{\xi}^g| - c_4 T^g \qquad (4.41)$$

where $c_1, c_2, c_3, c_4 \in \mathbb{R}$ are weighting constants, $\boldsymbol{\xi}^s$ is the simulated state of the robot, $\boldsymbol{\xi}_{t,q=1}^m$ is the modulation speed of the first trial retrieved using GMR (see Equation 3.24), $\boldsymbol{\xi}^g$ is the target position and T^g is the time needed to reach the goal. The Fitness allows us to compute the Signal/Noise (S/N) ratio for each experiment:

$$\eta_{q,i} = -20log(f_{q,i}) \qquad (4.42)$$

By using the S/N ratio, we give more importance to the experiments with a small fitness (in our case the best trajectories) and much less importance to the experiment with a bigger fitness. Depending on the optimized system and on the parameter combination, it can happen that the fitness of the generated trajectories explods. The S/N ratio is used to avoid a too big influence of such trajectories on the parameter optimization process. It can be compared to a low pass filter that allows the system to filter the trajectory with a bad fitness. Thus, we have now 54 S/R ratio $\boldsymbol{\eta_{q,s}}$ attached to the 54 experiments of the OA.

By using the S/N ratio, we have now to build a response table. The response table represents all levels of all parameters. For each level m of each parameter n, we compute the mean S/N ratio obtained during the 54 experiments. It means that for a given column of the OA (i.e. a given parameter), we sum the S/N ratio obtained when the given level is used, and we divide the result by the number of terms of the sum.

$$\bar{\eta}(m,n)_q = \frac{3}{54} \sum_{i,OA(i,n)=m} \eta_{q,i} \qquad (4.43)$$

In this response table, for each parameter n, the level with the largest $\bar{\eta}_q$ will be identified as the optimal level for the corresponding parameter.

The next step is to run an experiment using the optimal parameters $\boldsymbol{\mu}_{k,q}^{opt}$ in order to compute the reference fitness $f_{ref}(q)$ for the iteration q. As the optimal combination may not appear in the OA, this must be done in order to have an indication on the efficiency of this combination of parameter value and then to determine the convergence of the system along the different iterations.

For the next iteration q, we redefine the 3 levels for each parameter. Each parameter $\boldsymbol{\mu}^2_{k,q}$ is redefined by:

$$\boldsymbol{\mu}^2_{k,q} = \boldsymbol{\mu}^{opt}_{k,q-1} \tag{4.44}$$

As before, levels 1 and 3 will be computed by using an updated range $R_{k,q}$:

$$R_{k,q} = \alpha_{red} R_{k,q-1} \tag{4.45}$$

Where α_{red} is a reducing rate that is used to reduce the optimization range as a function of the iteration step. Depending on the system, this coefficient can be adjusted in order to optimize convergence time. A reduction rate α_{red} close to 1 will induce a system that takes more time to converge and a smaller reduction rate may cause the failure of the system by reducing too much the search space.

With this kind of system, we have to check if the new value for $\boldsymbol{\mu}^1_{k,q}$ or $\boldsymbol{\mu}^3_{k,q}$ are inside the bounds fixed for the parameters optimization. The update is done as follows:

$$\boldsymbol{\mu}^1_{k,q} = \begin{cases} \boldsymbol{\mu}^1_{k,q-1} - R_{k,q} & \text{if } \boldsymbol{\mu}^1_{k,q-1} - R_{k,q} \geq \boldsymbol{\mu}^{min}_k \\ \boldsymbol{\mu}^{min}_k & \text{if } \boldsymbol{\mu}^1_{k,q-1} - R_{k,q} < \boldsymbol{\mu}^{min}_k \end{cases} \tag{4.46}$$

$$\boldsymbol{\mu}^3_{k,q} = \begin{cases} \boldsymbol{\mu}^3_{k,q-1} + R_{k,q} & \text{if } \boldsymbol{\mu}^3_{k,q-1} + R_{k,q} \leq \boldsymbol{\mu}^{max}_{\boldsymbol{k}} \\ \boldsymbol{\mu}^{max}_k & \text{if } \boldsymbol{\mu}^3_{k,q-1} + R_{k,q} > \boldsymbol{\mu}^{max}_k \end{cases} \tag{4.47}$$

If $\boldsymbol{\mu}^1_{k,q}$ or $\boldsymbol{\mu}^3_{k,q}$ are out of bounds, they are set respectively to the lower bound $\boldsymbol{\mu}^{min}_k$ or to the upper bound $\boldsymbol{\mu}^{max}_{k,q}$ depending on the case.

The algorithm is then run until complete convergence. The criterion used to determine the convergence is a threshold on the difference between the two last reference fitness. The algorithm is stopped when:

$$f_{ref}(q) - f_{ref}(q-1) < F \tag{4.48}$$

Where F is a threshold set to 0.05 for the experiments done in Section 4.8.

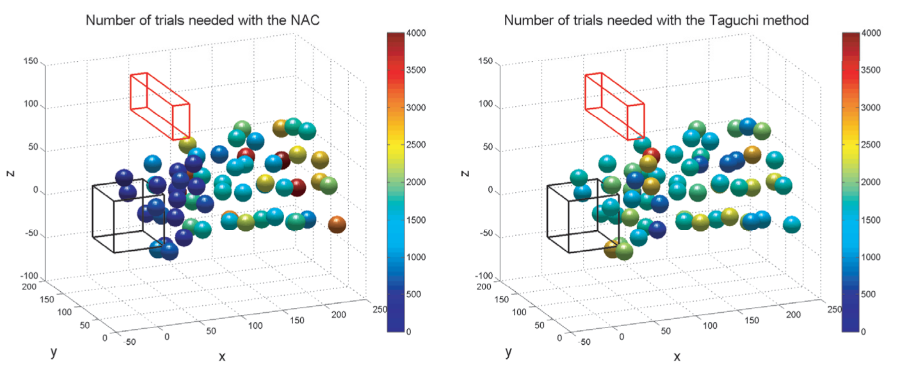

Figure 4.23: These diagrams represent the statistics done in order to compare the NAC and the extended Taguchi method on the same setup. This experiment was done on the system described in Section 4.2 and the learned parameters are the Gaussian centers of the GMM that modelized the desired speed $\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}$ used to modulate the DS. The two diagrams represent respectively the number of trials needed by each algorithm (the NAC on the lefthand side and the extended Taguchi method on the righthand side) for generating a satisfying trajectory to reach the goal. The color of the spheres vary with the number of trials.

4.8 Experiments

In order to be able to compare Taguchi and NAC, we have implemented the two methods on a given example.

4.8.1 Comparison of Taguchi and NAC

The extended Taguchi method and NAC have been implemented on the system presented in Section 4.2, the modulation of the DS at the acceleration level. The parameters learned by the two different methods are the Gaussian centers of the GMM that are used to modelize the desired speed learned by demonstration $\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}$. Again, the position of the box and of the obstacle are fixed and only the starting positions vary. The box is located about 7cm in front of the robot torso. The movement are performed in simulation using the right arm of the HOAP 3 robot. 60 starting positions have been chosen in the workspace of the robot. 10 trajectories have been ran for each starting position and each algorithm. Another important condition of this experiment is that we have trained the GMM only once, it means that the GMM before the self-learning phase is the same for each trial. As the extended Taguchi method does not use any stochastic method for exploring the space of solutions, the final solutions is deterministic for a given starting position, thus, it is sufficient to run this algorithm only one time for each of the starting positions.

Figure 4.23 presents the results obtained in terms of number of trials needed for finding a satisfying solution. The total mean of the number of trials for each algorithm is 1556 trials for the NAC and 1716 trials for the Taguchi method. Concerning results obtained with Taguchi, there are two small remarks to do,

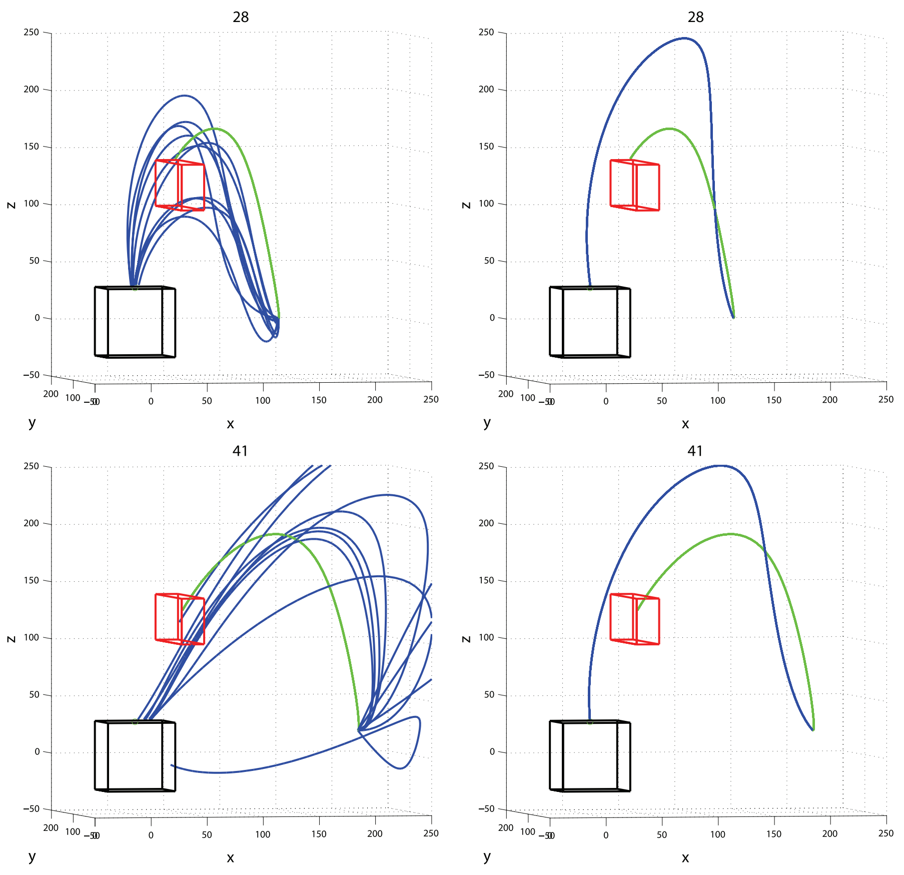

Figure 4.24: This 4 diagramss shows the solution found by the NAC on the left and by the Taguchi method on the right for respectively position number 28 and 41 (see Figure 4.23). The green Trajectory represent the trajectory before the learning.

first, the mean is done on 60 trials only. Due to the fact that the results are deterministic, it is a non-sense to perform more than one run for each starting position. Second, here the number of trials means the total number of trajectories generated in order to find the optimal solution. As we use in this example an orthogonal array of 54 rows, we have 54 trials per iteration. This allow us to better compare the two algorithms.

By comparing the two diagrams, even if the mean number of trials is smaller for the NAC, it seems that the dependence to the starting position is bigger than for the Taguchi method and that we have more disparity in the results. The standard deviation confirms that. The standard deviation for the NAC is of 1801 trials while we have 592 trials for the extended Taguchi method. In fact, as explained in Section 4.6 there is still a small number of cases where it is difficult to find a good solution for the NAC. Here, we can identify two part of the workspace where the NAC is seems to have some difficulties. The portion of the workspace that is the most distant from the target and a small portion right under the obstacle.

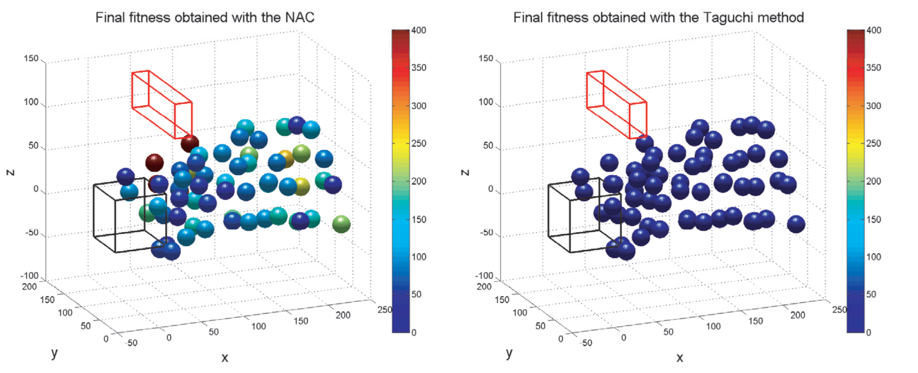

Figure 4.25: These diagrams represents the statistics done in order to compare the NAC and the extended Taguchi method on the same setup. These experiments were done on the system described in Section 4.2 and the learned parameters were the Gaussian centers of the GMM that modelized the desired speed $\boldsymbol{\mu}_{k,\dot{\boldsymbol{\xi}}}$ used to modulate the DS. The two graphics represent respectively the value of the fitness obtained for the final trajectory by each algorithm (the NAC on the left and the extended Taguchi method on the right). The absolute value of the reward function used for the NAC (see Section 4.5.2) is equal to the fitness function described in Section 4.7.1.

Figure 4.24 shows the results for the trajectories in one unfavorable position (position number 41) and one favorable position (position number 28). The results for the same positions with the Taguchi method are presented on the right, the solution in the two cases is proper and avoid the obstacle without any problems. Concerning the NAC, we can see here that at least one solution did not reach the goal for position number 41. As the maximum number of trials allowed is 10'000, this increases considerably the mean and the standard deviation for this position.

Figure 4.25 presents the same type of graphics than in Figure 4.23, but this time, the information given by the size and color of the sphere is an information about the reward obtained for the final trajectory of each algorithm. Here, the value of the reward is defined as the absolute value of the reward function described in Section 4.5. This value is equivalent to the Fitness used for the extended Taguchi method presented in Section 4.7.

The first thing to observe here is the great difference between the performance of the two algorithm. The mean of the final fitness over the different starting positions is 14 with the extended Tagushi method while we have a mean of 142 for the NAC. Moreover, the standard deviation for the extended Tagushi method is 2,7 while the standard deviation for the reward obtained by the NAC is 194.

If the difference between the two algorithms is not so big in terms of convergence time, in terms of rewards, the extended Taguchi method gives much better results. One explanation for such a big difference is the two different ways of exploring the solution space. By using the Taguchi method, we cover the complete chosen range for each parameter, in this example, the acceleration. As we know the specification of the robot, we are able to cover fully the space of solutions.

By using the NAC, we follow a gradient that is computed from local data. If the noise introduced during the exploration of the solution space is not big enough, we will not be able to guaranty that the optimum obtained is a global minimum.

By observing the result of this experiment, the extended Taguchi method seems to be quite efficient compared to the NAC. Here, one big advantage for the extended Taguchi method is that the system is well defined, and it is easy to fix the boundaries for each parameter we want to learn, we can imagine that the situation can be different if the system is not known apriori. It is not possible to explore the solution out of the boundaries with this algorithm. It is not the case with the NAC were no range have to be fixed apriori.

4.9 Summary

We have presented in this Chapter an improved algorithm for learning and reproducing goal-directed tasks with a humanoid robot. First the system attempts to reproduce the task by generating a trajectory using a DS modulated by a velocity profile generated by a GMM trained with a few demonstrations performed by the human user. This algorithm is already robust enough to handle perturbation in the goal position or in the starting position of the robot's arm, but if some perturbations occurred that can not be handled by the DS alone, a RL method can be used in order to allow the robot to relearn its own model of the task and adapt it to the new situation to find another smooth solution.

The system has been extensively tested on a task which consists in putting an object into a box. Two different algorithms have been tested for the self/learning module used to adapt the model of the task. The first algorithm is a RL algorithm, the NAC, while the second one is an operational research algorithm, the extended Taguchi method. For this specific problem, each algorithm has its own advantages and disadvantages. While it is quite difficult to tune the parameters of the NAC to have a smooth and stable convergence, the extended Taguchi method is entirely dependent on the bounds we fixed for the parameters to optimize. In the specific example of the box task, the extended Taguchi method gives better results in terms of final reward function for similar results in terms of convergence time, thus the extended Taguchi method seems to be a good choice for this task. This is mainly due to the fact that the bounds are given by the speed limits of the robot's joint angles, the system is thus well known, if the system is not known apriori, it can be much more difficult to set the bounds.

