Modular Cartesian Controllers for Humanoid Robots: Design and Implementation on the iCub

Ugo Pattacini

DIST

University of Genoa, Italy

Istituto Italiano di Tecnologia, Italy

A thesis submitted for the degree of

Philosophiae Doctor (Ph.D.)

February 2011

Declaration

The work of this thesis was conducted from the January 2008 to February 2011 under the supervision of Francesco Nori and Giorgio Metta at Istituto Italiano di Tecnologia (IIT).

to my Family

"This is the true joy of life, the being used up for a purpose recognized by yourself as a mighty one; the being thoroughly worn out before you are thrown on the scrap heap; the being a force of nature instead of a feverish, selfish little clot of ailments and grievances, complaining that the world will not devote itself to making you happy."

G.B. Shaw

ACKNOWLEDGMENTS

The time when I decided to come back to the academic world after having gained an extensive background in the automotive and aerospace fields has been a crucial turning point for my career. On one hand, the excitement of having experienced the challenge of F1 racing was diminished with the new employment in an attractive space company that instead revealed to be mainly paperwork; on the other hand, I felt the hesitation of a leap in the dark leaving a stable and well-paid job with confident perspectives to reinvent myself as student. Therefore I could have not undertaken such adventure without the moral support and the economic effort of my family, whose love and continuous help I have never missed throughout these years as always during my life.

I would like to express my sincere thanks to the director Giulio Sandini and the leader of the Cognitive Humanoids Lab Giorgio Metta, who both gave me the great opportunity to concretely rediscover the pleasure for research, letting me exercise again the fondness of algorithms design and code writing in a discipline, the robotics, I have always dreamt about since I was a child. Robotics is a frontier science and many aspects remain unknown as it happens for the brain and the deep space, so that most of the times I happily spent nights in the laboratory encouraged to find the key to make iCub behave more and more humanly, somehow like the astronomer who waits for the dusk to admire and inquire the stars. Nonetheless I was not alone: I profitably shared ideas, emotions so as the nightly investigations with researchers inspired by the same passion; colleagues who have become dear friends: Matteo, Carlo, Francesco, Marco, Lorenzo, Vadim, Stephane, Zenon. Without their precious suggestions and motivations I would not have been able to achieve some of the results I am particularly proud of. In this respect I would like to especially mention the guide my tutor Francesco has meant to me, who has shaped, enforced and shielded the view of mitigating the enthusiastic learning with robust modeling, the contribution Matteo brought to my study of contact detection with his

invaluable expertise in debugging mechatronic issues, as well as the collaboration I have been having with Carlo, whose perseverance and imagination were the central components that allowed us developing the independent motion detector and the learning of robot eye-hand coordination.

ABSTRACT

The work presented in this thesis deals with the design and the implementation of Cartesian controllers for humanoid robotic platforms employing a model-based approach. The purpose was to achieve a modular architecture that allows coping robustly with all the issues that arise while controlling a humanoid robot in reaching and gazing tasks, such as the compliance with complex obstacles expressed both in joint and operational space, the quality of human likeness, the reliability and repeatability of the action, having at the same time as main requisite the attainment of significant performance in terms of speed and accuracy of the movements. Therefore it is illustrated how the proposed serial configuration composed of a nonlinear constrained optimizers with a minimum-jerk controller placed in cascade fulfills the given specifications.

Moreover, despite the fact that these tools are fundamental for enabling the successive development of higher level modules exhibiting cognitive behaviors, the current research in robotics has not shown the same attention usually reserved for theoretical frameworks to their physical realizations that would address the problem from an organic standpoint, concentrating rather in solutions that remain specific to the single experiment, even not openly accessible in most of the cases, and finally not owning the desirable property of being scalable and portable to different platforms. The Cartesian controllers for reaching and gazing have been thus devised to respond to these central requirements and then put to tests on the *iCub* humanoid.

Furthermore, the study conducted here have been also motivated by the idea that a solid model-based structure can truly represent a boost to learning which by contrast is dominantly applied in literature already at early stages without the resort to any *a priori* knowledge of the task, eventually producing weak results compared to the achievements of the traditional control policies. Modeling and learning can be profitably exploited in a common hierarchical architecture where the former paradigm serves to guarantee at low

level the robustness and the performance of the design, whereas the latter lets the system to adapt to the unknown offsets and perturbations that may occur at higher level while the robot interacts with the environment, when for instance it has to reaches for a target or gazes at a point in the space making mistakes that need to be compensated. The benefits that derive from this approach are demonstrated in the particular case of robot eye-hand coordination.

TABLE OF CONTENTS

CHAPTER 1

SCIENTIFIC APPROACH

1.1 Motivation

As researchers in the field of humanoid robotics we commonly find ourselves promising our funders that soon we will see the world populated by robots substituting humans in everyday tasks. At the same time – and quite sadly – in our laboratories students spend an enormous amount of time facing tasks that everyone considers quite easy or at least "already solved" in the literature. The problem is that, if on the one hand the scientific literature is full of papers describing techniques that solve virtually all possible tasks, on the other hand it is difficult to find the implementation of those techniques and use them out of the box. This is not to say that scientific papers are incorrect, do not contain good work or are not useful. It is true, however, that researchers put a lot of effort in writing papers and developing new techniques, but rarely concentrate on writing good implementations of these techniques and making them publicly available for comparison purposes or just as building blocks to work out more sophisticated tasks.

In particular when we decided to implement a Cartesian Controller for the *iCub* platform we found that it was not as easy as we initially expected. In humanoid robotics we often deal with kinematics structures that have a large (and variable) number of degrees of freedom. The problem is further complicated because trajectories have to be computed quickly in real-time. Finally, humanoid robots are programmed to produce smooth movements. All these aspects rule out the possibility to resort to expensive off-line solutions that are typically employed in industrial settings (Sciavicco and Siciliano 2005).

To deal with these issues we designed a novel Cartesian Controller for reaching and gazing tasks as detailed in Chapter 2 and Chapter 3 respectively, extending the Multi-

Referential dynamical system approach (Hersch and Billard 2008) in two aspects. Firstly we have modified the trajectory generator to produce trajectories that have minimum-jerk profile. Secondly, we have applied an interior point optimization technique (Wätcher and Biegler 2006) to solve the inverse kinematics problem in real-time. We have shown that our solution has some advantages with respect to (Hersch and Billard 2008) and standard approaches in the literature (Chiaverini, Siciliano and Egeland 1994), (Liegeois 1977). We also conducted experimental comparisons between our implementation and publicly available software, demonstrating the performance gain of our technique in terms of smoothness, speed, repeatability and robustness.

The choice of basically employing conventional methods taken from the control theory was also motivated by the need of partially reviewing the current trend in robotics. In fact, over the past years studies in developmental robotics such as the work of (D'Souza, Vijayakumar and Schaal 2001), (Rolf, Steil and Gienger 2010) and (Wrede, et al. 2010), to cite a few examples mainly involving the inverse kinematic problem, have been focusing more and more on the possibility to learn high dimensional maps for many different motor tasks, where the ultimate goal is to discard any kind of a priori physical/geometrical model in favor of the ability to generalize expressed by abstract paradigms widely used in machine learning. Despite the appealing property of being top-down approaches that essentially require only a minimal description of the environment, it is a matter of fact that these paradigms suffer from a number of somewhat critical weaknesses, mainly regarding engineering aspects such as performance, repeatability and reliability issues. Figure 1-1 attempts to summarize the dichotomy between traditional control theories and learning applied to robotics. On one side the conventional approach depicts the robot brain as a hardwired machine, coded fairly in advance, merely processing the incoming inputs according to the preconceived model of the environment; on the opposite side, the robot brain is perceived much like a living plant that interacts with the surroundings to gather useful information in order to build up an internal representation of the reality.