5 Potential Fields

5.1 Outline

In Chapter 4, we have presented a system to imitate reaching movements. This system can be divided in three parts. The first part consists in a DS that generates a trajectory to bring the robot's arm smoothly to the target. The second part consists in a speed profile generated from the demonstrations and used to modulate the signal generated by the DS. Finally, the last part consists in a trial-and-error learning algorithm that allows the robot to optimize its model of the task and thus, trough the modulation speed profile, to optimize its reproduction of the task.

However, some problems persist with this system. The first problem is the fact that we try to combine a signal of defined length (the modulation $\dot{\boldsymbol{\xi}}^m$) with a signal of unknown length (the signal given by the DS $\dot{\boldsymbol{\xi}}$). In that way, the modulation can not influence the DS on the whole trajectory, but only at the beginning. As we are only able to learn or relearn the modulation trajectory $\boldsymbol{\xi}^m$, it is then not possible to learn a new path near the target at the end of the trajectory.

The second problem is more related to the solution chosen for avoiding a possible obstacle. In fact, in the system described in Chapter 4, the robot learns a new trajectory that implicitly avoids the obstacle, but does not learn an explicit relation between the trajectory of the end-effector and the obstacle. It means, that the system learns how to avoid an obstacle at a given location, but if the location of the obstacle changes, the learning has to be performed again.

The idea behind the development of the systems presented in this Chapter is to propose a more coherent system which tends to solve these two problems. The first point is then to find a way to describe the relation between an unexpected obstacle and the end-effector movement. As the DS defines an attractor on the target, the more logical idea is to define a repulsor on the obstacle. By considering the attractor (the target) and the repulsor (the obstacle) as objects that exert a force on the end-effector, we have now to work at the acceleration level in order to work with value that are proportional to those forces.

Now, as we have defined a force that brings the end-effector to the target and a force that brings it away from the obstacle, the last thing to do in order to integrate the informations accumulated during the demonstrations in the system is to define a force that brings the end-effector on a trajectory that has been learned by demonstration.

Section 5.2 will introduce the algorithm inspired by the potential field method. The three different acceleration components used to compute the resulting trajectory will be presented in Section 5.2.1, Section 5.2.2 and Section 5.2.3.

Section 5.3 will present the experiments done in order to test the system. A first experiment is performed in order to test the system extensively in the workspace of the robot. For this experiment, the parameters of the repulsive Gaussian centered on the obstacle are set experimentally. A second experiment has been conducted in order to evaluate the possibility for the robot to learn the parameters of the repulsive Gaussian centered on the obstacle.

Section 5.4 proposes an idea in order to have a more coherent algorithm in which we do not have the problem of the modulation trajectory defined for a given number of steps. This proposition consists in learning directly the dynamic of the task. In other words, to modelize the acceleration of the end-effector of the robot given the current position and speed.

Section 5.5 will present some preliminary results in order to give a small idea of what we can expect with this type of algorithm.

Finally, a small summary will close the Chapter in Section 5.6.

5.2 Acceleration modulation of the DS

As introduced in 5.1, in this chapter, the goal is to develop an algorithm in which we have an explicit relation between a potential obstacle and the end-effector. Here, we will take inspiration on the potential field method. The potential field method is a classical and attractive path-planning method for this type of problem. Potential fields methods are popular thanks to their simplicity and low computational costs. Early work on potential fields method for obstacle avoidance were conducted by Khatib (Khatib, 1986). Since then several versions have been studied to avoid local minima problems and to adapt the method to more complex environments (Borenstein & Koren, 1989; Arkin, 1989; Warren, 1989; Ko & Lee, 1996; Wang & Chirikjian, 2000). In our work, as we have a simple workspace, we have decided to take inspiration on the potential field method to generate dynamically a trajectory for the reproduction of the learned task. Thus, we work now in acceleration and the desired acceleration $\ddot{\boldsymbol{\xi}}^d$ used as command for the robot is the weighted sum of three components:

$$\ddot{\boldsymbol{\xi}}^d(t) = \alpha_1 \ddot{\boldsymbol{\xi}}^{DS}(t) + \alpha_2 \ddot{\boldsymbol{\xi}}^{A}(t) + \alpha_3 \ddot{\boldsymbol{\xi}}^{R}(t) \tag{5.1}$$

Where $\alpha_1 + \alpha_2 + \alpha_3 = 1$ are weighting constants, $\ddot{\boldsymbol{\xi}}^{DS}(t)$ is the acceleration component due to the VITE (Bullock & Grossberg, 1988) inspired DS centered on the target, $\ddot{\boldsymbol{\xi}}^{A}(t)$ is the component due to the DS centered on the learned by demonstration trajectory $\boldsymbol{\xi}^m(t)$ and finally $\ddot{\boldsymbol{\xi}}^R(t)$ is the repulsive component due to the obstacle.

5.2.1 The DS component

The component given by the VITE (Bullock & Grossberg, 1988) inspired DS consists in the acceleration $\ddot{\boldsymbol{\xi}}^{DS}$ that tends to bring the end-effector to the final target of the reaching movement. This acceleration is given by the Equation 4.2 already described in Section 4.4. To recall, $\ddot{\boldsymbol{\xi}}^{DS}(t)$ is given by:

$$\ddot{\boldsymbol{\xi}}^{DS}(t) \quad = \quad \alpha(-\dot{\boldsymbol{\xi}}^s(t) + \beta(\boldsymbol{\xi}^g - \boldsymbol{\xi}^s(t))) \tag{5.2}$$

Where the constants $\alpha, \beta \in \mathbb{R}_{[0,1]}$ are control parameters, $\dot{\boldsymbol{\xi}}^s$ and $\boldsymbol{\xi}^s$ are the current speed and position of the simulated movement of the robot and $\boldsymbol{\xi}^g$ represents the target position (the goal). Usually, the goal is determined by using stereo vision.

5.2.2 The learned trajectory component

In order to integrate the movement learned during the demonstrations provided by the human user into the new system, one simple solution is to generate a trajectory $\boldsymbol{\xi}^m$ using the position data retrieved by GMR and to define a DS with a moving attractor following this trajectory step by step. At the end of the modulation trajectory $\boldsymbol{\xi}^m$, if the time T^g needed to reach the goal is bigger than the number of time steps T of $\boldsymbol{\xi}^m$ ($T^g > T$), then, the attractor simply stops on the last point.

As the attractor stays at the last point of the learned trajectory for $t > T$ (where t is the current time step), we have to ensure that the trajectory retrieved by GMR does reach the target $\boldsymbol{\xi}^g$ of the DS. Otherwise, the end-effector may be attracted by two different points in space, the target $\boldsymbol{\xi}^g$ of the DS and the end point of the retrieved trajectory $\boldsymbol{\xi}^m(T)$. Thus, the end-effector may never reach the main target $\boldsymbol{\xi}^g$.

The technique chosen to solve this problem consists in collecting data in 2 referential spaces during the demonstration phase. One referential space is centered on the starting position of the robot's end-effector and the other is centered on the target of the reaching movement.

In Chapter 4, only one referential space centered on the robot has been used and the data collected during the demonstration were speed data. This implies that for the box task, if we move constantly the box during the demonstrations, we are not able to learn an explicit relationship between the movement of the end-effector and the box. The only constraint that we are able to extract is a vertical movement going up and down without any displacement in the horizontal plan.

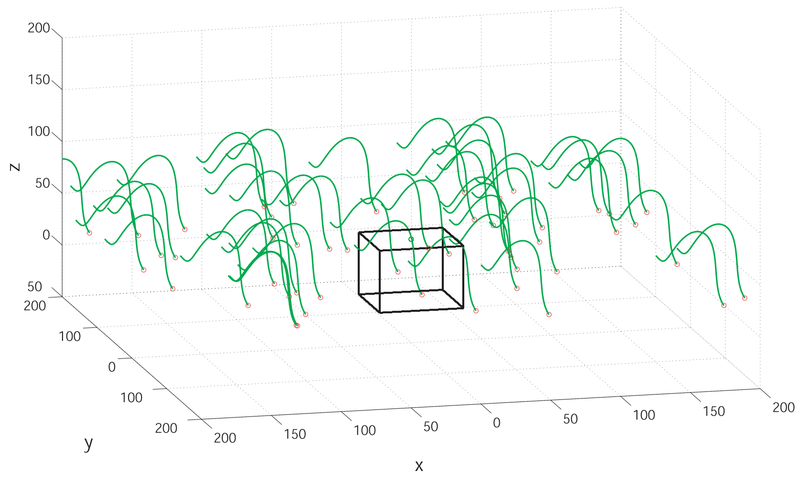

Figure 5.1: This diagram represents trajectories generated by the system described in Section 4.2. The red circles represent the starting positions and the green circle, the target above the box. The data encoded here are the speed profiles of the demonstrations in the robot's referential. Thus, during the learning, no informations are learned on the position of the target. That is why we generate here the same trajectory for each starting position. In the original system, this trajectory is used to modulate the DS and it is the DS that provide the attraction to the target.

Figure 5.1 represents several trajectories retrieved by GMR using the speed data and only one referential. Here, we can observe a vertical displacement and an horizontal one. The horizontal displacement comes from the fact that more demonstrations show a movement from the right to left. A similar case can be observed in the demonstrations shown by Figure 4.10 with the difference that in Figure 4.10, the location of the box is the same for each demonstration.

By using two referentials centered on the box and on the starting position respectively, we are able to learn a relationship between the box and the starting position. In the referential of the box, all demonstration trajectories end at the same point right above the box, thus the end the resulting trajectory is very constrained in comparison to the beginning. In the second referential centered on the starting position, the beginning of the resulting trajectory is very constrained. If the demonstrations vary sufficiently, by combining the two referentials, we are then able to generate a trajectory $\boldsymbol{\xi}^m$ that begins at the starting position and reaches the goal. For that, we modelize the two joint probabilities $p(t, \boldsymbol{\xi}^{(1)})$ and $p(t, \boldsymbol{\xi}^{(2)})$ for respectively referential (1) and (2). Then, by estimating $p(\boldsymbol{\xi}|t) = p(\boldsymbol{\xi}^{(1)}|t) \cdot p(\boldsymbol{\xi}^{(2)}|t)$, we can compute $\boldsymbol{\xi}^m = \mathbb{E}[p(\boldsymbol{\xi}|t)]$ that provide us the trajectory for the moving attractor that is used to generate $\ddot{\boldsymbol{\xi}}^A(t)$.

Figure 5.2 represents a sample of the trajectories generated with this system for a fixed box and a random starting position. Even if we observe here that each starting position (represented by a green circle) and first points of each correspondent trajectory are sometimes not exactly at the same position, each

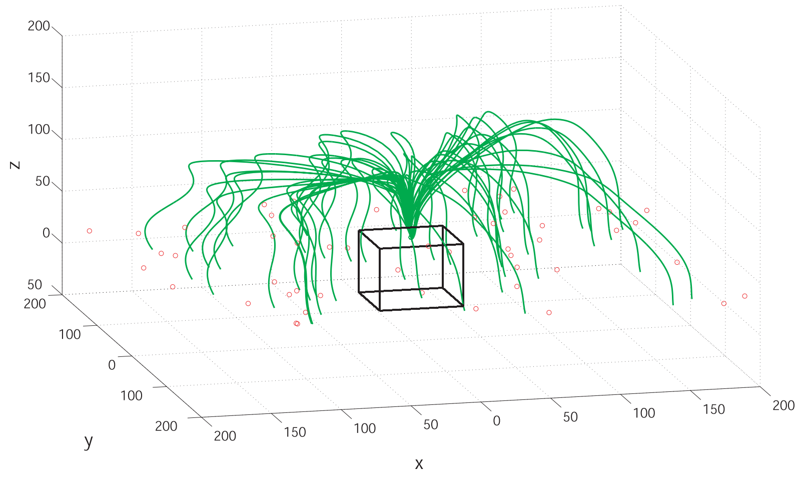

Figure 5.2: This diagram represents trajectories generated by GMR with two different referentials for random starting positions. One referential is the referential of the object and the other is the referential of the starting position of the end-effector. The system is described in Section 5.2.2. The red dots represents the starting positions and the green dot the target upward the box. Compared to Figure 5.1, here the trajectories reach the goal.

trajectory reaches the goal. The differences between the locations of the starting points and the first point of each corespondent trajectory is probably due to the variance in the two referentials. At the beginning of the trajectory, even if the constraint in the referential of the box are smaller, the influence on the final trajectory is not negligible. In the other side, since we reach the target for each trajectory, it seems that the influence of the constraint in the starting position referential are negligible at the end of the final trajectory. This, however, depends only on the demonstrations.

By using this system, we can now generate $\ddot{\boldsymbol{\xi}}^A(t)$. For that, we use a DS with an attractor that follow the resulting trajectory $\boldsymbol{\xi}^m(t)$ step by step:

$$\ddot{\boldsymbol{\xi}}^A(t) = \alpha_A(-\dot{\boldsymbol{\xi}}(t) + \beta_A(\tilde{\boldsymbol{\xi}}^m(t) - \boldsymbol{\xi}(t))) \tag{5.3}$$

Where $\boldsymbol{\xi}(t)$ and $\dot{\boldsymbol{\xi}}(t)$ are the current position and speed of the end-effector and $\tilde{\boldsymbol{\xi}}^m(t)$ is given by:

$$\tilde{\boldsymbol{\xi}}^m(t) = \{ \begin{array}{ll} \boldsymbol{\xi}^m(t) & \text{if } t \leq T \\ \boldsymbol{\xi}^m(T) & \text{if } t > T \end{array} \tag{5.4}$$

5.2.3 The repulsive force component

In order to have an explicit influence of the obstacle on the trajectory, we compute a repulsive force that brings the end-effector away from it. This repulsive

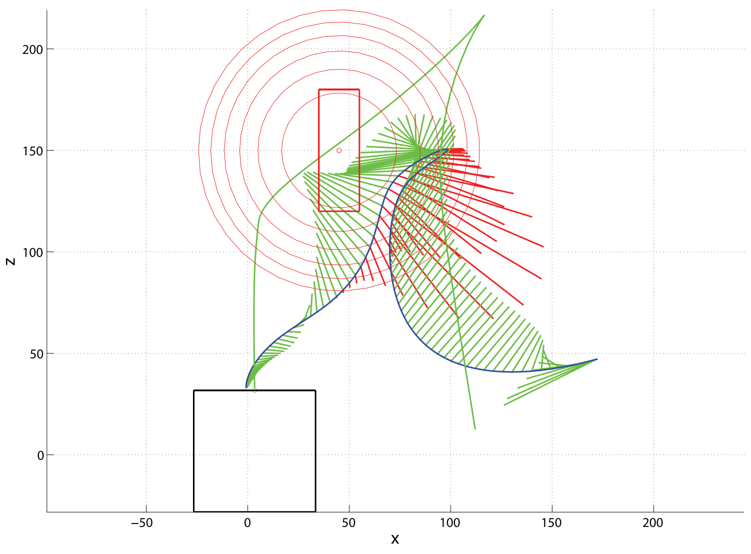

Figure 5.3: This diagram represents a 2D example of the complete system presented in Section 5.2. The green trajectory is the modulation trajectory $\boldsymbol{\xi}^m(t)$ and the blue trajectory represent the desired trajectory $\boldsymbol{\xi}^d(t)$. The green lines represent the influence of the modulation $\ddot{\boldsymbol{\xi}}^A(t)$ at each time step t while the influence of the obstacle is represented by the red lines. The repulsive Gaussian centered on the obstacle is represented by the red circles.

force acts on the end-effector as an acceleration component $\ddot{\boldsymbol{\xi}}^R(t)$ defined by a 3D Gaussian centered on the obstacle as follows:

$$\ddot{\boldsymbol{\xi}}^R(t) = A_R \frac{1}{\sqrt{(2\pi)^3 * |\Sigma_R|}} e^{(-\frac{1}{2}(\ddot{\boldsymbol{\xi}}-\boldsymbol{\mu}_R)^T \boldsymbol{\Sigma}_R^{-1}(\ddot{\boldsymbol{\xi}}-\boldsymbol{\mu}_R))} \tag{5.5}$$

Where A_R is a coefficient used to control the amplitude of $\ddot{\boldsymbol{\xi}}^R(t)$ and $\boldsymbol{\mu}_R$ and $\boldsymbol{\Sigma}_R$ are respectively the mean and the variance of the repulsive Gaussian. μ_R is the center of the obstacle and Σ_R depends on the volume of the obstacle and defines the range around the obstacle where the repulsive force is active (non negligible).

Figure 5.3 represents an example of the system's reaction when an obstacle blocks the desired trajectory. In this example, we do not have any learning. The blue line is the desired trajectory reproduced by the system while the green line represents the modulation trajectory $\boldsymbol{\xi}^m(t)$ which is used to generate the acceleration $\ddot{\boldsymbol{\xi}}^A(t)$ represented here by the green straight lines. The length of these lines are proportional to the acceleration $\ddot{\boldsymbol{\xi}}^A(t)$. The repulsive acceleration $\ddot{\boldsymbol{\xi}}^R(t)$ generated by the obstacle at each time step is represented by the red straight lines. We can also observe on this diagram the Gaussian centered on the obstacle and represented by red circles. Here, values of the coefficient A_R,

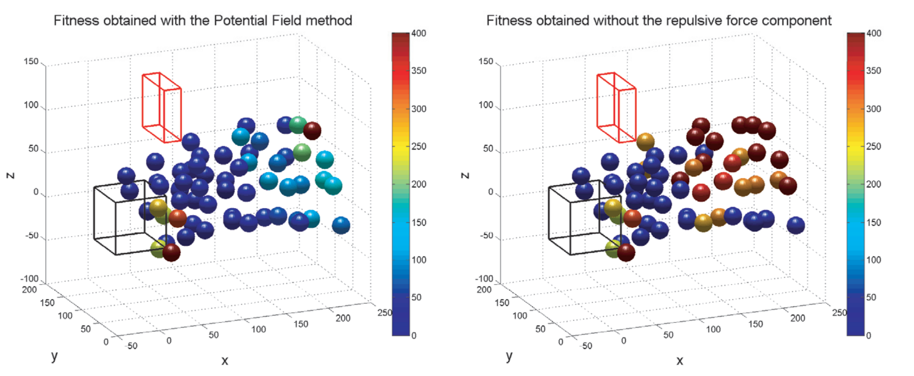

Figure 5.4: On the left, the diagram represents the statistics for the system presented in Section 5.2. Several trajectories have been generated from different starting positions in the workspace of the robot. The color of the spheres placed on each starting position represents the value of the mean fitness obtained for the corresponding trajectories. On the right, the diagram represents the same statistics for trajectories generated using the same system without the repulsive force generated by the obstacle. For the two diagrams, the fitness function used is the one defined in Equation 4.41 and used in the experiments presented in Section 4.8.

of the mean $\boldsymbol{\mu}_R$ and of the variance $\boldsymbol{\Sigma}_R$ are set empirically. We can see that the variance is isometric, but we can imagine also to have a asymmetric variance that vary following the obstacle shape.

5.3 Experiments and results

5.3.1 The potential field method

To test this new system, we have used the same task as in Chapter 4. The demonstration dataset used is the same as the one used in Section 4.6. As in Section 4.8.1, we have ran several trajectories from different starting positions in the workspace. The position of the box and the position of the obstacle remain the same for the different trials. The main advantage expected from the potential field method used here is to generate a trajectory that avoid the obstacle properly. In order to be able to evaluate the capacity of this algorithm to avoid the obstacle, a second experiment has been ran without the repulsive component $\ddot{\boldsymbol{\xi}}^R(t)$ generated by the Gaussian centered on the obstacle. To compare the results of the two experiments, we have used the fitness function 4.41 defined in Section 4.7.1. Note that in this experiments, the value of the coefficient A_R, of the mean $\boldsymbol{\mu}_R$ and of the variance $\boldsymbol{\Sigma}_R$ have been set empirically. Results are presented in Figure 5.4.

In Figure 5.4, a sphere is centered on each starting position. The color of the spheres represent the mean of the fitness obtained for the trajectories generating for each corresponding starting position. On the left, the diagram presents the results of the experiment in terms of fitness with the repulsive force component

$\ddot{\boldsymbol{\xi}}^R(t)$ while the diagram on the right represents the results without the repulsive force component $\ddot{\boldsymbol{\xi}}^R(t)$.

If we compare the two diagrams, we can observe that the difference in terms of fitness is not so big for the starting positions located near the obstacle. Bigger differences can be observed in the upper right part of the explored workspace. In fact, the differences between these two diagrams gives us an indication on the starting positions for which this specific location of obstacle is problematic. When the difference is small, the influence of the obstacle on the reproduction is small and when we have a bigger difference, it means that the system is not able to find a trajectory that reach the goal without using the repulsive force component. Thus, by observing the diagram on the right, we can clearly see the advantages of using the potential field method.

If we compare these two diagrams with the diagrams of Figure 4.25 in Section 4.8.1, we can see that, in terms of Fitness, we have a system which is comparable to the system described in Chapter 4 using the NAC for the RL module, here we have a mean fitness of 139.5 (against 142 for the NAC). The big advantage here is that we do not need to relearn the trajectory to avoid the obstacle.

In order to have a idea of the trajectories generated by this system, we have plot them for four different starting positions out of the sixty positions used for the experiment. Figure 5.5 represents those examples of trajectories. The four starting positions are presented in Figure 5.4. In these diagrams, the green trajectories represent the trajectories $\boldsymbol{\xi}^m(t)$ generated using GMR and used as attractor to modulate the main DS. For each run of the algorithm, we have trained a new GMM. As the initial conditions used for the clustering of the data are defined stochastically, we have different modulation trajectories for each starting position. The trajectories generated by the system are represented by the blue lines. The red lines represent the trajectories generated by the system without the repulsive acceleration component $\ddot{\boldsymbol{\xi}}^R(t)$.

Positions number 08 and 11 are two positions located in the lower part of the tested workspace. We can see that for these two starting positions, at least one generated trajectory (two for position 08) is blocked if we do not use the repulsive component $\ddot{\boldsymbol{\xi}}^R(t)$ to avoid the obstacle (see the red lines). For the two other examples, number 46 and 57, the corresponding starting positions are located higher in the workspace. In these two examples, the algorithm seems to have more difficulties to find a suitable solution. In the two diagrams, at least one trajectory generated with the repulsive component $\ddot{\boldsymbol{\xi}}^R(t)$ hits the obstacle and does not reach the target.

The problem here is that if we put the center of the repulsive Gaussian in the center of the obstacle, it may happen that the repulsive component $\ddot{\boldsymbol{\xi}}^R(t)$ is directly opposed to the displacement of the end-effector $\dot{\boldsymbol{\xi}}^d(t)$. In other word, the perpendicular component of $\ddot{\boldsymbol{\xi}}^R(t)$ relatively to the speed vector of the reproduced trajectory $\dot{\boldsymbol{\xi}}^d(t)$ is near zero and thus, $\dot{\boldsymbol{\xi}}^d(t)$ is decreasing, but the trajectory $\boldsymbol{\xi}^d(t)$ is not deviate before hitting the obstacle. This situation can be

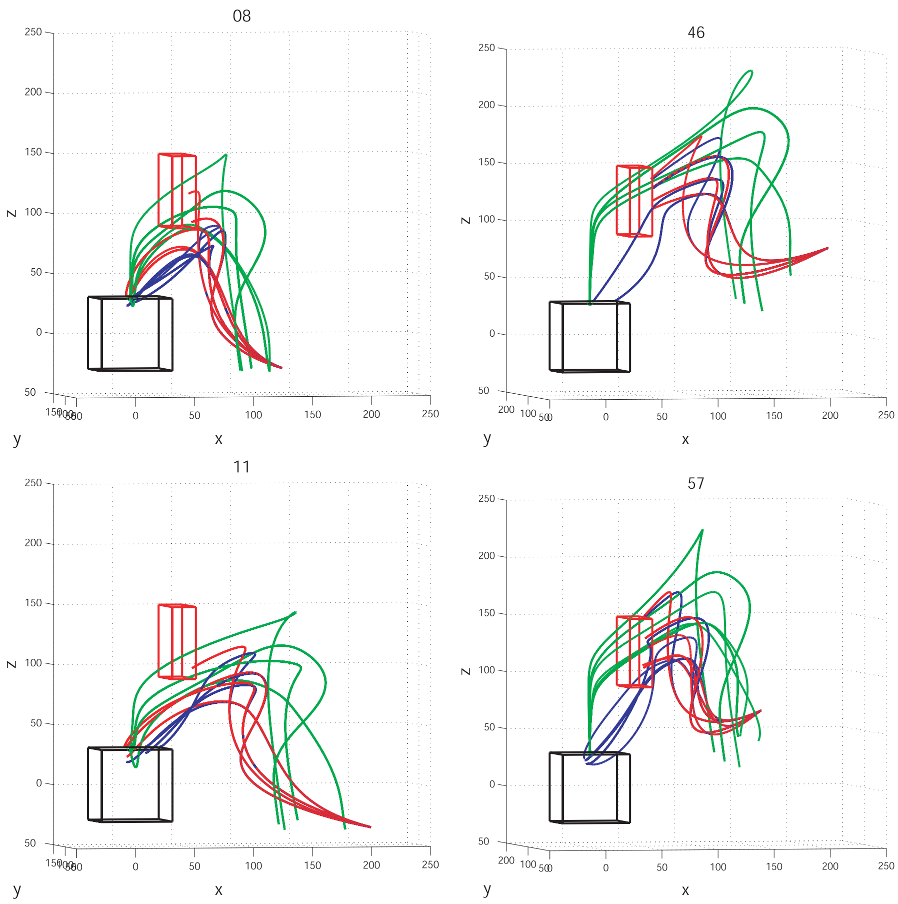

Figure 5.5: These diagrams represent the trajectories generated by the system presented in Section 5.2 for the four different starting positions indicated on Figure 5.4. The green lines represent the modulation trajectories $\boldsymbol{\xi}^m(t)$ used to generate the acceleration components $\ddot{\boldsymbol{\xi}}^A(t)$ that represent the input of the human user in the system. The blue lines represent the reproduction generated by the system while the red line represent the reproduction without the repulsive component.

observed two times for starting position number 46 and one times for starting position number 57.

In these examples, it is also interesting to observe the modulation trajectory $\boldsymbol{\xi}^m(t)$. Here again, the first point of the modulation trajectories does not coincide with the starting position. As explained in Section 5.2.2, this is due to the combination of the two trajectories in different referentials. At the beginning of the trajectory, even if the constraint given by the trajectory of the obstacle referential is smaller than the constraint given by the trajectory in the starting point referential, it has an influence on the final trajectory that will bring the desired starting point away from the current starting point of the end-effector.

As presented in the Outline 5.1 of this Chapter, there was two main problems that we wanted to solve by designing this algorithm. The first problem was to have a system where we can have an explicit relation between the reproduced trajectory and a possible obstacle. The second point was to be able to avoid an obstacle even if this obstacle is located near the target. With the system presented in Chapter 4, the function $\gamma(t)$ allow the modulation only for the first part of the trajectory. With this system, we do not have such a function and the modulation is also possible near the obstacle. Thus, even if the obstacle is near the target, the system is able to avoid it. In Figure 5.6, we have an example of trajectory generated with two obstacles on the way. Again, the green trajectory represents the modulation trajectory generated by the GMR system presented in Section 5.2.2. The green lines are the acceleration components $\ddot{\boldsymbol{\xi}}^A(t)$ at each time step produced by the DS centered on $\boldsymbol{\xi}^m(t)$. The red lines are the repulsive components $\ddot{\boldsymbol{\xi}}^R(t)$ resulting from the obstacles as described in Section 5.2.3. And finally, the blue trajectory is the trajectory generated by the complete system $\boldsymbol{\xi}^d(t)$. We can observe that one of the two obstacle is located near the target and does not block the trajectory. In fact, the only problem here when the obstacle is too close from the target can be that the repulsive component $\ddot{\boldsymbol{\xi}}^R(t)$ prevents the robot to reach the target.

5.3.2 Learning the Repulsive Force

In the precedent experiment, the parameters of the repulsive Gaussian centered on the obstacle have been defined empirically. For the robustness of the algorithm, it will be much more consistent to have a system that is able to learn these parameters. If we provide the system with such a capacity, we allow the robot to learn the influences that the objects present in the set-up have on the end-effector trajectory independently from their size or shape.

As presented in Section 5.2.3, the Gaussian centered on each possible obstacle is defined by its center $\boldsymbol{\mu}_R$, its variance $\boldsymbol{\Sigma}_R$ and one additional coefficient A_R that is used to tune the strength of the Gaussian. As the Gaussian is centered on the obstacle, we will here learn the variance $\boldsymbol{\Sigma}_R$ and the coefficient A_R. The algorithm used for the learning is the extended Taguchi method. This method

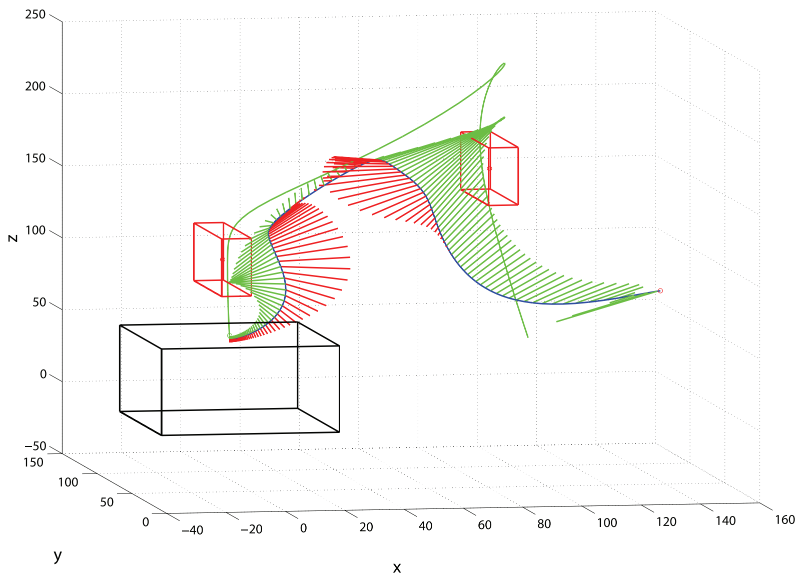

Figure 5.6: This Diagram represents a trajectory generated using the system described in Section 5.2. In green, we have the trajectory retrieved by GMR, in blue, the final trajectory generated by the system, the green lines represent the components $\ddot{\boldsymbol{\xi}}^A(t)$ of the system at each time step and the red lines represent the components $\ddot{\boldsymbol{\xi}}^R(t)$ of the system at each time step. The box (the target) is in black and two obstacles were added (in red) in order to emphasize the influence of this kind of repulsor on the resulting trajectory.

has been chosen for the easiness of implementation and the good result obtained in the experiment presented in Section 4.8. The fitness function used to evaluate the trajectories has been defined in Equation 4.41.

These experiments have been conducted in 2D. The demonstration data used to train the GMM of each referential have been projected in 2D. Moreover, the GMM have been trained only once at the beginning of the experiment and we use the same model for each run. A couple of examples are presented in Figure 5.7. On the left, we have the version of the trajectory generated by the system with the default parameters for the repulsive Gaussian. These parameters have been set empirically and are the same than those used in Section 5.3.1. On the right, the parameters have been learned using the extended Taguchi method. The learning has been done specifically for each starting position.

In Figure 5.7, by comparing the two diagrams for each position, we can observe clearly the improvements brought by the learning of the repulsive Gaussian parameters. However, considering the differences in the shape of the Gaussian across the different starting positions, we can also observe one drawback of this type of learning. If the learning is done for a specific starting position, the repulsive Gaussian is modified in order to have a optimal resulting trajectory $\boldsymbol{\xi}^d(t)$ for this specific setup, but this solution can be far from the optimal solution for other setups.