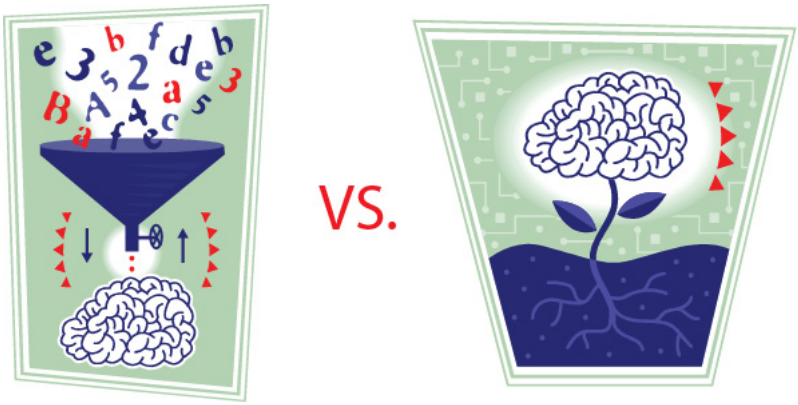

Figure 1-1. The dichotomy between classical methods that rely on *a priori* models and AI paradigms (left) and the developmental approach that predicates the fundamental idea of solving tasks with the experience the system can gain through the interaction with the environment.

The need to sidestep these difficulties naturally gives rise the idea of combining the advantages of "old-fashion" well-established techniques (control theory, nonlinear optimization, etc) with the benefits of learning-oriented methodologies (reservoir networks, support vector machines, locally-weighted regression) into a unique framework that ensures good performance indexes in achieving motor tasks through an accurate model-based design and at the same time enforces the capability of learning the errors and adapting to unmodeled circumstances. This view might shed a new light on the controversial relationship proposed by the sketch in Figure 1-1, where the conflict expressed by the big red “vs.” can be rather replaced by a proper slider bar that modulates from time to time the influence of each component (Figure 1-2).

A clear demonstration of the effectiveness of such collaborative behavior between modeling and learning is presented in Chapter 5, where the problems of adaption to reaching errors and learning of robot eye-hand coordination are tackled starting from a robust model-based layer and then plugging on top simple machine learning techniques.

Interestingly, this practical yet structured "workaround" that breeds from an engineering standpoint in the humanoid robotics can find conversely its conceptual motivation in cognitive sciences and researches in human behavior where the role of

internal representation of the world and the mechanisms that allow humans to react against unpredictable changes are fundamental topics constantly inquired. One of the major key point would be to understand to which extent humans rely on inherent models of their motor activities to then adjust results when perturbations come into play or predictions reveal to be wrong, and furthermore how new models might be built up on the basis of observations, eventually partially or even fully replacing old descriptions of the self motion as well as the environment.

Therefore, inspired by hypotheses and experiments on humans, the perspective of blending model-based and learning-oriented policies in robotics may receive relevant guidelines for future researches.

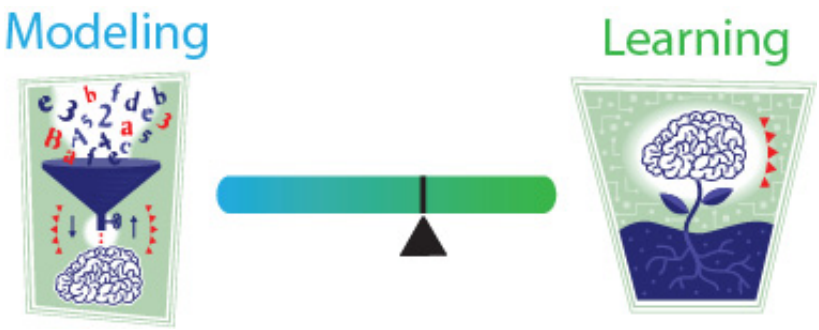

Figure 1-2. A sketch that evokes the proposed combination of modeling and learning as a dynamic slider setting up the correct mixture of ingredients between *a priori* knowledge and learning algorithms.

In this context, a central component turns out to be the feedback: on one hand the feedback allows control systems to compensate for uncertainties and to be robust against disturbances while retaining the skill to accomplish tasks successfully; on the other hand it provides the error signals used by online algorithms to learn the deviations from the model. Once this errors map is properly explored, the closed loop part of the system that is dominant in the early stage can be ideally switched off promoting the feed forward element that corrects the model predictions with the knowledge acquired from the experience. The goal is to achieve tasks objectives more effectively compared to the case

of traditional closed-loop methods as for example to execute limbs movements extremely fast preserving precision, to overcome the limitation imposed by the feedback reading rate.

1.2 Modularity

The term *modularity* represents a key aspect that pervades the whole science in many different fields, and having a fairly general meaning makes it difficult to provide an accurate definition. We can then try to give an explanation by borrowing the idea of (Schilling 2000) for whom "modularity is a general systems concept, typically defined as a continuum describing the degree to which a system's components may be separated and recombined; it thus refers to both the tightness of coupling between components, and the degree to which the rules of the system architecture enable (or prohibit) the mixing and matching of components".

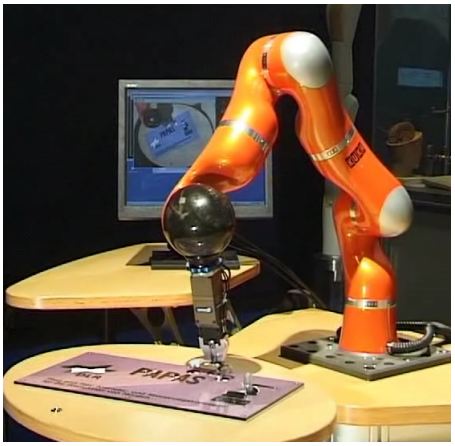

Figure 1-3. The KUKA manipulator. From the basis to the end-effector only one basic mechatronic module is employed to actuate the whole set of the available degrees of freedom.

There exist significant examples of applications of the modularity in diverse contexts: in biology, where the term may be used to refer to organisms that own an indeterminate structure wherein modules of various complexities are assembled without strict limits on

their number or placement such as the plants, or the focus may be not the morphology but rather the functionality of the low-level components as the genes that are able to act in a unified way; in mechanical design and specifically in robotics, where lately dexterous industrial manipulators (Figure 1-3) are conceived in such a way that their realization can rely on one single mechatronic module employed for each degree of freedom equipping the structure; in studies of primate perception of objects, where structural primitives of the ventral stream have been identified (Riesenhuber and Poggio 2000), showing out a general organization of the visual cortex in a series of layers, having a specialization of the simple units to specific functions; in software technology with the well-established design pattern methodology, where the modularity plays an important role guaranteeing the code reuse (the *YARP* middleware belongs to this category); even in mathematics, where the Fourier analysis, the orthogonal polynomials, the Spline interpolation, the Wavelet Multiresolution approximation (adopted in Chapter 6) are all examples of methods that employ particular modules such as sinusoids, orthonormal basis of polynomials, wavelets that properly combined together can describe a rich set of generic maps.

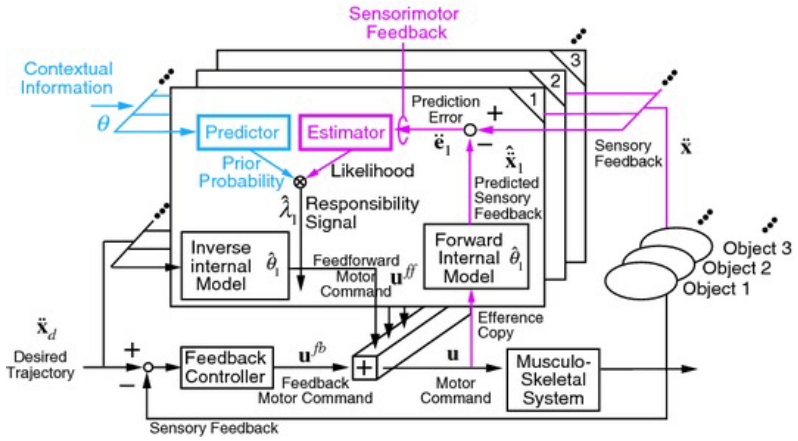

Figure 1-4. A single module of the *MOSAIC* system within the multiple paired internal model.

However, in this thesis the central topic is represented by motor control, and also in this field interesting applications of modularity can be gathered and analyzed. In (Ijspeert, Crespi, et al. 2007) for instance the focus is in using numerical simulations and robots to get a better understanding of animal locomotion and movement control, and in using inspiration from biology to design novel types of robots and locomotion controllers.

Further, in order to replicate in a mirroring manner the human ability to generate accurate and appropriate motor behaviors under many different and often uncertain environmental conditions, the *MOSAIC* model has been proposed (Wolpert and Kawato 1998), whose architecture is intrinsically modular: it is based on multiple pairs of forward (predictor) and inverse (controller) models organized in a competitive way with a set of internal rules subject to a continuous update mechanism capable of selecting the right pair both prior to movement and subsequently during movement (Figure 1-4). The ability to correct online an inappropriate activation of modules while the model is presented with a novel shape-dynamic pairing is encapsulated as well.

It has been extensively showed that notions and ideas from neurophysiological studies can be helpfully employed to provide the direct connection between cognition and intentionality with motor control encapsulated in the robot platform; consequently, to justify the modular approach in motor control one can refer to significant findings highlighted in the work of (Mussa-Ivaldi and Bizzi 2000) in which by analyzing some experiments the authors suggest the existence of motion primitives hardwired in the central nervous system (*CNS*) of frogs and rats. These primitives act on limbs in terms of muscle synergies, called spinal fields. These synergies have been characterized in terms of the isometric force fields elicited at the limb extremity. The main features observed in the experiments are the following:

- Sensory-motor systems of frogs and rats are organized into a finite number of linearly combinable modules, called spinal fields;
- Each spinal field recruits a specific pattern of muscles that direct the limb towards a given configuration, regardless of the initial condition;
- Simultaneous activation of multiple spinal fields leads to the vectorial summation

of the corresponding force fields.

There are therefore evidences that the complex nonlinearities that characterize the limbs of living creatures are somehow eliminated, since the *CNS* acts linearly, applying the superposition principle and showing out at the same time the robustness of control.

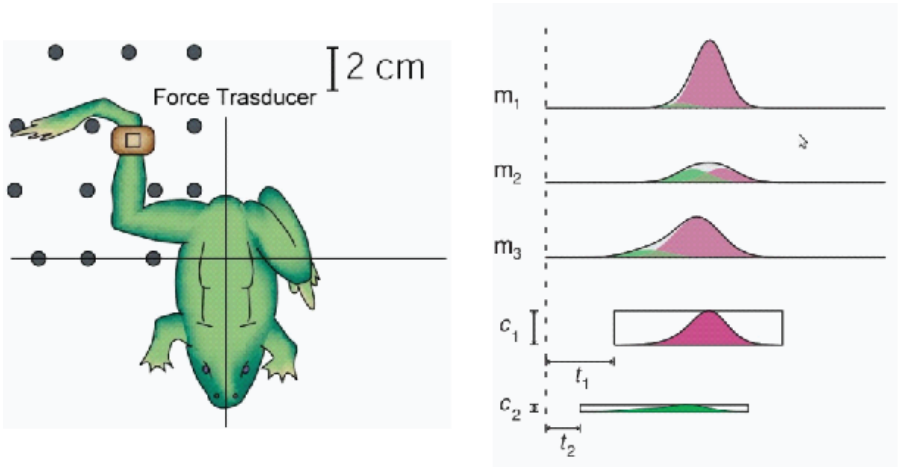

Figure 1-5. Experiments conducted on frogs reveal the underlying modular nature of the Central Nervous System in handling motor commands.

Inspired by these studies, (Nori and Frezza 2005) rethought the experimentally observed motion data under a control theory perspective, trying to answer the following questions: (1) which information about the system dynamics can be represented in a spinal field and how can it be used to perform a given action; (2) how does the *CNS* combine the elementary modules to generate different instances of the same action; (3) how are the modules activated and modified to reject external disturbances and adapt to changes in the system dynamics. Remarkably, the authors managed to identify some relevant properties of the modular control approach modeled in terms of motion primitives, whose final action u is given as linear combination of elementary (but generally time-varying nonlinear) modules $\Phi^1,\dots\Phi^K$ depending on the current state $x\left(t\right)$ and responsible each for driving the system to the corresponding equilibrium points $x_f^1,\dots x_f^K$. They finally demonstrated that the problem of controlling complex systems is

simplified by recruiting two different parts, one (Φ^K) depending only on the system internal states and a second represented by the set of combinators depending only on the task to be executed. On this basis, the learning ability as well as the system robustness can be achieved by appropriately placing the set of Φ^K for performing a given action.

In the context of this thesis the concept of modularity mainly involves two different kinds of topics. The first refers to the high-level hierarchical structure of the Cartesian controllers that are composed of a nonlinear real-time solver and the subsequent minimum-jerk controller, both treating each joint as a elemental module in the kinematic chain, being actuated or not depending on the task; thus, the modularity addresses the requirement of exhibiting the scalability property (the number of degrees of freedom is generic) and the portability property (the robotic platform along with its kinematic chain is generic too). The second aspect is strictly related to the argument of motion primitives inquired by (Nori and Frezza 2005). The last chapter indeed introduces a study on the trajectories encoding, which corresponds to a collection of methods aiming to represent a generic position or velocity profile by mean of coefficients that linearly combine suitable basis functions such as the wavelet functions. With this meaning, as detailed in Chapter 6, the wavelets can be thought to be the basic modules underlying an enhanced version of the designed controllers that instead of commanding instant by instant the system with the actual output computed on the basis of the feedback, describe the desired behavior at once in terms of expansion combinators, in a fully feed-forward manner.

1.3 The *CHRIS* Project

The work conducted for this thesis had as one of the main aims to support the European FP7 ICT project No. 215805 (*CHRIS*)[1] that addresses the fundamental issues which would enable safe Human Robot Interaction (*HRI*). Specifically this project deals with the problem of a human and a robot performing co-operative tasks in a co-located space where it is crucial to handle communication of a shared goal (verbally and through

[1] http://www.chrisfp7.eu

gesture), perception and understanding of intention (from dexterous and gross movements), cognition necessary for interaction, and active and passive compliance.

The project is based on the essential premise that it will be ultimately beneficial to the socioeconomic welfare to generate service robots capable of safe co-operative physical interaction with humans. The key hypothesis is that safe interaction between human and robot can be engineered physically and cognitively for joint physical tasks requiring co-operative manipulation of real world objects.

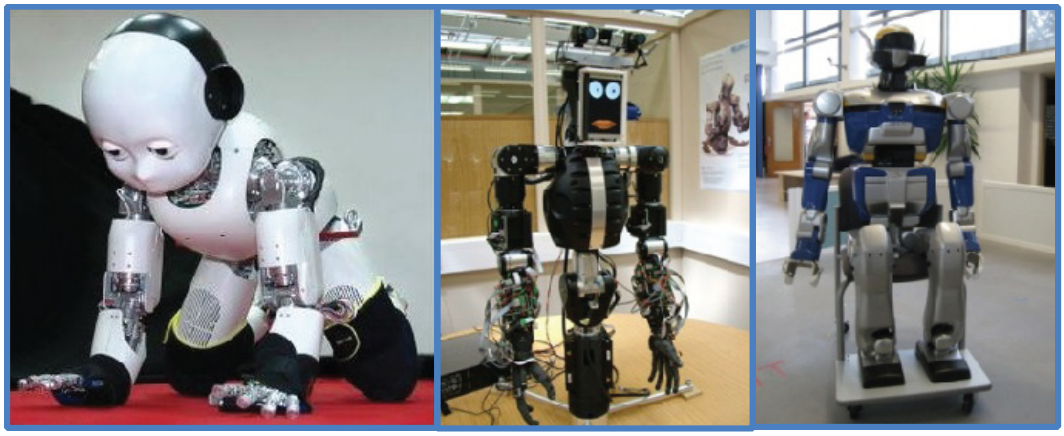

Figure 1-6. The robotic platforms used within the *CHRIS* project. From left to right: *iCub*, *BERT2*, and *HRP2*.