Figure 5.8 represents two examples of setup for starting positions 02 and 03 of Figure 5.7. Here, the repulsive Gaussian parameters have been learned specifically for starting position number 01. We can observe that for trajectory number 02, even if the shape of the Gaussian is different than in Figure 5.7, the resulting trajectory is very similar. This come from the fact that the coefficient A_R learned by the system for position 01 is very small, thus the influence of the Gaussian modelized by the acceleration $\ddot{\boldsymbol{\xi}}^R(t)$ is near zero even if the current point $\boldsymbol{\xi}^d(t)$ is very close to the obstacle. In this specific case, it is not a problem, as the generated trajectory does not hit the obstacle even without repulsive force. The problem appears in the starting position number 3. Here, the generated trajectory hits the obstacle and as $\ddot{\boldsymbol{\xi}}^R(t)$ is near zero, the repulsive force is not strong enough to allow the system to avoid the obstacle properly.

In Figure 5.9, the parameters of the repulsive Gaussian are learned specifically for the case number 03 of Figure 5.7. Here, we can observe that in the setup number 01, there is no problem. In fact, in cases number 01 and 03, the shape of the repulsive Gaussian is very similar, only the coefficient A_R is different, A_R is much bigger in case 03. In case number 02, we can see that the problem is not to hit the obstacle. Here, the trajectory is functional in the sense that we do not hit the obstacle and we reach the box from above. However, we can clearly observe a big deformation of the resulting trajectory compared to the optimal solution.

As conclusion, we can say that by learning the repulsive force generated by an obstacle, we are able to generate more proper trajectory in a given setup.

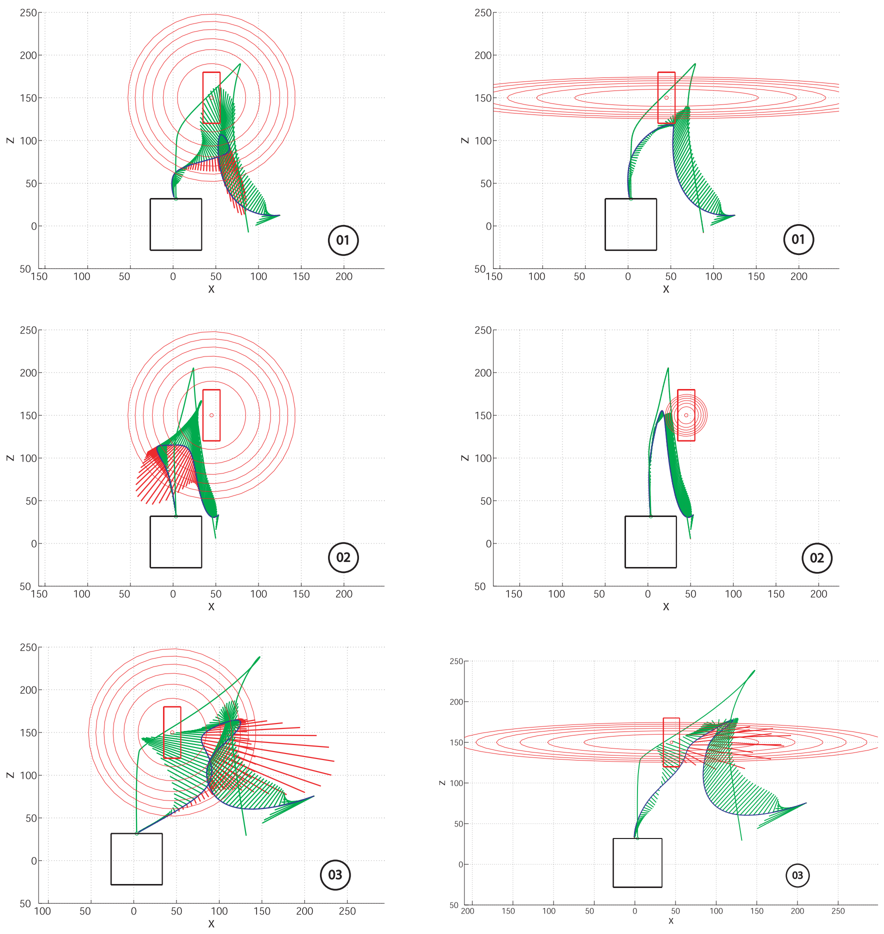

Figure 5.7: These diagrams represent two different situations for three selected starting positions. On the left, the diagrams represent the trajectory generated by the system using the default parameters of the repulsive Gaussian that have been defined empirically. On the right, the diagrams represent the trajectory generated by the system when the parameters of the repulsive Gaussian have been learned for each specific starting position. The green trajectory is the modulation trajectory $\boldsymbol{\xi}^m(t)$, the blue trajectory is the trajectory of the end effector $\boldsymbol{\xi}^d(t)$, the green and red lines are respectively the acceleration $\ddot{\boldsymbol{\xi}}^A(t)$ and $\ddot{\boldsymbol{\xi}}^R(t)$ and the red circles represent the shape of the repulsive Gaussian.

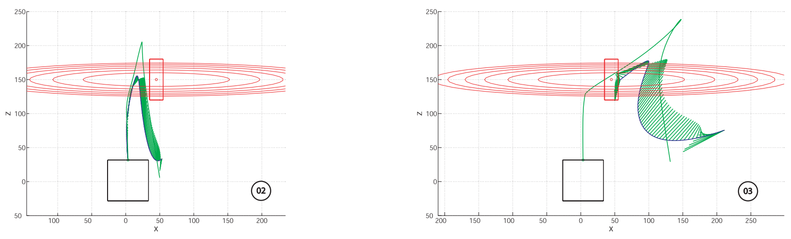

Figure 5.8: These diagrams represent two trajectories generated using the potential field method. Here, the parameters of the repulsive Gaussian have been learned in the setup number 01 of Figure 5.7. The two starting positions represented here are the two other starting position also represented in Figure 5.7. The green trajectory is the modulation trajectory $\boldsymbol{\xi}^m(t)$, the blue trajectory is the trajectory of the end effector $\boldsymbol{\xi}^d(t)$, the green and red lines are respectively the acceleration $\ddot{\boldsymbol{\xi}}^A(t)$ and $\ddot{\boldsymbol{\xi}}^R(t)$ and the red circles represent the shape of the repulsive Gaussian.

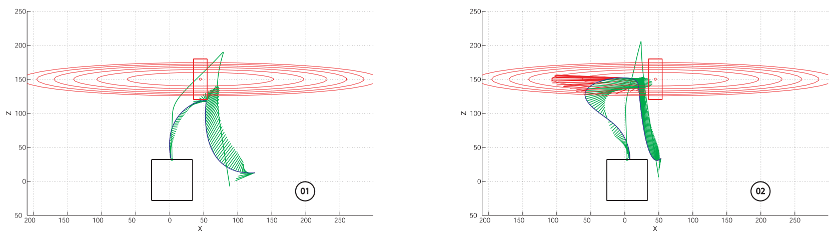

Figure 5.9: These diagrams represent two trajectories generated using the potential field method. Here, the parameters of the repulsive Gaussian have been learned in the setup number 03 of Figure 5.7. The two starting positions represented here are the two other starting positions also represented in Figure 5.7. The green trajectory is the modulation trajectory $\boldsymbol{\xi}^m(t)$, the blue trajectory is the trajectory of the end effector $\boldsymbol{\xi}^d(t)$, the green and red lines are respectively the acceleration $\ddot{\boldsymbol{\xi}}^A(t)$ and $\ddot{\boldsymbol{\xi}}^R(t)$ and the red circles represent the shape of the repulsive Gaussian.

However, the Gaussian that is learned for a specific case will not always work properly in other situations, thus, if we want to have a optimal reproduction in every situation, we should have a learning phase before each reproduction.

5.4 Learning of the DS

The system presented in 5.2 is very complicated and still presents some problems. The main problem is still related to the predefined time of the modulation trajectory. The problem is not elegantly solved here. For example, when the time to reach the target T^g is longer than the time of the modulation trajectory T, the end-effector is attracted at the same times by the end of the modulation trajectory through the acceleration $\ddot{\boldsymbol{\xi}}^A(t)$ and the target through the VITE inspired DS component $\ddot{\boldsymbol{\xi}}^{DS}(t)$. It may happen that the demonstrations are not good enough to retrieve a modulation trajectory that ends in the target. In these cases, the system may fail.

In order to come up with a more coherent system, one idea is to learn directly the dynamics of the demonstrations. By using the GMM (see 3.4.2), we can learn the acceleration of the end-effector knowing the position and the speed. The output variables are then the acceleration $\ddot{\boldsymbol{\xi}}$ and the input variable are the position $\boldsymbol{\xi}$ and speed $\dot{\boldsymbol{\xi}}$.

The mean vector μ_k and covariance matrix $\boldsymbol{\Sigma}_k$ are given by:

$$\begin{aligned} \boldsymbol{\mu}_k &= [\boldsymbol{\mu}_{k,\boldsymbol{\xi}\dot{\boldsymbol{\xi}}}^T \ \boldsymbol{\mu}_{k,\ddot{\boldsymbol{\xi}}}^T]^T & (5.6) \\ \boldsymbol{\Sigma}_k &= \begin{pmatrix} \boldsymbol{\Sigma}_{k,\boldsymbol{\xi}\dot{\boldsymbol{\xi}}} & \boldsymbol{\Sigma}_{k,\boldsymbol{\xi}\dot{\boldsymbol{\xi}}\ddot{\boldsymbol{\xi}}} \\ \boldsymbol{\Sigma}_{k,\ddot{\boldsymbol{\xi}}\dot{\boldsymbol{\xi}}\boldsymbol{\xi}} & \boldsymbol{\Sigma}_{k,\ddot{\boldsymbol{\xi}}} \end{pmatrix} & (5.7) \end{aligned}$$

And the acceleration is then given by:

$$\ddot{\boldsymbol{\xi}}(t) = \sum_{k=1}^{K} h_k(t)\big(\boldsymbol{\mu}_{k,\ddot{\boldsymbol{\xi}}} + \boldsymbol{\Sigma}_{k,\ddot{\boldsymbol{\xi}}\dot{\boldsymbol{\xi}}\boldsymbol{\xi}}\boldsymbol{\Sigma}_{k,\boldsymbol{\xi}\dot{\boldsymbol{\xi}}}^{-1}([\boldsymbol{\xi}^T \ \dot{\boldsymbol{\xi}}^T]^T - \boldsymbol{\mu}_{k,\boldsymbol{\xi}\dot{\boldsymbol{\xi}}})\big) \tag{5.8}$$

where the $h_k(t)$ are given by:

$$h_k(t) = \frac{\pi_k \mathcal{N}([\boldsymbol{\xi}^T \ \dot{\boldsymbol{\xi}}^T]^T; \mu_{k,\boldsymbol{\xi}\dot{\boldsymbol{\xi}}}, \boldsymbol{\Sigma}_{k,\boldsymbol{\xi}\dot{\boldsymbol{\xi}}})}{\sum_{k=1}^{K} \pi_k \mathcal{N}([\boldsymbol{\xi}^T \ \dot{\boldsymbol{\xi}}^T]^T; \mu_{k,\boldsymbol{\xi}\dot{\boldsymbol{\xi}}}, \boldsymbol{\Sigma}_{k,\boldsymbol{\xi}\dot{\boldsymbol{\xi}}})} \tag{5.9}$$

The variance of this conditional probability distribution is given by:

$$\boldsymbol{\Sigma}_{\ddot{\boldsymbol{\xi}}}(t) = \sum_{k=1}^{K} h_k^2(t)\big(\boldsymbol{\Sigma}_{k,\ddot{\boldsymbol{\xi}}} - \boldsymbol{\Sigma}_{k,\ddot{\boldsymbol{\xi}}\dot{\boldsymbol{\xi}}\boldsymbol{\xi}}\boldsymbol{\Sigma}_{k,\boldsymbol{\xi}\dot{\boldsymbol{\xi}}}^{-1}\boldsymbol{\Sigma}_{k,\boldsymbol{\xi}\dot{\boldsymbol{\xi}}\ddot{\boldsymbol{\xi}}}\big) \tag{5.10}$$

5.5 Experiments and preliminary results

In order to validate the system, we have tried to learn a simple VITE inspired DS (as described in Section 4.4). Demonstration data has been created using some random positions around a reference point as starting positions and a fixed target located at the origin of the referential frame.

Figure 5.10 represents the results of this experiment. The three upper diagrams represent the position, speed and acceleration for the original DS. The three lower diagrams represent the position, speed and acceleration for the trajectory retrieved by GMR. To build this trajectory, the acceleration at time t is given by GMR knowing the position and speed at time $t-1$. The trajectory is then built step by step. What we can observe here is that we have a kind of smoothing of the acceleration. Between time $t = 0$ and $t = 1$, for the original DS, we have a discontinuity, as the initial speed is equal to 0, the acceleration at time $t = 1$ depends only on the distance between the initial position and the target. In the GMM, this discontinuity is smoothed, however, if we except this, the two systems have very similar reactions. The convergence times are very similar. The question is now to know if it is also possible to learn the DS by using trial-and-error learning.

5.5.1 Learning the DS by using Taguchi

For this experiment, we will apply the Taguchi method on the system described in Section 5.4. The Taguchi method has been chosen instead of the NAC for its simplicity of implementation and the good results obtained (see Section 4.8). The parameters $a(n)$ that will be optimized are the centers of the Gaussian of the GMM output $\boldsymbol{\mu}_{k,\ddot{\boldsymbol{\xi}}}$. The $\boldsymbol{\mu}_{k,\ddot{\boldsymbol{\xi}}}$ needs to be bounded. As we work in the Cartesian space, we set the bounds to the minimum and maximum acceleration $\boldsymbol{\mu}^{min}_{k,\ddot{\boldsymbol{\xi}}}$ and $\boldsymbol{\mu}^{max}_{k,\ddot{\boldsymbol{\xi}}}$ allowed by the robot for its end-effector. Here, for security reasons, we will fix those bound to $1\frac{m}{s^2}$ which is a little lower than the effective acceleration allowed.

Figure 5.11 represents one example of the results we can expect by running Taguchi algorithm on the system described in section 5.4. The black line represents the trajectory obtained by the system after the use of the Taguchi method for optimizing the center of the GMM. We can observe that we obtain a smooth trajectory that reaches the goal without hitting the obstacle. Further work are conducted by David Koch and Seyed Mohammad Khansari Zadeh (Khansari Zadeh & Billard, 2009) for exploring these type of solution for the efficient reproduction of a learned task.

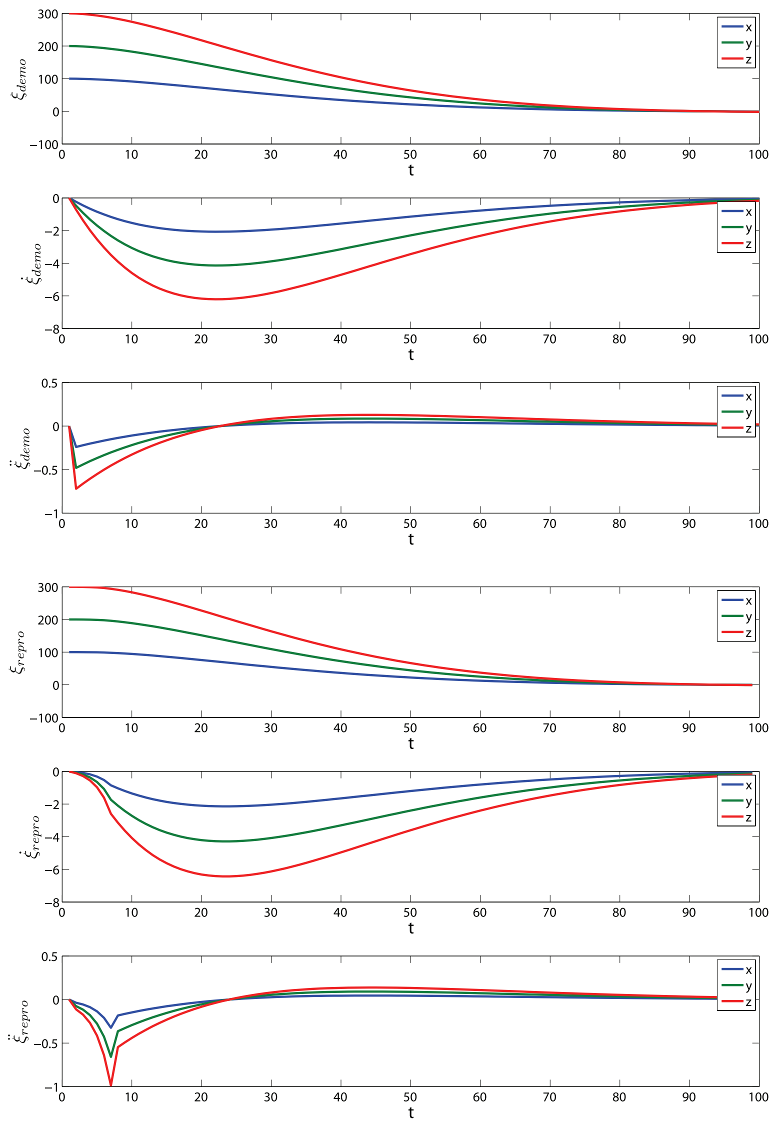

Figure 5.10: The first three diagrams represent respectively the position, speed and acceleration for a typical demonstration trajectory generated with a VITE inspired DS 4.2. The last three diagrams represent the position, speed and acceleration for a trajectory generated by a GMM trained using several examples of trajectories generated by the VITE system that retrieve the acceleration in function of the position and speed.