A diverse set of disciplines have been brought together to realize an inter-disciplinary solution. The starting point for understanding cooperative cognition is from the basic building blocks of initial interactions, those of young children. Engineering principles of safe movement and dexterity have been explored on the three available robot platforms (see Figure 1-6) hosted by the project partners[2], and developed with principles of

[2] The partners of the *CHRIS* project are: (1) *Istituto Italiano di Tecnologia* (*IIT*) in Genoa, Italy; (2) *University of West England* (*UWE-BRL*) in Bristol, UK; (3) *University of Bristol* (*UoB-BRL*) in Bristol, UK; (4) *Centre National de la Recherche Scientifique* (*CNRS*) in Toulouse, France; (5)

language, communication and decisional action planning where the robot reasons explicitly with its human partner.

In particular, in the context of safe *HRI* it is essential that robots have the capability to perceive their environment, extrapolate information and utilize this to move safely around their environment and in coordination with their human counterpart. To achieve that, it is required to investigate the construction of a controller for a humanoid robot which includes learning from examples while retaining safety. The controller must be *modular* in the extent exposed in the previous paragraph, so that a higher level decisional planner can appropriately sequence and rely on it to perform more complicated goals.

1.4 The *iCub* Platform

Even though the approach presented in this thesis owns as its primary goal the requirement to generically address any direct and inverse kinematic problem with serial link chains, the experiments reported herein were performed mainly on the *iCub* platform (Metta, Vernon, et al. 2008), a humanoid robot (Figure 1-7) shaped as a human child with 53 degrees of freedom. The *iCub* was designed to investigate cognitive robotics and particularly to study manipulation so most of the mechanical complexity is in the arms and hands, which are actuated by a total of 16 motors each (9 and 7 in the hand and arm respectively). The *iCub* is equipped with cameras, force sensors, and gyroscopes. All the software running on the *iCub* – including the software specifically developed during the thesis work – is released open-source (*GPL*) and is freely available for download with the aim to advance the research in robotics by sharing algorithms, results and accumulate an open archive of experimental facilities.

Max Planck Gesellschaft zur Foerderung der Wissenschaften (*MPG*) in Leipzig, Germany; (6) *Institut National de la Santé et de la Recherche Médicale* (*INSERM*) in Lyon, France.

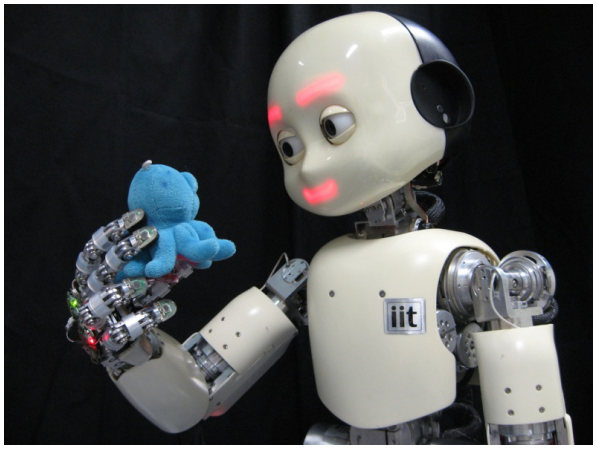

Figure 1-7. The *iCub* robot.

1.5 The *YARP* Middleware

Software modules in the architecture are interconnected using *YARP* (Fitzpatrick, Metta and Natale 2007), an open source library written to support software development in robotics. In brief *YARP* provides an intercommunication layer that allows processes running on different machines to exchange data. Data travels through named connection points called ports. Communication is platform and transport independent: processes are not aware of the details of the underlying operating system or protocol and can be relocated at will across the available machines on the network (Figure 1-9).

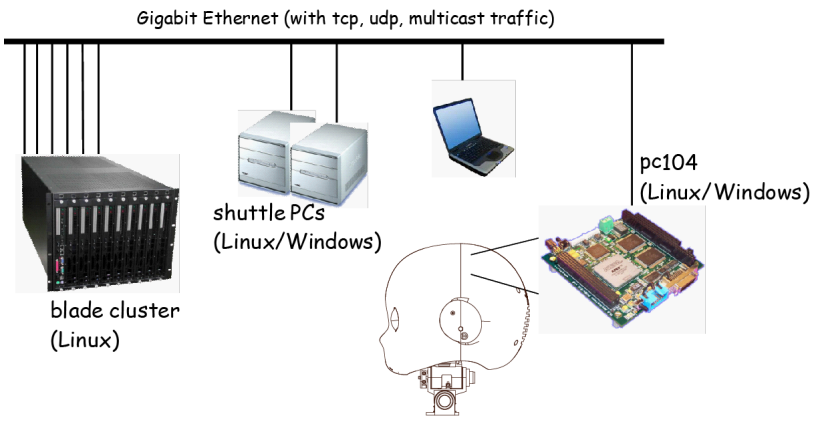

Figure 1-8. The physical network of machines running *YARP* modules that control the robot.

More importantly, since connections are established at runtime it is easy to dynamically modify how data travels across processes, add new modules or remove existing ones. Interface between modules is specified in terms of *YARP* ports (i.e. port names) and the type of data these ports receive or send (respectively for input or output ports). This *modular* approach allows minimizing the dependency between algorithm and the underlying hardware/robot; different hardware devices become interchangeable as long as they export the same interface.

Finally, *YARP* is written in C++, so it is normally used as a library in C++ code. However, any application that has a TCP/IP interface can talk to *YARP* modules using a standard data format. Within the *CHRIS* project this turned out to be of fundamental importance as it allowed to "glue" together different applications into a single integrated, working system.

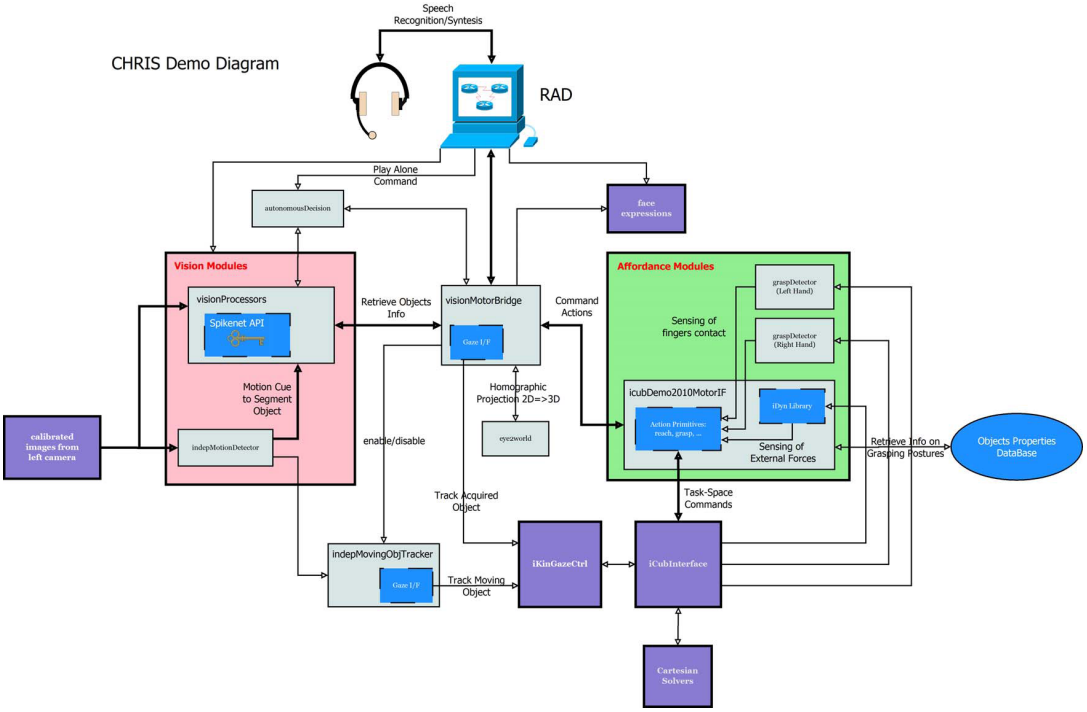

Figure 1-9. An example of complex flow chart designed specifically for the *CHRIS* project, containing *YARP* modules along with their allocation and connections.

1.6 Thesis Outline

The remaining is organized as follows: Chapter 2 deals with the implementation details of the Cartesian controller employed to execute reaching task, whereas Chapter 3 inherits the same principles treating the gaze control problem; Chapter 4 introduces the important elements of software development that took the majority of the efforts of this thesis aiming to provide the modular architecture to be used by all the partners; in Chapter 5 two experiments are analyzed where the devised model-based components and machine learning are profitably used in combination to tackle the errors compensation issue, and specifically the adaption to reaching unknown offsets and the robot eye-hand markerless coordination; finally, Chapter 6 reports a method for encoding trajectory that resorts to the wavelet Multiresolution approximation with the ultimate goal to perform motor control at high level in a feed-forward fashion.

CHAPTER 2

THE CARTESIAN CONTROLLER

2.1 Introduction

The problem addressed in this chapter regards in its most general formulation the design of a controller capable of steering the robot end-effector to track a desired trajectory as sketched in Figure 2-1.

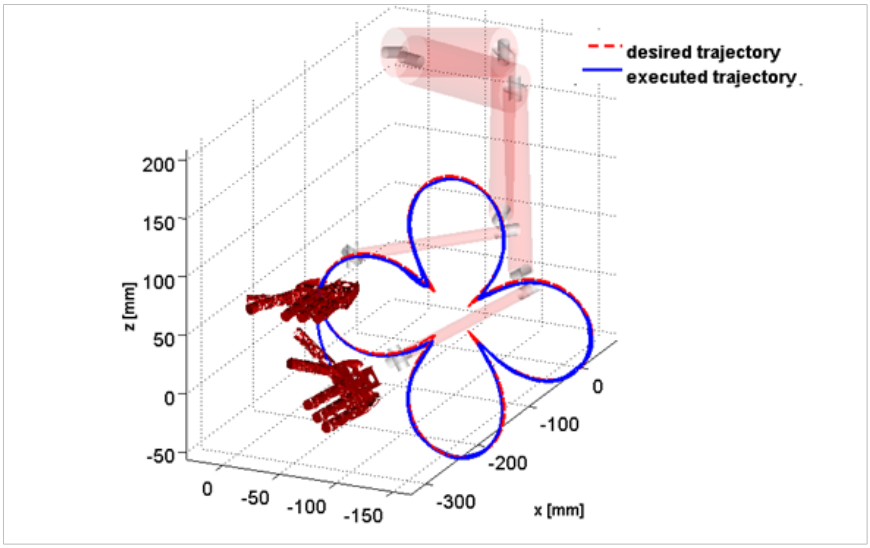

Figure 2-1. Tracking a desired trajectory with a lemniscate shape in the operational space. For better understanding the figure shows two configurations of the arm: the lower one depicts the starting pose, whilst the upper one shows the commanded hand orientation during the task.

Given the Cartesian position of a target object, reaching is performed in two separate modules (Figure 2-2). The first stage employs a *nonlinear optimization technique* to

determine the arm joints configuration that achieves the desired pose (i.e. end-effector position and orientation). The second stage consists of a *biologically inspired kinematic controller* that computes the velocity of the motors to produce a human-like quasi-straight trajectory of the end-effector. In the following these two modules composing the structure of the proposed Cartesian controller are analyzed in depth.

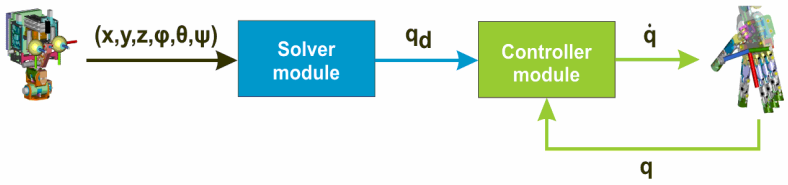

Figure 2-2. Diagram of proposed Cartesian controller.

2.2 The Solver: the IpOpt Solution

We consider the general problem of computing the value of joint encoders $q^* \in \mathbb{R}^n$ in order to reach a given position $x_d \in \mathbb{R}^3$ and orientation $\alpha_d \in \mathbb{R}^4$ of the end-effector (where α_d is represented in axis/angle notation[3]). At the same time, the computed solution has to satisfy a set of given constraints expressed as inequalities. Formally this problem can be stated as follows:

$$q^* = \arg\min_{q\in\mathbb{R}^n} \left(\left\| \alpha_d - K_\alpha(q) \right\|^2 + \beta \cdot (q_{\text{rest}} - q)^T W (q_{\text{rest}} - q) \right)$$
$$\text{s.t.} \begin{cases} \left\| x_d - K_x(q) \right\|^2 < \varepsilon \\ q_L < q < q_U \end{cases}, \tag{2.1}$$

[3] In axis/angle representation any rotation is described by a unit vector ω, indicating the direction of rotation, and an angle θ that accounts for the magnitude of rotation around the axis according to the right-hand rule.

Hence it holds: $\alpha_d = \begin{bmatrix} \omega & \vartheta \end{bmatrix}^T$, $\omega \in \mathbb{R}^3$, $\|\omega\| = 1$, $\theta \in [0, \pi]$

where K_x and K_α are the forward kinematic functions that respectively compute the position and the orientation of the end-effector from the joint angles q; q_{rest} is a preferred joint configuration[4], W is a diagonal matrix of weighting factors, β is a positive scalar weighting the influence of q_{rest} and ε is a parameter for tuning the precision of the movement: typically $\beta < 1$ and $\varepsilon \in \left[10^{-5}, 10^{-4}\right]$. Moreover, the solution to problem (2.1) has to comply with a set of additional constraints: for example, we required that the solution lies between lower and upper bounds ($q_L, q_U \in \mathbb{R}^n$) of physically admissible values.

In our case the joints vector has 10 components (7 joints for the arm, 3 joints for the torso) and we have chosen the value of q_{rest} so that the torso of the robot when controlled is as close as possible to the vertical position. We proposed to use an interior point optimization technique to solve the problem(2.1), in particular we used *IpOpt* (Wätcher and Biegler 2006), a public domain software package designed for large-scale nonlinear optimization.

This approach owns the following advantages:

1. **Quick convergence**. *IpOpt* is reliable and fast enough to be employed in real-time as demonstrated in the remainder, especially compared to more traditional iterative methods such as the Cyclic Coordinate Descent (*CCD*) (Wang and Chen 1991) adopted by Hersch et al.

2. **Scalability**. The intrinsic capability of the optimizer to treat nonlinear problems in any arbitrary number of variables is here exploited to make the controller's structure easily scalable with the dimension n of the joint space. For example, it is possible to switch at run-time from the control of the 7-DOF *iCub* arm to the complete 10-DOF

[4] Notably, when q_{rest} remains unspecified within the task, it can be suitably employed to minimize also the displacement with respect the current joint configuration in order to take into account energy consumption considerations.

structure inclusive of the torso or to any combination of the joints depending on the task.

3. **Automatic handling of singularities and joint limits**. This technique automatically deals with singularities in the arm Jacobian and joint limits, and can find solutions in virtually any working conditions.

4. **Tasks hierarchy**. The task is split in two subtasks: the control of the orientation and the control of the position of the end-effector. Different priorities can be assigned to the subtasks. In our case the control of the position has higher priority with respect to the orientation subtask (the former is handled as a nonlinear constraint and thus is evaluated before the cost) because we deemed that to accomplish a successful grasp which is the ultimate goal of reaching for a humanoid it is central to cope with circumstances when the final object is attainable in position and thus can be touched, but the orientation cannot be reached perfectly at the same time.