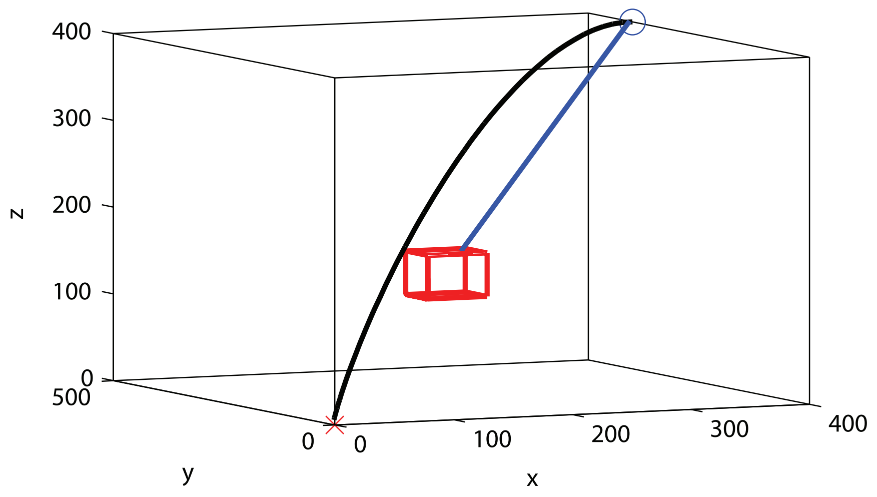

Figure 5.11: This Graphic represents the kind of results we can expect by running Taguchi method on the system described in Section 5.4. The blue line represents the trajectory originally reproduced by the learned DS. The black line represents the trajectory reproduced by the system after the optimization of the system using Taguchi.

5.6 Summary

We have presented in this chapter an algorithm to solve the problem of the reproduction of a goal-directed task learned by demonstration. Compared to the solution presented in Chapter 4, we keep the idea of having a DS which bring smoothly the arm of the robot to the target. The difference appears in the modulation of this DS. The modulation is done here at the acceleration level. As we work in acceleration, we can introduce the idea of forces that influence the trajectory generated by the DS, this forces intervene as different accelerations that influence the acceleration generated by the main DS. One force modulate the DS in order to tend to a trajectory learned using the demonstrations of the human user, and and the other one acts as a repulsive force that prevent the end-effector to hit an eventual obstacle.

The system has been tested in the same setup than in Chapter 4 in order to be able to compare the result to the result obtained with the precedent system. For this experiment, the box and the obstacle are in a fixed position and we test the algorithm with 60 different starting positions distributed in the workspace. The parameters of repulsive Gaussian centered on the obstacle have been defined empirically and are fixed for the whole experiment. It means that the trajectories generated during these experiment do not require any learning phase to avoid the obstacle. The result in terms of fitness (or reward) are comparable to the results obtained with the precedent system by using the

NAC algorithm to relearn the modulation trajectory, but with a much improved computation time as we do not have to perform any learning.

An second experiment has been conducted on the possibility to learn the parameters of the repulsive Gaussian in order to allow the robot to adapt its own model of the obstacle that can interfere in the scene. This experiment have shown the possibility to improve the resulting trajectory for a specific case, but the problem is that the optimum can change from one case to the other, if we choose to use a trial-and-error learning algorithm, we have to use it for every different cases and that increase considerably the computation time.

Finally, a idea for a further more coherent algorithm has been presented in the last Section. The idea is to modelize the dynamic of the task by using a GMM. With this system, we should be able to retrieve a acceleration for a given position and speed. Further work on these type of system are currently conducted at the LASA.

6 Discussion

In this Chapter, a couple of points relevant to this work will be discussed. Firstly, as presented in Chapters 2 and 3, the development of robots that are similar to human beings is a strong hypothesis for the resolution of the *correspondence problem* in this work. To what degree should the robot be similar to a human to avoid the drawbacks that an android robot can engender on the human user reaction. The second point will discuss our approach to solve the *correspondence problem* as presented in Chapter 3 by balancing the data of different frames of reference. The third point is a discussion about the restrictions due to the main hypothesis of imitating reaching movement in Chapter 4. The Fourth point will discuss the advantage and disadvantage of generating trajectories at the position level, the speed level or the acceleration level. Finally the last point will discuss the difference between optimization and learning techniques used to allow the robot to learn by trial-and-error its own model of the task.

6.1 Humanoids or androids?

There are a couple of good reasons to choose human morphology to develop robots intended to work and interact with people. From a technical point of view, humanoid robots help to reduce the complexity of transferring skills from a human teacher to a robot imitator. It also helps the user to have a robot with which he/she can identify. Thus the user can better interact with the robot since he can interpret its motion and infer its next action. The opposite is also true because if we suppose that the robot is able to detect human movements, it could also identify the motions and be able to infer next actions and thus better help its user.

Currently, robots, and more specifically humanoid robots, are still rarely in direct contact with unexperienced people and rarely leave the laboratories. A lot of research is done in order to study the potential acceptance of such robots in daily life. Science Fiction still commonly describes a world where humans created an artificial intelligence that surpasses human intelligence and leads to the extinction of the human race. In this type of scenario, robots usually play the role of interface between the virtual world and the real world and are commonly presented as humanoids or more particularly androids. Androids are humanoid robots that not only reproduce the kinematics of human beings but also the general aspect (skin, eyes, hair...). The ideal androids would not be

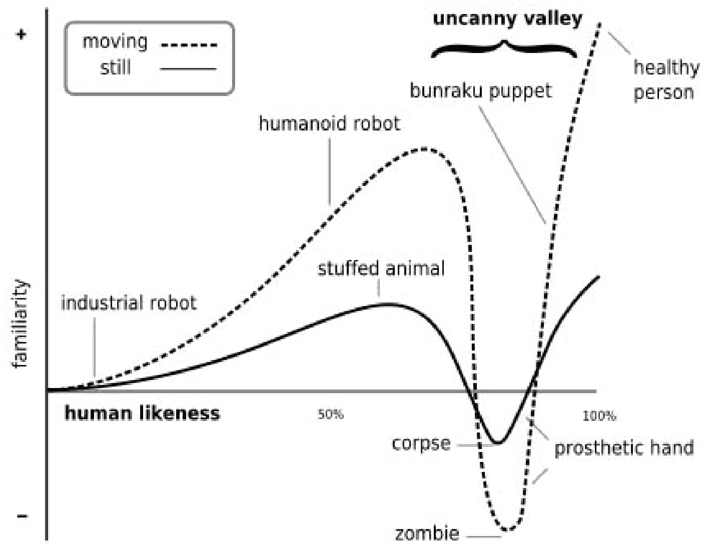

Figure 6.1: Hypothesized emotional response of human in function of the similarity of humanoid robots with human following Mori theory. This diagram has been presented in the translation of "Bukimi no tani" (the uncanny valley) by Karl F. MacDorman and Takashi Minato.

externally differentiable from a human being.

In 1970 Masahiro Mori already cautioned against building humanoid robots that look too human. He developed the theory known as the theory of the "uncanny valley" (Mori, 1970). This theory describes the emotional response of a human confronted to a human-like robot. In fact, following this theory, the more human-like a robot is, the more positive is the emotional response of the human subject. The positive aspect of the response increases with the human likeliness of the robot until a certain point were the response turns into a strong repulsion. However, if the similarity of the external appearance and motion of the robot with a human being continues to increase, the response becomes positive again and the interaction tends to be the same interaction than with another human being. The area between a negative response - engendered by robots that are very similar to humans but not enough to engender the same response as a human being - and a positive response - engendered by a very close similarity between robots and humans - is known as the "uncanny valley" (See Figure 6.1).

This theory is widely discussed among roboticists. It is quite difficult to prove this theory and the first reason is that we are currently far from building an android similar enough to human beings to verify the last part of the curve. Some people as Hiroshi Ishiguro and Karl F. MacDorman are specialized in androids science and try to build such androids (Minato, Shimada, Ishiguro, & Itakura, 2004). They argue that using the uncanny valley theory and building robots more and more similar to humans will help to better understand the

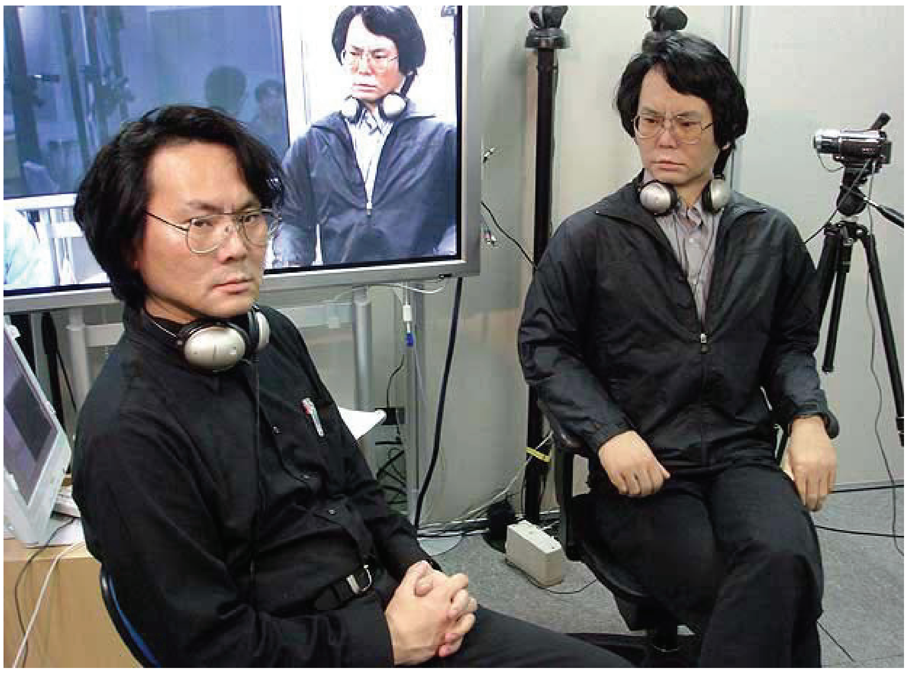
Figure 6.2: Hiroshi Ishiguro and his Geminoid, a teleoperate android created in his image.

communication and social relationships between human beings (MacDorman & Ishiguro, 2006). In fact, by building an Android similar enough to human beings to allow a natural interaction, they should be able to verify which parameters influence the social interaction between humans by controlling and testing these parameters. But at the moment, even if the external aspect of his robot is impressive (see Figure 6.2), the problem is rather situated in the movement control.

The same problem can be encountered with Robota, where one of the constraints of the project is the aesthetic aspect of the robot. The question is: "Where should we put the limits?" For the first robot with five DOFs, the aspect of the robot was the aspect of a doll, a toy like those we can find everywhere. The robot was quite well accepted because it was assimilated to an object (to a toy). More problems are to be expect with the second version of Robota, Robota II. This version is much more evolved and seems to be more than a simple doll. For the time being, only the head is sufficiently advanced to have an idea of the final robot. For the face, the aspect is not anymore the aspect of a doll. Because of the mechanism used to move the eyes, a new face had to be designed. This new face has not the right proportion of a human face, but everything is human-like, see Figure 6.3. The problem becomes more evident when the head begins to move and the eyes of the robot follow your movement. As Robota is designed to be used with autistic children, the first prototype was presented to parents of autistic children during an "open doors"

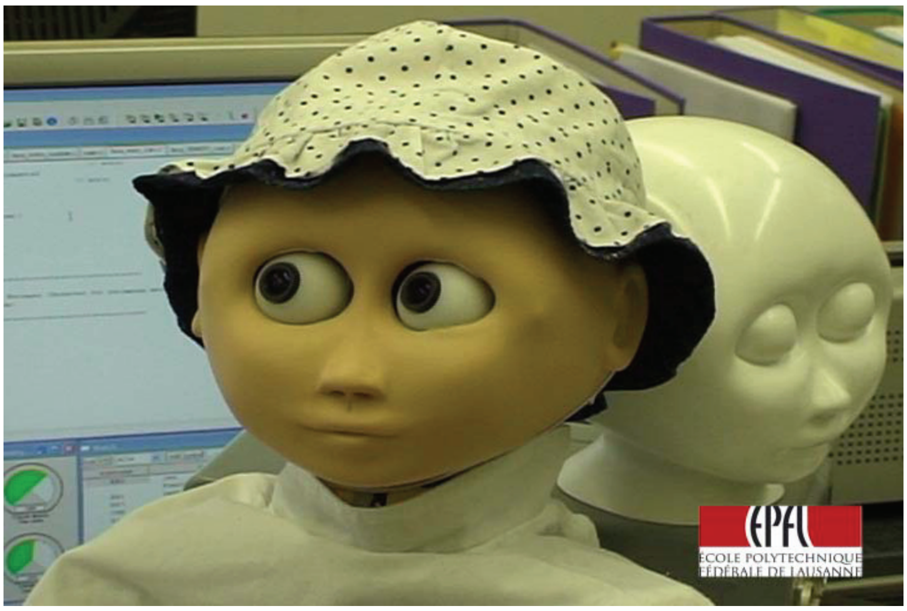

Figure 6.3: Picture of the head of RobotaII.

event organized at EPFL to bring researchers, therapists and parents together. During the presentation of Robota II, a majority of people were worried about the acceptance of Robota II by autistic children due to the “strange” impression that it induces when it moves.

Masahiro Mori argued that the best way to design humanoid robots was to try to reach the first maximum of the curve presented in Figure 6.1, because the second maximum was totally out of reach for the time. The question is now to test where Robota is situated on this curve in order to be able to improve the design and to have the best chances to see the day in which this robot will work with autistic children.

6.2 Solving the correspondence problem, balancing data

As first stressed by Nehaniv and colleagues (Nehaniv, 2003), there are a multitude of correspondence problems when attempting to transfer skills across various agents and situations. One may consider:

1. **Different embodiments:** the demonstrator does not share the same morphology and characteristics as the imitator.
2. **Different situations:** the environment and constraints during the demonstration and the reproduction are different.

In this section, we will first discuss the problem 1, different embodiments. As stressed in Chapter 2 and 3, an important part to reduce the difference of embodiment between the demonstrator and the imitator is to have a robot that is morphologically as similar as possible to the demonstrator[1]. As explained in Chapter 3, we use different sets of data laying in different reference frames. Typically, for a simple example, we use both joint angles and Cartesian coordinates of the robot end-effector. The similarity of morphology intervenes in the way of measuring the joint angles. During the demonstrations, we are able to measure the joint angle that we need for the robot directly on the human demonstrator. For the Hoap 2 and 3 for example, we need 3 angles for the shoulder and 1 for the elbow. By using motion sensors fixed on the demonstrator arm and the adequate model, we directly extract these 4 angles from the demonstration.