5. **Description of complex constraints**. It is easy to add new constraints as linear and/or nonlinear inequalities either in task or joint space. In the case of the *iCub*, for instance, we added a set of constraints that avoid breaking the tendons that actuate the three joints of the shoulder: these tendons indeed are shared among the joints, whose movements are thus limited by the tendons lengths within a compact subset of the convex hull described by the physical joints bounds (Parmiggiani, et al. 2009). Thereby, three linear inequalities hold among the shoulder joints that are conveniently included into (2.1) taking the form $l \leq C \cdot q_{\mathrm{sh}} \leq L$, being q_{sh} the vector of the three shoulder joints, *l*, *L* the lower and upper limits imposed by the tendons lengths and *C* a suitable 3-by-3 matrix.

2.3 The Controller: Design of Minimum-Jerk Solution

The goal of the controller's module is to determine the smooth velocity profiles in the joint space which steer the arm from the current posture q to the final configuration q^*,

while at the same time ensuring that the joints lie within well defined limits. This can be obtained by applying the *Multi-Referential Dynamical Systems* approach (Hersch and Billard, Reaching with Multi-Referential Dynamical Systems 2008), in which two dynamical controllers, one in joint space and one in task space, evolve concurrently (Figure 2-3). The coherence constraint between the two tasks – providing that $\dot{x} = J\dot{q}$ is guaranteed at each instant with J the Jacobian of the forward kinematic map – is enforced with the Lagrangian multipliers method and can be used to modulate the relative influence of each controller (i.e. to avoid joint angles limits). The advantage of such a redundant representation of the movement is that a quasi-straight trajectory profile can be generated for the end-effector in the task space reproducing a human-like behavior (Abend, Bizzi and Morasso 1982), (Flash and Hogan 1985), while retaining converge property and robustness against singularities (Hersch and Billard, Reaching with Multi-Referential Dynamical Systems 2008).

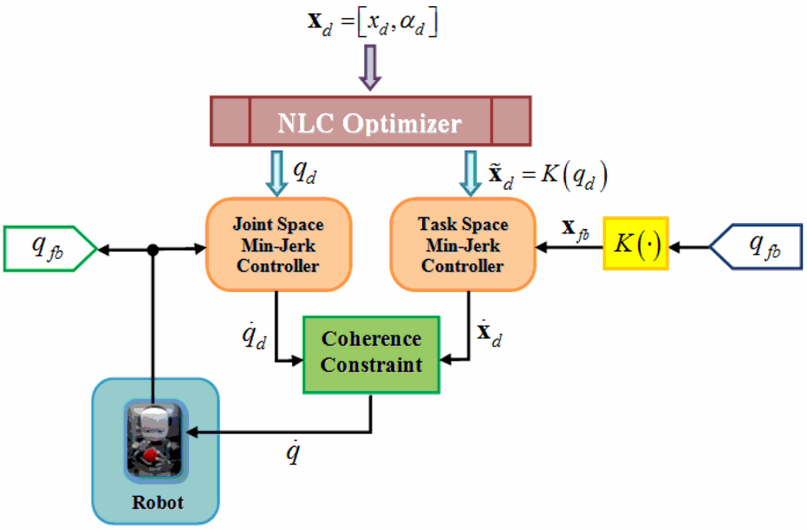

Figure 2-3. Multi-Referential scheme. $K(\cdot)$ is the forward kinematic map.

In (Hersch and Billard, Reaching with Multi-Referential Dynamical Systems 2008) the two controllers are implemented with Vector-Integration-To-Endpoint (*VITE*) models

(Bullock and Grossberg 1988), which approximate the neural signal commanding a pair of agonist-antagonist muscles and whose behavior is regulated by a second order differential equation as follows:

$$\begin{cases} \ddot{q} = \alpha \cdot \left(\beta \cdot \left(q_d - q \right) - \dot{q} \right) \\ \ddot{x} = \alpha \cdot \left(\beta \cdot \left(x_d - x \right) - \dot{x} \right) \end{cases}, \tag{2.2}$$

where the first equation of (2.2) holds in the joint space, whereas the second equation holds in the task space; moreover, α is the damping factor and $\alpha \cdot \beta$ is the stiffness.

According to the implementation in (Hersch and Billard, Reaching with Multi-Referential Dynamical Systems 2008), the angular velocities, output of the coherence constraint block, are integrated to generate position references which are then sent to a second cascade controller that is in charge of yielding the velocity profiles in closed loop with a proportional law (Figure 2-4).

Aside from the connection to biological evidences, a second significant merit of this approach is the description of the model given as a compact and autonomous dynamic equation which makes the controller implementation straightforward and robust against external perturbation of the movement creating an attractor landscape towards the goal, i.e. the target configuration (Ijspeert, Nakanishi and Schaal, Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots 2002). On the other hand, the specific choice of a coupled second order dynamic systems in cascade with a P controller entails a major disadvantage when applied to the control of a robotic limb: notably, the generated velocity profiles become less human-like as the required execution time becomes shorter. When a fast response is requested, trajectories approach an exponential response (typical of a first order dynamical system), irrespectively of how the controller's parameters are tuned; therefore, the corresponding velocities are no longer bell-shaped, having a steep acceleration at the beginning followed by a slow decay. The reason is twofold: primarily a second order system cannot reproduce the smoothness typical of biological motion (Flash and Hogan 1985) (for example it does not impose zero

acceleration at starting point) and secondly the presence of the proportional controller reduces the performances since it cannot guarantee the velocities computed by the coherence constraint block. As result fast movements tend to be jerky producing unwanted vibrations.

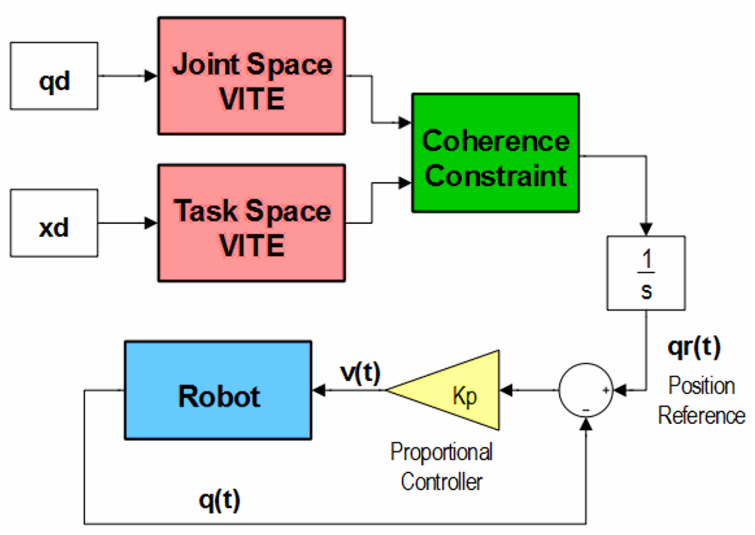

Figure 2-4. Schematic of implemented Multi-Referential *VITE* controllers in the work of (Hersch and Billard, Reaching with Multi-Referential Dynamical Systems 2008).

To overcome these limitations, we maintained the multi-referential approach and replaced the *VITE* with more complex controllers which reproduce a trajectory that resemble a minimum-jerk profile both in joint and task space: movements are still represented and controlled in multiple frames of reference but preserve a smooth (bell shaped) velocity profile. To this end one might consider to rely on a trajectory generator which codes for example the minimum-jerk profile over the time interval T and specific starting and ending points x_0, x_d : in literature (Hoff and Arbib 1992) it is well known that the desired shape depicted in Figure 2-5 (red lines) is given by the following fifth order polynomial:

$$x(t) = x_0 + (x_d - x_0) \cdot \left(10\left(\frac{t}{T}\right)^3 - 15\left(\frac{t}{T}\right)^4 + 6\left(\frac{t}{T}\right)^5 \right). \quad (2.3)$$

The seeming straightforwardness of this formulation hides a number of somewhat important issues that need to be taken into account by an effective design: it is required indeed to generate an internal temporal scale t that has to be reinitialized any time the feedback is acquired modifying the coefficients of (2.3) on-line; moreover, the feedback turns out to be mandatory since the coherence block disturbs the true velocity command causing eventually drifts if not compensated by feedback. Therefore, a regulator appears to be a more natural answer for the task and joint space minimum-jerk elements. Possibly the generator (2.3) can be applied as the feed-forward component of the regulator, operating merely on the target position and leaving a PID to take into account the feedback: even so the PID would work hard to stabilize the response against the drift, delivering velocities that do not comply with the requisite of human-likeness. This ultimately suggests devising a controller that intrinsically embeds the desirable property of smoothness.

We took inspiration from the feedback formulation of the minimum-jerk trajectory as the solution of an optimal control problem as reported in (Shadmehr and Wise 2005), where a third order linear time-varying (*LTV*) differential equation is derived:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\frac{60}{(T-t)^3} & -\frac{36}{(T-t)^2} & -\frac{9}{T-t} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{60}{(T-t)^3} \end{bmatrix} x_d. \quad (2.4)$$

The weak point of (2.4) is that it contains coefficients whose values become infinite

when t approaches the execution time T. To sidestep this difficulty, we decided to employ a linear time-invariant (*LTI*) system of the same order whose parameters are tuned to better approximate a minimum jerk trajectory: in other words we sought the third order differential system that is the best time-invariant version of (2.4) minimizing the same jerk measure over the interval $\left[0, T\right]$. Formally, we started from the parametric equation of the trajectory expressed in the form:

$$x\left(t\right) = C_1 \exp\left(\lambda_1 \cdot t\right) + C_2 \exp\left(\lambda_2 \cdot t\right) + C_3 \exp\left(\lambda_3 \cdot t\right) + x_d, \quad (2.5)$$

that is a particular solution of a stable third order differential system with three independent real negative poles λ_i. The selection of real negative roots stems from the objective to identify a stable system and avoid damped resonant terms since we require a monotonic trend to the target, without overshoots, as it comes to be relevant for joints limits avoidance; in addition, oscillating components certainly introduce jerkiness in the response.

On the other hand, the function in (2.5) takes into account only the specific case of three roots with multiplicity 1; the remaining two cases, i.e. (1) one simple and one double root and (2) one root with multiplicity 3, will be treated afterwards in order to gain a complete insight of the problem.

The coefficients C_i can be determined for the special case $x_d = 1$ and $x\left(0\right) = 0$ without any loss of generality, by imposing the following initial conditions:

$$\begin{cases} x_1\left(0\right) = 0 \\ \dot{x}_1\left(0\right) = 0 \\ \ddot{x}_1\left(0\right) = 0 \end{cases} \Rightarrow \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \frac{1}{\left(\lambda_1 - \lambda_2\right) \cdot \left(\lambda_2 - \lambda_3\right)} \cdot \begin{bmatrix} -\lambda_2 \cdot \lambda_3 \\ \lambda_1 \cdot \lambda_3 \\ -\lambda_1 \cdot \lambda_2 \end{bmatrix}. \quad (2.6)$$

Therefore, defined $M_1\left(t\right)$ the measure of the squared jerk $\dddot{x}_1^2\left(t\right)$ accumulated up to time t as:

$$M_1\left(t\right) = \int_0^t \dddot{x}_1^2\left(\tau\right) d\tau, \quad (2.7)$$

we seek for a solution to the following minimization problem:

$$\mathrm{P1}: \ \lambda^* = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \arg\min_{\lambda \in \mathbb{R}^3} \left(M_1\left(\infty\right) \right)$$
$$\text{s.t.} \begin{cases} \lambda_i < 0, \ i = 1,2,3 \\ \lambda_i \neq \lambda_j, \ i \neq j \\ x_1\left(1\right) \geq 1 - \varepsilon_1 \end{cases} . \quad (2.8)$$

The first constraint in P1 imposes that the system is stable, whereas the second constraint requires the independence of all the roots, that is strictly necessary to solve the linear system for the coefficients and, further, allows us to write the solution as in (2.5). These latter nonlinear bounds can be profitably simplified by resorting to suitable continuously differentiable inequalities such as:

$$\exp\left(-\frac{\left(\lambda_i - \lambda_j\right)^2}{\sigma^2} \right) \leq \varepsilon_2, \quad (2.9)$$

which, depending on the values assigned to σ and ε_2, guarantees that the roots are non coincident (e.g. $\sigma = 10^{-3}$ and $\varepsilon_2 = 0.1$ imply $\left|\lambda_i - \lambda_j\right| > 10^{-3}$).

Finally, the third constraint in P1 forces the solution to reach the steady-state value of 1 with a "rate" specified by the parameter ε_1. Without this lower bound on $x_1\left(1\right)$, any possible monotonically increasing function would be allowed, even those functions with very slow time constants. In other words, by setting the parameter ε_1 we are able to tune the final execution time which, in our case, will be as close as possible to 1. It is worth to point out that the quantity $M_1\left(\infty\right)$ in (2.8) can be easily resolved as function of the

roots since the (2.5) is known, but more importantly its gradient has a close formula in λ_i that enables to carry out the minimization with reliable gradient-based methods.

To complete the treatment the method has to be applied also to the remaining two cases:

1) *One single root* λ_1 *and one double root* λ_2. The general solution takes the well known form:

$$x_2(t) = C_1 \cdot \exp(\lambda_1 \cdot t) + \exp(\lambda_2 \cdot t) \cdot (C_2 + C_3 \cdot t). \qquad (2.10)$$

Thus the problem is modified slightly in:

$$\mathrm{P2}: \ \lambda^* = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \arg\min_{\lambda \in \mathbb{R}^2} \left(M_2(\infty) \right)$$
$$\text{s.t.} \begin{cases} \lambda_1 < 0, \ \lambda_2 < 0 \\ \lambda_1 \neq \lambda_2 \\ x_2(1) \geq 1 - \varepsilon_1 \end{cases} . \qquad (2.11)$$

2) *One root* λ *with multiplicity 3*. As consequence we have:

$$x_3(t) = \exp(\lambda \cdot t) \cdot \left(C_1 + C_2 \cdot t + C_3 \cdot t^2 \right). \qquad (2.12)$$

The problem simplifies accordingly in:

$$\mathrm{P3}: \ \lambda^* = \arg\min_{\lambda \in \mathbb{R}} \left(M_3(\infty) \right)$$
$$\text{s.t.} \begin{cases} \lambda < 0 \\ x_3(1) \geq 1 - \varepsilon_1 \end{cases} . \qquad (2.13)$$

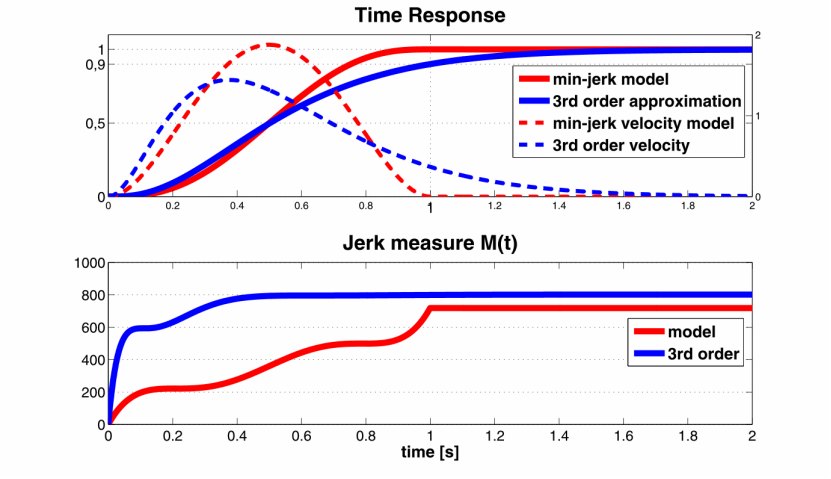

Figure 2-5. Comparison between the responses of the 3rd order dynamical time-invariant system found through the minimization (blue) and the responses of the minimum-jerk model (red).