Concerning the problem 2, in the general case, for a manipulation task, we also need to express explicitly the position of the end-effector relative to the objects involved in the task. So, the metric defined in Section 3.4.4 is used to find a solution that minimizes the error in three referential frames: the object-hands referential, the end-effector referential and the joint angles referential. In a goal-directed framework, the three referential have different importance. If an object is manipulated, the first referential shows the highest importance. If there is no object in the scene and the hands paths follow an invariant pattern, the second referential is of highest importance. It is the case when we want to write a letter for example. Reproducing the exact gesture is often of lower importance for manipulation tasks but can become highly relevant for motion such as waving the hand or dancing. Bekkering and colleagues have set up experiments to show that imitation is goal-directed, using several gestures to reproduce during an imitation game (Bekkering et al., 2000). They showed that the hand paths and hand-object relationships have different levels of importance, forming a hierarchy of relevance. They also showed that the use of the different levels highly depends on the working memory capacities of the infants/adults playing the game. As robots have an advantage over humans in terms of working memory capacities, it conducted the choice of keeping the different levels of imitation to find a solution, i.e. to use decreasing weight instead of hierarchy that keeps only some levels of variables and discards the information coming from the preceding levels.

During the demonstration, the importance of these different levels are extracted (see Section 3.4). During the reproduction, the more important level is reproduced in priority and the other levels are used to optimize the movement. For example, if more importance is given to the Cartesian trajectory of the end-effector, during the reproduction , the algorithm will try in priority to follow the desired trajectory as closely as possible in the Cartesian space while staying as near as possible to the desired joint angle trajectory. In other words, because of the difference of embodiment, the desired solution is not consistent with a different kinematics. We will thus not be able to perfectly reproduce the desired trajectory in the whole reference frame. So, the more the data of

[1]The problem of humanoid morphology for robot is discussed in the Section 6.1.

a reference frame are important for accomplishing a task, the more the desired trajectory will be respected.

A drawback of this system (as presented in Section 3.4.4) is that the different reference frames influence the trajectory even if their respective ponderation are low. In other words, because the ponderation of each component of the trajectory is proportional to the inverse of each signal variance, this ponderation is never zero. It means that for a reference frame that is not important for the reproduction of a task, even if the ponderation is very low, a big error in this reference frame will influence the final trajectory and can for example prevent the trajectory to reach the target. Moreover, the proposed system is open-loop and aimes at providing a solution to the reproduction of a task, which is a generalization of the demonstrations produced. The trajectory is then generated before being executed by the robot. Thus, once the initial position of the object in the scene is recorded, any change in the setup will cause a failure of the system. The system is not able to tackle dynamical changes in the setup. That can be a problem when working in real environment, not only for the reproduction of the task, but also for the security. The robot must be able to react in real time in order to avoid hitting the user or any people around for example. This is one of the main reasons that lead us to further develop the algorithm and try to use DS techniques and potential fields method to be able to react online to unexpected events.

6.3 Restrictions of reaching movement for reproducing general tasks

In order to be more robust for the reproduction of the movement - in terms of generalizing the movement into the whole workspace even if the demonstrations are done in a small subspace (see Chapter 4 and 5) - we added a DS component to the system. The DS used, inspired by the VITE modelization of human movement, is very interesting in order to produce an online trajectory specifically generated to reach a target.

To integrate this DS component into the algorithm, we had to put the hypothesis that we are able to decompose every manipulation movement into simple reaching movements. The idea is that every manipulation task is based on simple pick and place movements. An example widely used in the laboratory for experiments with the robot is teaching it how to move chess pieces. In this kind of game, every movement can be decomposed in a simple reaching movement. For example, from a resting position, the robot has to reach a chess piece, when the location of the chess piece is reached it can grasp it and move it to another position. Once the new position is reached, it releases it and reaches a resting position. Thus, the complete movement can be decomposed into 3 reaching movements. As we used a trajectory learned by demonstration to modulate the

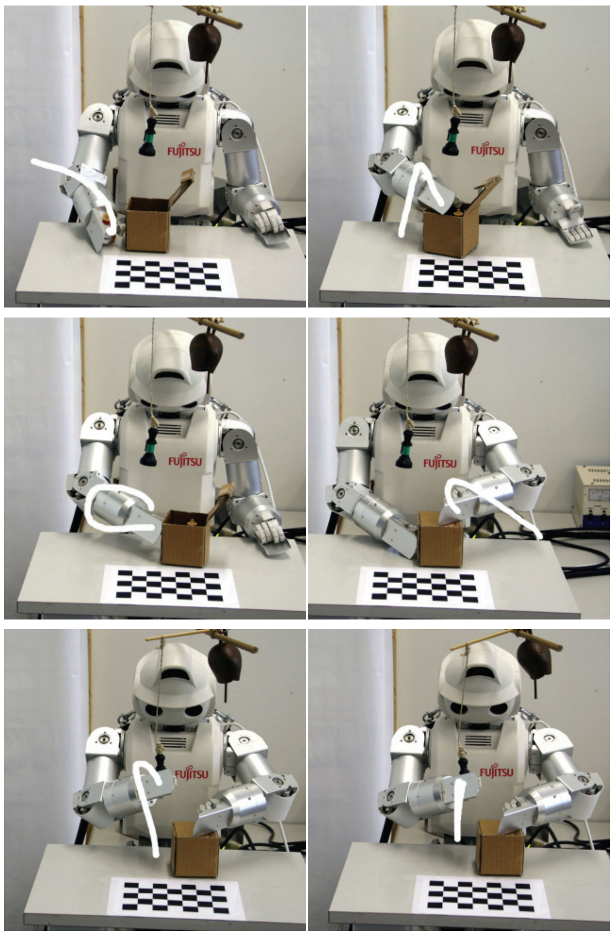
Figure 6.4: The trajectories generated for the execution of the packaging task presented at HUMANOIDS 2006. In the first row, the robot reaches for an object and puts it into the box. In the second row, it holds the box with the right hand and closes it with the left hand. In the third row, it reaches for the bell and rings it.

DS, it is not a problem to learn more complicated movements as the movement of the knight in the chess game.

This hypothesis can be very restrictive if we want to teach the robot other movements or tasks than manipulation movements. In (Hersch et al., 2006), we wanted to teach the robot different movements in order to pack wood pieces in a box (See Figure 6.4). The last movement performed by the robot was to ring a little bell through a cyclic movement with the hand. By using the system presented in Chapter 4 without the RL module, the reproduction was good for the first cycle, but due to the constant attraction of the target that was placed at the initial position, every following cycle lost more and more amplitude.

With this type of algorithm, it is not possible to teach the robot tasks like writing letters as it was done by Sylvain Calinon (Calinon & Billard, 2005) or more abstract movements like dancing or communicating by using sign language.

For communication, there are a lot of movements that cannot be modelized with a Dynamical system. So, even if the robustness to displacement of objects in the setup is greatly increased, there are still important restrictions in terms of movement that can be learned by using a DS for the reproduction of a task. However, by changing slightly the type of target and by dividing the movement in an appropriate sequence, it would probably be possible to adapt the type of algorithms presented in this work for different types of movement. Then the most difficult part would be to design a higher level controller that is able to divide the movement to learn in a sequence of reaching movements and to identify a target for each sub-task. By identifying a target, I mean also to determine in which frame of reference we have a target. For the time being, the target was part of an apriori knowledge.

6.4 Working in position, speed or acceleration?

We tried different ways of encoding the data during this work. In the first version, the position data were used in order to train the model. From the hardware point of view, it was an advantage. Even if, basically, we can control electrical DC motors by varying the tension on the motor - in other words, by controlling the torque of the motor - in most of the robots we use, the low level control is designed to control the robot in position. It is the case for the HOAP 3 robot, the Katana robots and even the control module used in Robota 2.

However, depending on the algorithm, it is sometimes easier to work at different levels. In robot control, for example, the inverse kinematic problem is usually solved by using the Jacobian matrix that gives a relation between the speed in the Cartesian coordinate and the speed in the joint angles of the robot (see Chapter 3 and 4). It is then more consistent to work at the speed level if we use these techniques. Thus, we do not need to transform the data of the desired trajectory given by the GMM. Another advantage is that we are more independent for the location of the starting position. When the desired trajectory is given by a speed profile, we can start from a different position. The only problem is to adapt the speed profile to the robot's capacity to be sure to avoid reaching the joint limits.

If we use a DS to generate trajectory robust to perturbation, we work at the acceleration level. The DS is a system that computes an acceleration knowing the current speed and position of the end-effector for example. Using GMM in order to encode the acceleration profile observed during the demonstration seems then the right solution in order to combine the two systems. In Chapter 5 different experiments have been conducted with such a system. These tests include a first trial to learn directly the dynamics of the demonstration data to avoid an obstacle in simple configuration. An extension of this work is presented in (Khansari Zadeh & Billard, 2009) where the dynamics is learned for both discrete and cyclic movements. The advantage of working at the acceleration level is that acceleration is directly proportional to the torque. That allows us to better integrate external forces that influence the trajectory as for example repulsive forces centered in an eventual obstacle. This provides us a method to avoid the obstacles if necessary. The problem to be able to fully exploit this control system is still the hardware problem. All the robots used in the LASA lab are robots that are controllable only in position. It means that we still have to work in position at the low level and thus pass by an integration of the signal in order to be able to work with the robot.

6.5 Learning or optimization?

In Chapter 4, we have compared two different methods to allow the robot to update its own model of the task by using trial-and-error learning. The first method is a Reinforcement Learning technique while the second is an operations research technique.

The principle of the extended Taguchi method is to set a plan of experiments and to use it to tune some parameters that are believed to influence the output of the system to be optimized. It was originally widely used for example in quality management for mass production, but this algorithm is now also used to optimize the efficiency of other complex algorithms. Following this idea, I have try to adapt the extended Taguchi method for the trial-and-error component of the system presented in Chapter 4. The ability of the extended Taguchi method to deal with several parameters and the simplicity of the system were the two main reasons to choose this algorithm.

The main difference between the two methods comes from the fact that we have a learning algorithm and an optimization algorithm. In fact, the two methods have the same goal, to optimize the reward function (respectively the fitness function) of the trajectory generated by the robot, but the way to reach this goal is different. For the RL algorithm, we have a stochastic exploration of the solution space around the current trajectory. The solution space is not defined and does not have any boundary. During the learning, the trajectory and the search space attached evolve step by step and can continue to evolve indefinitely. In the extended Taguchi method, the search space is defined at the beginning and is explored systematically by the system. At each step of the algorithm, the search space is reduced until a complete convergence of the system. In this sense, there is no learning, because there is no evolution of the solution. The system is solely looking for the optimal solution in the bounded search space.

The two techniques have their advantages and disadvantages. The main advantage of the RL is that, due to the evolution of the solution during the learning process, we are able to interrupt the learning at any time if the current solution seems to be satisfactory even if it is not optimal. However, it is very difficult to tune the parameters of the NAC in order to have a correct convergence of the system.

Concerning the extended Taguchi method, the boundaries has to be fixed at the beginning, there is thus no possibility to go out of the boundaries. This means that the boundaries has to be set carefully depending of the problem to solve. In the examples presented in Chapter 4, the boundaries were set to the mechanical limit of the robot (in terms of speed), thus we ensure the exploration of the whole space of possible solutions. The other problem of this method is that as the information of the GMM covariance matrices are not used, we loose the idea to keep the constraint learned during the demonstration for the exploration of the solution space. Only the fitness function guaranties that we have a solution with a similar shape than the original trajectory. The big advantages of the extended Taguchi method is its simplicity and efficiency to find a very good solution.

For the experiment of learning the shape of the object in chapter 5, the extended Taguchi method has been chosen for its simplicity of use and for its searching method. In principle, the NAC follows better the idea of a robot that is able to learn by itself, but in practice, it is very difficult to have an efficient tuning of the parameter to obtain a correct convergence of the system.

7 Conclusion

In this thesis, we were interested in developing different algorithms for programming robot by demonstration. We were especially interested to solve the specific problem known as the “how to imitate” problem. To solve this problem, we assume that the problem known as the “what to imitate” problem which consists in finding a solution to extract the constraints necessary to accomplish a given task from a set of demonstrations provided by the user, is already solved.

Within the “how to imitate” problem, two main problems have been identified. The first problem is related to the transfer of skills from a human to a robot and concerns the problem of the difference of embodiment between the two actors. This problem is also known as the “correspondence problem”. The second problem is related to the difference of situation that can be encountered when the robot has to reproduce a learned task. We need to have a algorithm that can handle situations that have not been encountered during the demonstrations.

In Chapter 2, we present the hardware platform developed or used throughout this thesis. The project Robota is introduced and the motivations to design a new version of Robota is presented. The design of a humanoid robot with great similarities in terms of morphology is already the first part of the solution to the “correspondence problem”. The transfer of skills in much easier with a humanoid robot than with all other types of robot. The humanoid robot HOAP 3 has also been presented.

In Chapter 3, we begin with the main part of the thesis, the development of a control algorithm to reproduce some tasks learned by demonstration. This Chapter described the principles used in this thesis to solve the “correspondence problem” with the hypothesis to use a humanoid robot as imitator. The main idea is to use data in different referential frame like, for example, the referential frame of the end-effector in terms of Cartesian coordinates and the referential frame of the robot’s arm in terms of joint angles. The redundancies of the data collected in these different referential frames are used to create a movement as similar as possible to the learned movement despite of the difference of embodiment. Inverse kinematics techniques are used in order to combine the different referential frame data.

In Chapter 4, we present the first algorithm developed in order to solve the second problem identified within the “how to imitate” problem, the problem of the difference of situation. This algorithm has been designed in collaboration with two other PhD students and works under the hypothesis that every manipulation movements can be decomposed in a set of reaching movements. The proposed algorithm consists mainly in a DS module inspired by a VITE model of human movement. This DS brings the end-effector of the robot smoothly to a desired target. A modulation is then introduced to add the component of the movement taught by the user. This modulation has the form of a speed profile that perturbs the trajectory of the DS. A last module is used to allow the robot to relearn its own model of the task (its own modulation trajectory) in order to optimize the reproduction or to handle unexpected events. Two methods are then compared in order to create this module, the Natural Actor Critic Algorithm (NAC), a RL algorithm and the extended Taguchi method, a operations research algorithm.

In Chapter 5, we present the last evolution of the algorithm. Here, we use the same kind of algorithm than in Chapter 4 with a modulation at the acceleration level. The idea to work in acceleration allow us to introduce a repulsive acceleration that bring the arm of the robot away from eventual obstacles. We are now able to use the self learning module to learn the influence of eventual obstacles on the reproduction trajectory.

Finally, a couple of key points are presented in Chapter 6 to better discuss the choices made in this work.

This thesis contributes to the research on RbD by proposing a solution to the "correspondence problem" through the use of humanoid robots and the combination of redundant frames of reference for generating a reproduction trajectory. In addition to that, we have introduced the idea that imitation is not a passive repetition of a task, but rather an explorative process to optimize the reproduction of this task. By combining trial-and-error learning and imitation learning, we showed that we can create more effective learning algorithms.

A Appendix

A.1 Additional results for the learning of the DS modulation

This experiment have been conducted using the system described in Chapter 4. Here, the goal is not to avoid a obstacle, but to optimize the grasping of a chess piece. In order to force the robot to reach for the chess piece from a desired direction, a forbidden volume V has been defined and the reward function has been slightly modified to penalize the penetration of the the volume V by the end-effector of the robot.

$$r_q(\dot{\xi}^s, \xi^s) = \sum_{t=0}^{T} -c_1|\dot{\xi}^s_t - \dot{\xi}^m_{t,q=1}| + c_2 P(\dot{\xi}^m_{t,q}) - c_3|\xi^s_T - \xi^g| \tag{A.1}$$

where

$$P(\dot{\xi}^m_{t,q}) = \begin{cases} -1 & \text{if } \dot{\xi}^m_{t,q} \in V \\ 0 & \text{if } \dot{\xi}^m_{t,q} \notin V \end{cases} \tag{A.2}$$

For this experiment, the position of the chess piece is fixed and we have chosen 23 starting positions distributed along a horizontal line in front of the robot. The first position is situated in front of the robot and the last position in the right. The task is reproduced using the right arm. For this experiment, the stopping condition is that ξ^s_q has avoided V and that $\xi^s_{T,q}$ is on the target. Fig. A.1 represents the trajectories for 6 selected starting positions. We observe that when the starting position is not in front of the queen, the preferred solution is to grasp the queen from above, but we observe some exceptions for starting positions 1, 9, 17, and 21.

Fig. A.2 represents the statistics of 11 runs of the system for each starting point. As in the box task, we observe cases of failure of the algorithm. This phenomenon is more accentuated for the leftmost positions (from 1 to 9). This is due to the fact that we are near the joint angle limit for these starting positions. For starting positions 10 to 14, the task is reproduced correctly at the first trial. These starting positions are in front of the chess queen and then the dynamical system is sufficient for grasping the queen. For starting position on the right,

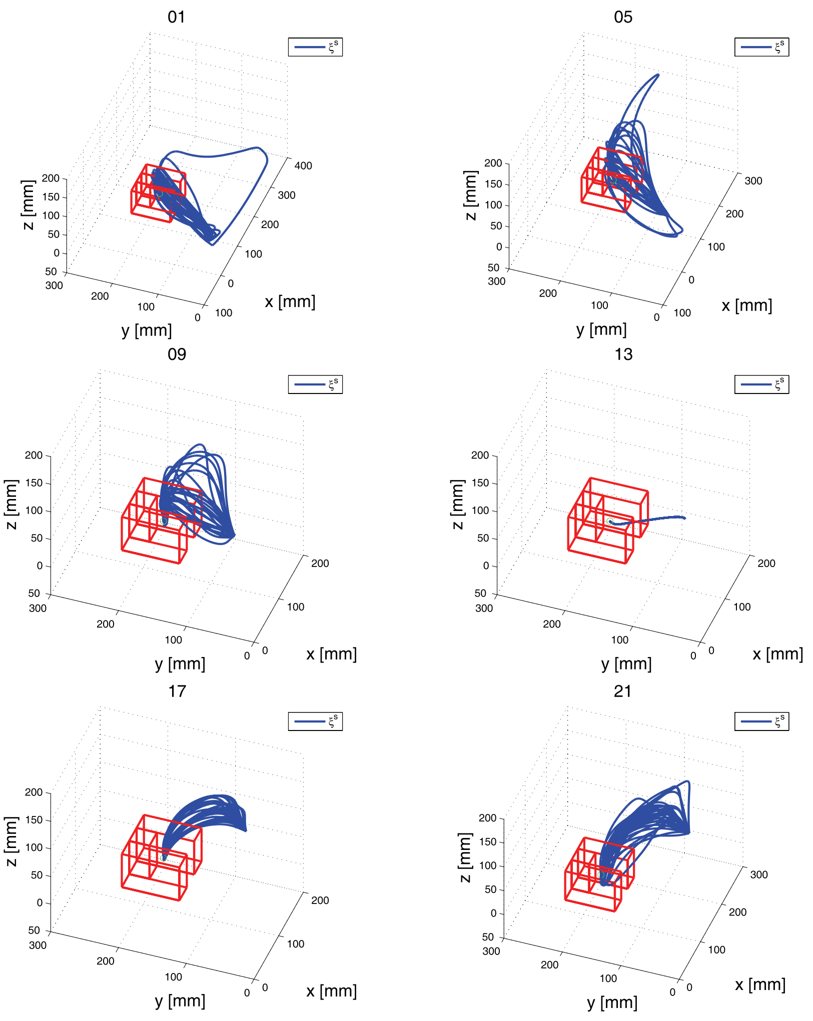

Figure A.1: This figure represents the trajectories for 6 starting point of the chess task.

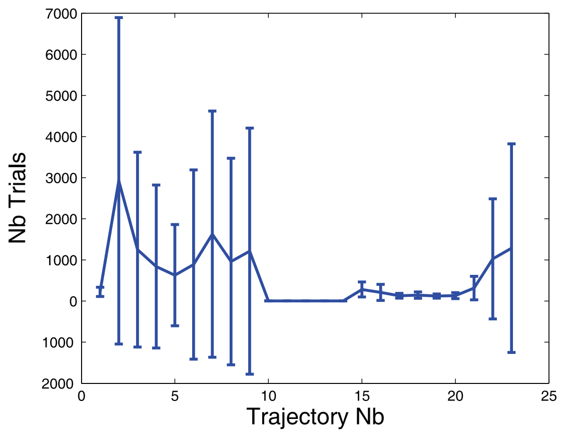

Figure A.2: This graphic represents the statistics of the number of trials needed to converge to a solution for the chess task. As abscissa, we have the starting position (23 starting positions equally distributed along a horizontal line from the center to the right of the robot). As ordinate, we have the statistics on the number of trials needed to find a solution depending on the starting position.

reinforcement learning is necessary but a solution is found in a small number of trials, except for the two last positions that are near the joint limits.

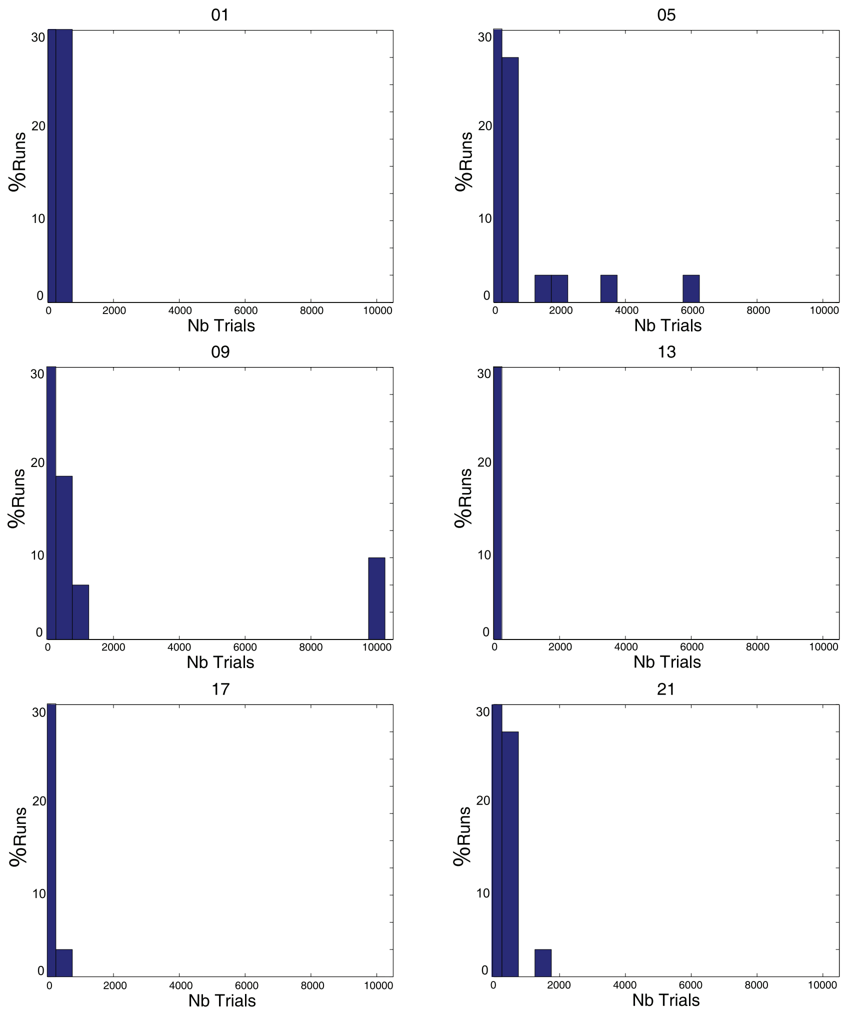

Figure A.3: This figure represents the histogram of the number of trials for each run for 6 selected starting positions of the chess task.

A.2 Example of an OA Table

1	1	1	1	1	1	1
2	2	2	2	2	2	1
3	3	3	3	3	3	1
1	1	2	3	2	3	1
2	2	3	1	3	1	1
3	3	1	2	1	2	1
1	2	1	3	3	2	2
2	3	2	1	1	3	2
3	1	3	2	2	1	2
1	3	3	1	2	2	2
2	1	1	2	3	3	2
3	2	2	3	1	1	2
1	2	3	2	1	3	3
2	3	1	3	2	1	3
3	1	2	1	3	2	3
1	3	2	2	3	1	3
2	1	3	3	1	2	3
3	2	1	1	2	3	3

Table A.1: Orthogonal Arrays of 18 runs for 7 parameters, 3 levels and a strength of 2 (OA 18.7.3.2). A strength of 2 means that for every submatrix of 18 by 2, all the distinctive possible raw (here, $2^2 = 4$ distinctive possible raw) appear the same number of times. In most of experiments conducted in this thesis, we use a OA 54.20.3.2

References

Abbeel, P., & Quigley, M. (2006). Using inaccurate models in reinforcement learning. In *International conference on machine learning.*

Alissandrakis, A., Nehaniv, C., & Dautenhahn, K. (2002). Imitating with alice: Learning to imitate corresponding actions across dissimilar embodiments. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*.

Amari, S. (2000). Natural gradient works efficiently in learning. *Neural Computation, vol. 10, pp. 251-276*.

Ambrose, R., Aldridge, H., Askew, R., Burridge, R., Bluethmann, W., Diftler, M., et al. (2000). Robonaut: Nasa's space humanoid. *IEEE Intelligent Systems and Their Applications, Vol 15, Issue 4, pp. 57-63*.

Andry, P., Gaussier, P., Moga, S., Banquet, J., & Nadel, J. (2001). Learning and communication via imitation: An autonomous robot perspective. *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans, Vol. 31, No. 5, pp. 431-442*.

Arkin, R. (1989). Motor schema-based mobile robot navigation. *The International Journal of Robotics Research, Vol. 8, No. 4, pp. 92-112*.

Atkeson, C., & Schaal, S. (1997). Learning tasks from a single demonstration. In *Proceedings of the ieee/rsj int. conference on robotics and automation (icra'97)* (Vol. 2).

Atkeson, C. G., & Schaal, S. (1995). Memory-based neural networks for robot learning. *Neurocomputing*, *9:3*, 243-269.

Barto, A., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications, 13, pp. 343-379*.

Baumann, R. (1997). *Haptic interface for virtual reality based laparoscopic surgery training environment.* Unpublished doctoral dissertation, Ecole Polytechnique Federale de Lausanne (EPFL).

Bekkering, H., Wohlschlger, A., & Gattis, M. (2000). Imitation of gestures in children is goal-directed. *Quarterly Journal of Experimental Psychology*, *53A (1)*, 153-164.

Bentivegna, D., Ude, A., C.G., A., & Cheng, G. (2002). Humanoid robot learning and game playing using pc-based vision. In *Ieee/rsj international conference on intelligent robots and systems (iros).*

Billard, A. (1999). Drama, a connectionist architecture for on-line learning and control of autonomous robots: Experiments on learning of a synthetic proto-language with a doll robot. *Industrial Robot*, *26:1*, 59-66.

Billard, A. (2003). Robota: Clever toy and educational tool. *Robotics & Autonomous Systems*, *42*, 259-269.

Billard, A., Epars, Y., Calinon, S., Cheng, G., & Schaal, S. (2004). Discovering optimal imitation strategies. *Robotics & Autonomous Systems, vol. 43:2-3, pp. 69-77*.

Billard, A., Epars, Y., Cheng, G., & Schaal, S. (2003). Discovering imitation strategies through categorization of multi-dimensional data. In *Proceedings of the 2003 ieee/rsj intl. conference on intelligent robots and systems.*

Billard, A., & Schaal, S. (2001). A connectionist model for on-line learning by imitation. In *Proceedings of the ieee/rsj int. conference on intelligent robots and systems.* hawaii.

Billard, A., & Siegwart, R. (Eds.). (2004). *Robotics and autonomous systems, special issue: Robot learning from demonstration* (Vol. 47) (Nos. 2,3). Elsevier.

Borenstein, J., & Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man and Cybernetics , vol. 19, No. 5, pp. 1175-1187.*

Boyan, J. (2002). Technical update: Least-squares temporal difference learning. *Machine Learning, 49, pp. 233-246.*

Bratke, S., & Duff, M. (1994). Reinforcement learning methods for continuous-time markov decision problems. In *Neural information processing systems conference.*

Brooks, R., Breazeal, C., Marjanovic, M., & Scassellati, M., B. Williamson. (1999). The cog project: Building a humanoid robot. *Computation for Metaphors, Analogy, and Agents, LNCS 1562, pp. 52-87.*

Bullock, D., & Grossberg, S. (1988). Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation. *Psychological Review, vol. 95(1), pp. 49-90.*

Calinon, S. (2007). *Continuous extraction of task constraints in a robot programming by demonstration framework.* Unpublished doctoral dissertation, Ecole Polytechnique Fédérale de Lausanne (EPFL).

Calinon, S., & Billard, A. (2005). Recognition and reproduction of gestures using a probabilistic frameworkcombining pca, ica and hmm. In *International conference on machine learning (icml).*

Calinon, S., Guenter, F., & Billard, A. (2005). Goal-directed imitation in a humanoid robot. In *Ieee intl conference on robotics and automation.*

Calinon, S., Guenter, F., & Billard, A. (2007). On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, 37:2. Part B. Special issue on robot learning by observation, demonstration and imitation.*

Dafflon, M. (2008). *Préhenseurs, conditions et stratégies pour une micromanipulation de précision.* Unpublished doctoral dissertation, Ecole Polytechnique Fédérale de Lausanne (EPFL).

Dautenhahn, K., & Christaller, T. (1996). Remembering, rehearsal and empathy - towards a social and embodied cognitive psychology for artifacts. In S. O'Nuallain & P. M. Kevitt (Eds.), *Two sciences of the mind. readings in cognitive science and consciousness* (p. 257-282). John Benjamins North America Inc.

Dautenhahn, K., & Nehaniv, C. (Eds.). (2001). *Imitation in animals and artifacts.* The MIT Press.

Demiris, J., & Hayes, G. (2001). Imitation as a dual-route process featuring predictive and learning components: A biologically-plausible computational model. In C. Nehaniv & K. Dautenhahn (Eds.), *Imitation in animals and artifacs.* MIT Press.

Dillmann, R. (2004). Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems.*

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation, vol. 12, pp. 219-245.*

Driscoll, J. (2000). Comparison of neural network architectures for the modeling of robot inverse kinematics. *IEEE. Southeastcon 2000*.

D'Souza, A., Vijayakumar, S., & Schaal, S. (2001). Learning inverse kinematics. In *Ieee/rsj intl conference on intelligent robots and systems.*

Fujita, M., Kuroki, Y., Ishida, T., & Doi, T. (2003). A small humanoid robot sdr-4x for entertainment applications. In *Ieee/asme intl conference on advanced intelligent mechatronics.*

Ghahramani, Z., & Jordan, M. (1994). Supervised learning from incomplete data via an em approach. *Advances in Neural Information Processing Systems 6*.

Gouaillier, D., & Blazevic, P. (2006). A mechatronic platform, the aldebaran robotics humanoid robot. In *Ieee conference on industrial electronics (iecon).*

Guenter, F., Guignard, A., Piccardi, L., Calzascia, M., & Billard, A. (2004). Development of a miniature aerticulated arm and pair of eyes for the humanoid robot robota. In *Ieee/aps intl conference on mechatronics & robotics.*

Haken, H., Kelso, J., Fuchs, A., & Pandya, A. (1990). Dynamic pattern recognition of coordinated biological motion. *Neural Networks, vol. 3, pp. 395-401*.

Handbook of industrial robotics. (1999). John Wiley and Sons.

Hashimoto, s. e. a. (2002). Humanoids robots in waseda university - hadaly-2 and wabian. *Autonomous Robots 12, pp. 25-38*.

Hedayat, A. (Ed.). (1999). *Orthogonal arrays: Theory and applications.* Springer-Verlag.

Hersch, M., Guenter, F., Calinon, S., & Billard, A. (2006). Learning dynamical system modulation for constraint reaching tasks. In *Ieee-ras international conference on humanoid robots-humanoids'06.*

Hersch, M., Guenter, F., Calinon, S., & Billard, A. (2008). Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics, in press*.

Iida, S., Kanoh, M., Kato, S., & Itoh, H. (2004). Reinforcement learning for motion control of humanoid robots. In *Ieee/rsj international conference on intelligent robots and systems.*

Ijspeert, A., Nakanishi, J., & Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the ieee international conference on robotics and automation.*

Iossifidis, I., & Schoner, G. (2004). Autonomous reaching and obstacle avoidance with anthropomorphic arm of a robotics assistant using the attractor dynamics approach. In *Ieee international conference on robotics and automation.*

Kaiser, M., & Dillmann, R. (1996). Building elementary robot skills from human demonstration. In *Ieee intl conference on robotics and automation.*

Karlik, B., & S.Aydin, S. (2000). An improved approach to the solution of inverse kinematics problems for robot manipulators. *Engineering Applications of Artificial Intelligence*.

Kato, I. (1973). Development of wabot 1. *Biomechanism, 2, pp. 173-214*.

Kazuko, I. e. a. (2006). Development of a bioinstrumentation system in the interaction between a human and a robot. In *Ieee/rsj international conference on intelligent robots and systems.*

Khansari Zadeh, S. M., & Billard, A. (2009). Learning an adaptative model of the dynamics of arm motions by demonstration. In *Ieee international conference on robotics and automation, submited.*

Khatib, O. (1986). Real-time obstacle avoidance for manipulation and mobile robots. *International Journal for Robotics Research, vol. 5, No. 10, pp. 90-99*.

Kieffer, S., Morellas, V., & Donath, M. (1991). Neural network learning of the inverse kinematic relationships for a robot arm. In *Ieee intl conference on robotics and automation.*

Ko, N., & Lee, B. (1996). Avoidability measure in moving obstacle avoidance problem and its use for robot motion planning. In *Ieee/rsj international conference on intelligent robots and systems (iros).*

Konda, V., & Tsitsiklis, J. (2000). Actor-critic algorithms. *Advances in Neural Information Processing Systems 12. MIT Press*.

Konidaris, G., & Barto, A. (2006). Autonomous shaping: Knowledge transfer in reinforcement learning. In *International conference on machine learning.*

Kuniyoshi, M. (1994). The science of imitation -towards physically and socially grounded intelligence. In *Rwc joint symposium.*

Liegeois, A. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics, vol. 7 (12), pp. 868-871*.

MacDorman, K. F., & Ishiguro, H. (2006). The uncanny advantage of using androids in social and cognitive science research. *Interaction Studies, vol. 7(3), pp 297-337*.

Minato, T., Shimada, M., Ishiguro, H., & Itakura, S. (2004). Development of an android robot for studying human-robot interaction. In *Proc. of the seventeenth international conference on industrial and engineering applications of artificial intelligence and expert systems (iea/aie).*

Mizuuchi, I., Inaba, M., & Inoue, H. (2001). A flexible spine human-form robot - development and control of the posture of the spine. In *Ieee/rsj intl. conference on intelligent robots and systems.*

Mizuuchi, I., Tajima, R., Yoshikai, T., Sato, D., Nagashima, K., Inaba, M., et al. (2002). The design and control of the flexible spine of a fully tendon-driven humanoid kenta. In *Ieee/rsj intl. conference on intelligent robots and systems.*

Mizuuchi, I., Yoshikai, T., Sodeyama, Y., Nakanishi, Y., Miyadera, A., Yamamoto, T., et al. (2006). Development of musculoskeletal humanoid kotaro. In *Ieee international conference on robotics and automation.*

Mori, M. (1970). Bukimi no tani [the uncanny valley]. *Energy, vol. 7, pp 33-35*.

Morimoto, J., & Doya, K. (2001). Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics and Autonomous Systems, vol. 36, pp. 37-51*.

Nakamura, Y., & Hanafusa, H. (1986). Inverse kinematics solutions with singularity robustness for robot manipulator control. *ASME Journal of Dynamic Systems, Measurment and Control,vol. 108,pp. 163-171*.

Nedic, A., & Bertsekas, D. (2002). Least-squares policy evaluation algorithms with linear function approximation. *LIDS Report LIDS-P-2537, Dec. 2001; to appear in J. of Discrete Event Systems*.

Nehaniv, C. (2003). Nine billion correspondance problems and some methods for solving them. In *International symposium on imitation in animals & artefacts, pp. 93-95.*

Nehaniv, C., & Dautenhahn, K. (1999). Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In J. Demiris & A. Birk (Eds.), *Learning robots: An interdisciplinary approach.* World Scientific Press.

Ogura, Y. e. a. (2006). Development of a humanoid robot wabian-2. In *Ieee international conference on robotics and automation.*

Oyama, E., Agah, A., MacDorman, K., Maeda, T., & Tachi, S. (2001). A modular neural network architecture for inverse kinematics model learning. *Neurocomputing.*

Park, J., & Khatib, O. (2008). Robot multiple contact control. *Robotica 2008, Cambridge University Press.*

Pedersen, L., Bualat, M., Kunz, C., Lee, S., Sargent, R., Washington, R., et al. (2003). Instrument deployment for mars rovers. In *Ieee international conference on robotics and automation.*

Peters, J., Vijayakumar, S., & Schaal, S. (2003). Reinforcement learning for humanoid robotics. In *Humanoids2003, third ieee-ras international conference on humanoid robots.*

Peters, J., Vijayakumar, S., & Schaal, S. (2005). Natural actor-critic. In *16th european conference on machine learning, pp. 280-291.*

Pfeifer, R. (1998). Embodied system life. In *International symposium on system life.*

Pfeifer, R., & Scheier, C. (1997). Sensory-motor coordination: The metaphor and beyond. *Robotics and Autonomous Systems, vol. 20(2), pp. 157-178.*

Plaisant, C., Druin, A., Lathan, C., Dakhan, K., Edwards, k., Vice, J., et al. (2000). Storytelling robot for pediatric rehabilitation. In *Assets'00.*

Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrons - from backpropagation to adaptative learning algorithms. *Int. Journal of Computer Standards and Interfaces, Vol. 16, pp. 165-278.*

Righetti, L., & Ijspeert, A. (2006). Programmable central pattern generators: a application to biped locomotion control. In *Ieee international conference on robotics and automation.*

Robins, B., Dautenhahn, K., Boekhorst, R. te, & Billard, A. (2004). Effects of repeated exposure of a humanoid robot on children with autism. *S. Keates, J. Clarkson, P. Langdon and P. Robinson (eds), Designing a More Inclusive World. London Springer-Verlag,pp.225-236.*

Rosheim, M. (Ed.). (2006). *Leonardo's lost robots.* Springer.

Roy, R. (Ed.). (1990). *A primer on the tagushi method.* Van Nostrand Reinhold.

Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., & Higaki, K., N. Fujimura. (2002). The intelligent asimo: System overview and integration. In *Ieee/rsj intl conference on intelligent robots and systems.*

Sandini, G., Metta, G., & Vernon, D. (2007). The icub cognitive humanoid robot: An open-system research platform for enactive cognition. *50 Years of AI, Festschrift, LNAI 4850, pp. 359370, Springer-Verlag, Heidelberg.*

Schaal, S., Ijspeert, A., & Billard, A. (2003). Computational approaches to motor learning by imitation. *Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences, 358, 1431, pp.537-547.*

Schaal, S., Peters, J., Nakanishi, J., & Ijspeert, A. (2004). Learning movement primitives. In *International symposium on robotics research (isrr2003), springer tracts in advanced robotics.*

Schoner, G., Dose, M., & Engels, C. (1995). Dynamics of behaviour: Theory and application for autonomous robot architecture. *Robotics and Autonomous Systems, vol. 16, pp. 213-245.*

Sentis, L., & Khatib, O. (2006). A whole-body control framework for humanoids operating in human environments. In *Ieee international conference on robotics and automation.*

Sewan, K. (2004). Autonomous cleaning robot: Roboking system integration and overview. In *Ieee international conference on robotics and automation.*

Simsek, O., & Barto, A. (2006). An intrinsic reward mechanism for efficient exploration. In *International conference on machine learning.*

Skubic, M., & Volz, R. (2000). Acquiring robust, force-based assembly skills from human demonstration. In *Ieee transactions on robotics and automation* (Vol. 16:6, p. 772 -781).

Smith, J., Campbell, S., & Morton, J. (2005). Design and implementation of a control algorithm for an autonomous lawnmower. In *48th midwest symposium on circuits and systems, vol. 1, pp. 456-459.*

Solis, J. e. a. (2007). The waseda flutist robot no.4 refined iv: Enhancing the sound clarity and the articulation between notes by improving the lips and tonguing mechanisms. In *Ieee/rsj international conference on intelligent robots and systems.*

Stauffer, Y., Bouri, M., Schmitt, C., Allemand, Y., Gnemmi, J., S.and Fournier, Clavel, R., et al. (2007). Cyberthèse, mise en oeuvre d'un nouveau concept de rééducation pour paraplégiques. *Journal Européen des systmes automatisés (JESA), 41(2):261-278..*

Steil, J., Roethling, F., Haschke, R., & Ritter, H. (2004). Situated robot learning for multi-modal instruction and imitation of grasping. *Robotics and Autonomous Systems, vol. 47, no. 2-3, pp. 129-141.*

Sugano, S., & Kato, I. (1987). Wabot-2: Autonomous robot with dexterous finger-arm–finger-arm coordination control in keyboard performance. In *Ieee international conference on robotics and automation.*

Sutton, R., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems 12. MIT Press.*

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning : an introduction.* Cambridge, Mass. : MIT Press.

Tsakarakis, N., Metta, G., Sandini, G., Vernon, D., Beira, R., Becchi, L., F. Righetti, et al. (2007). icub - the design and realization of an open humanoid platform for cognitive and neuroscience research. *Journal of Advanced Robotics: Special issue on Robotic platforms for Research in Neuroscience, vol. 21(10), pp. 1151-1175.*

Ude, A., Atkeson, C., & Riley, M. (2004). Programming full-body movements for humanoid robots by observation. *Robotics and Autonomous Systems.*

Wampler, C. (1986). Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *Transactions on Systems, Man and Cybernetics, 16(1), pp. 93-101.*

Wang, Y., & Chirikjian, G. (2000). A new potential field method for robot path planning. In *Ieee international conference on robotics and automation (icra).*

Warren, C. (1989). Global path planning using artificial potential fields. In *Ieee international conference on robotics and automation (icra).*

Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, 8, pp. 229-256.*

Yamaguchi, J., Soga, E., Inoue, S., & Takanishi, A. (1999). Development of a bipedal humanoid robot - control method of whole body cooperative dynamic biped walking. In *Ieee international conference on robotics & automation.*

Yeasin, M., & Chaudhuri, S. (2000). Toward automatic robot programming: learning human skill from visual data. In *Ieee transactions on systems, man and cybernetics, part b* (Vol. 30:1).

Zhang, J., & Rössler, B. (2004). Self-valuing learning and generalization with application in visually guided grasping of complex objects. *Robotics and Autonomous Systems.*

Zoellner, R., Pardowitz, M., Knoop, S., & Dillmann, R. (2005). Towards cognitive robots: Building hierarchical task representations of manipulations from human demonstration. In *Ieee international conference on robotics and automation (icra).*

Florent Günter
PhD Student
EPFL-STI-IMT-LASA
Station 9
1015 Lausanne, Switzerland
Tel.: +41 21 693 29 63
Fax: +41 21 693 78 50
E-Mail: florent.guenter@epfl.ch

Curriculum

2004-2008	PhD in Robot Programming by Demonstration at the LASA. EPFL, Switzerland
2002-2004	MsC in Microengineering, specialization in Robotics. EPFL, Switzerland
1999-2002	BsC in Microengineering. EPFL, Switzerland
1979	Born in Sion, Switzerland.

Research interests

- Humanoid robotics and Mechatronics.
- Programming by demonstration.
- Self-learning.

Publications

Journal papers

Guenter, F., Hersch, M., Calinon, S. and Billard, A. (2007) "*Reinforcement Learning for Imitating Constrained Reaching Movements*". RSJ Advanced Robotics, Special Issue on Imitative Robots, vol. 21, pp. 1521-1544.

Hersch, M., Guenter, F., Calinon, S. and Billard, A. (2008) "*Dynamical System Modulation for Robot Learning via Kinesthetic Demonstrations*". IEEE Transactions on Robotics.

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, Part B. Special issue on robot learning by observation, demonstration and imitation, vol. 37, pp. 286-298.

Billard, A., Calinon, S. and Guenter, F. (2006) "*Discriminative and Adaptive Imitation in Uni-Manual and Bi-Manual Tasks*". Robotics and Autonomous Systems, vol. 54, pp. 370-384.

Conference papers

Guenter, F. and Billard, A. (2007) "*Using Reinforcement Learning to Adapt an Imitation Task*". In proceedings of the IEEE/RSJ International

Conference on Intelligent Robots and Systems (IROS), San Diego, USA.

Guenter, F., Roos, L., Guignard, A. and Billard, A. (2005) "*Design of a Biomimetic Upper Body for the Humanoid Robot Robota*". In proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids), Tsukuba, Japan.

Guenter, F., Guignard, A., Piccardi, L., Calzascia, M. and Billard, A. (2004) "*Development of a miniature articulated arm and pair of eyes for the humanoid robot Robota*". In proceedings of the IEEE/APS Conference on Mechatronics and Robotics (MechRob), Aachen, Germany.

Daidie, D., Barbey, O., Guignard, A., Roussy, D., Guenter, F., Ijspeert, A.J. and Billard, A. (2007) "*The Dof-Box Project: An Educational Kit for Configurable Robots*". In proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Zürich, Switzerland.

Hersch, M., Guenter, F., Calinon, S. and Billard, A. (2006) "*Learning Dynamical System Modulation for Constrained Reaching Tasks*". In proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids), Genova, Italy.

Calinon, S., Guenter, F. and Billard, A. (2006) "*On Learning the Statistical Representation of a Task and Generalizing it to Various Contexts*". In proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Orlando, USA.

Roos, L., Guenter, F. and Billard, A. (2006) "*Design of a Biomimetic Spine for the Humanoid Robot Robota*". In proceedings of the IEEE International Conference on Biomedical Robotics and Biomechatronics (BIOROB), Pisa, Italy.

Calinon, S., Guenter, F. and Billard, A. (2005) "*Goal-Directed Imitation in a Humanoid Robot*". In proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spain.

Pongas, D., Guenter, F., Guignard, A. and Billard, A. (2004) "*Development of a Miniature Pair of Eyes With Camera for the Humanoid Robot Robota*". In proceedings of the IEEE-RAS/RSJ International Conference on Humanoid Robots (Humanoids), Santa Monica, Los Angeles, USA.

Technical reports

Guenter, F. (2004) "*Construction et contrôle d'un bras mécanique pour le robot humanoïde Robota*". MsC thesis, EPFL, Switzerland.

Affiliation to funded research projects

COGNIRON	European Integrated Project COGNIRON (“The Cognitive Companion”) funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

Research supervision

Graduate level	“*Generation of Cartesian and joint feasible trajectories in HOAP-2’s 4 DOF robotic arm*”. MsC thesis, EPFL, 2005.
	“*Construction d’un torse mécanique pour le robot humanoïde Robota*”. MsC thesis, EPFL, 2005.
Undergraduate level	“*Implémentation sur le robot HOAP 3 d’un algorithme d’apprentissage basé sur l’imitation et le reinforcement learning*”. EPFL, 2007.
	”*Implémentation d’un module de locomotion pour le robot humanoid HOAP 3*”. EPFL, 2007.
	“*Contrôle d’un moteur synchrone avec un DsPic*”. EPFL, 2007.
	“*Système de sécurité et localisation pour les jeunes enfants*”. EPFL, 2006.
	”*Kit de robot humanoïde ou bio inspiré*”. EPFL, 2005.
	“*Mobile cameras for the eyes of Robota*”. EPFL, 2004.