We ran then an optimization algorithm to solve P1, P2, and P3 for several values of the parameter ε_1 relying on an interior-point algorithm as well: in Table 2-1 for each problem the resulting roots are reported along with the ratio $M_i(\infty)/M_{\text{ideal}}(\infty)$ that gives a measure of quality of our estimate in terms of amount of jerk with respect to the ideal model. By inspecting the outcomes, it follows that the three roots, solution of P1, tend to collapse into the root of multiplicity 3 which solves the problem P3 and converges on the best result; conversely, an approach based on P2 does not lead to good upshots.

Figure 2-5 compares the trajectories (position, velocity and the measure $M(t)$) of the ideal minimum-jerk model against those obtained with the time-invariant system derived with our approach by selecting $\varepsilon = 0.1$ (we retrieved $\lambda^* = -5.322$). As expected the *LTI* system approximates the ideal minimum-jerk position trajectory, having in particular a slightly faster onset followed by slower convergence to the steady state. At the same time, however, it provides a very good compromise between smoothness and simplicity of implementation.

ε_1	λ_i			$M_i(\infty)/M_{\text{ideal}}(\infty)$		
	P1	P2	P3	P1	P2	P3
$(x_i(1) \geq 0.99)$	$\begin{bmatrix} -8.417 \\ -8.403 \\ -8.399 \end{bmatrix}$	$\begin{bmatrix} -16.140^{(2)} \\ -5.424 \end{bmatrix}$	$-8.406^{(3)}$	10.886	65.512	10.929
$(x_i(1) \geq 0.95)$	$\begin{bmatrix} -6.333 \\ -6.278 \\ -6.275 \end{bmatrix}$	$\begin{bmatrix} -11.687^{(2)} \\ -3.774 \end{bmatrix}$	$-6.295^{(3)}$	2.576	11.166	2.575
$(x_i(1) \geq 0.9)$	$\begin{bmatrix} -5.339 \\ -5.319 \\ -5.309 \end{bmatrix}$	$\begin{bmatrix} -9.727^{(2)} \\ -3.052 \end{bmatrix}$	$-5.322^{(3)}$	1.112	4.014	1.112

Table 2-1. Results of optimization carried out on the problems P1, P2, and P3. The roots multiplicity is indicated in the superscript.

Once the root λ^* is known, it is possible to compute the elements of the dynamic matrix A and input matrix B and write the controller in the canonical form extended to the case of generic execution time T:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dddot{x} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \dfrac{a(\lambda^*)}{T^3} & \dfrac{b(\lambda^*)}{T^2} & \dfrac{c(\lambda^*)}{T} \end{bmatrix}}_{A} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ -\dfrac{a(\lambda^*)}{T^3} \end{bmatrix}}_{B} x_d \quad (2.14)$$

The coefficients $a(\lambda^*), b(\lambda^*), c(\lambda^*)$ can be computed from the expansion of the characteristic polynomial of A in the case $T = 1\,\text{s}$:

$$\left(\lambda - \lambda^*\right)^3 \equiv \lambda^3 - c\left(\lambda^*\right)\lambda^2 - b\left(\lambda^*\right)\lambda - a\left(\lambda^*\right) \quad (2.15)$$

Note how the unique non-null element of B is completely determined for convergence reasons and equal to the opposite of $a\left(\lambda^*\right)\Big/T^3$.

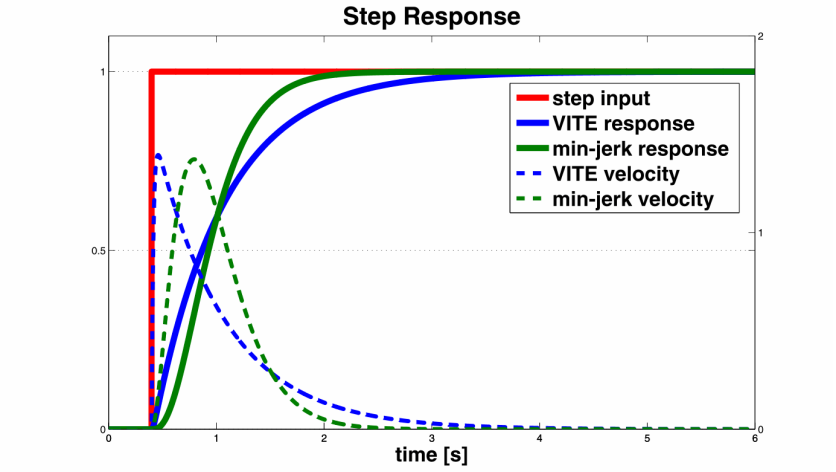

Figure 2-6. Comparison between step responses of *VITE* (blue) and minimum-jerk controller (green), providing the same velocity peak. The 3rd order system has a faster convergence and an (almost) bell-shaped velocity profile.

The system response in $t = T$ is equal to the 90% of the steady-state value as enforced by the constraint, and the transient can be considered extinguished for $t \geq 1.5 \cdot T$.

The first important result achieved by this controller is visible in Figure 2-6: it is clear that minimum-jerk controllers can provide, especially for fast movements, smooth velocity profiles that are more similar to the desired human-like prototypes if compared to the profiles generated by the *VITE* model.

2.4 Implementation Issues

The algorithm follows faithfully the three steps in (Hersch and Billard, Reaching with

Multi-Referential Dynamical Systems 2008) that are briefly resumed hereafter:

1) At any time instant the Multi-Referential controller receives the desired target found by the optimization process, the current value of q_{fb} (and $x_{fb} = K\left(q_{fb}\right)$) and computes the corresponding velocity profiles in the joint space $\dot{q}^d$ and task space $\dot{x}^d$ by integrating the *VITE* models (2.2).

2) Since the two controllers evolve independently, the trajectories are unlikely to satisfy the kinematic constraint given by $\dot{x}^d = J\dot{q}^d$. The coherence is thus enforced by computing the joint velocities that solve the following minimization problem which is proved to have a single minimum since it reveals to be a positive quadratic optimization problem under linear constraint:

$$\min_{\dot{q}_t, \dot{x}_t} \frac{1}{2}\left(\left(\dot{q}_t - \dot{q}_t^d\right)^T W_q \left(\dot{q}_t - \dot{q}_t^d\right) + \left(\dot{x}_t - \dot{x}_t^d\right)^T W_x \left(\dot{x}_t - \dot{x}_t^d\right)\right) \quad \text{s.t.} \quad \dot{x}_t = J\dot{q}_t; \tag{2.16}$$

By applying the Lagrangian multipliers method we can compute a closed form solution:

$$\dot{q}_{t+1} = \dot{q}_t^d + W_q^{-1} J^T \left(W_x^{-1} + J W_q^{-1} J^T\right)^{-1} \left(\dot{x}_t^d - J\dot{q}_t^d\right) \tag{2.17}$$

with W_q and W_x appropriate semi-definite diagonal matrices as defined in step 3.

3) The matrices W_q and W_x can be profitably used to give different importance to the joint or task space constraints. This can be exploited for example to implement a mechanism for joints limits avoidance. Normally W_q and W_x are set so that the task space constraint has higher priority and the arm follows a straight path. When one of the joints approaches a limit, however, W_q is increased so that the joint space

constraint assumes more importance. Formally when the i^{th} joint angle is within the bounds the corresponding weight $w_q^i \in W_q$ is close to zero; conversely, when the arm gets closer to one of the joint limits, the element w_q^i becomes larger, until, eventually, the ratio w_x / w_q^i goes to zero. In the latter situation, since the controller evolves in a convex space, the arm is guaranteed to respect the joint limits: for this reason it is crucial that the controllers produce monotonically increasing trajectories without overshoots.

In (Hersch and Billard, Reaching with Multi-Referential Dynamical Systems 2008) this modulation is achieved by imposing that at each time instant the following relation holds:

$$\frac{w_x}{w_q^i} = \frac{1}{2}\gamma\left(1 - \cos\left(2\pi \cdot \frac{q^i - q^i_{\min}}{q^i_{\max} - q^i_{\min}}\right)\right), \qquad (2.18)$$

where γ is a convenient normalization constant whose value is typically around 0.01. Furthermore, a possible strategy for modulating the weights is setting W_x^{-1} equal to identity matrix and W_q^{-1} equal to the right-hand side of (2.18).

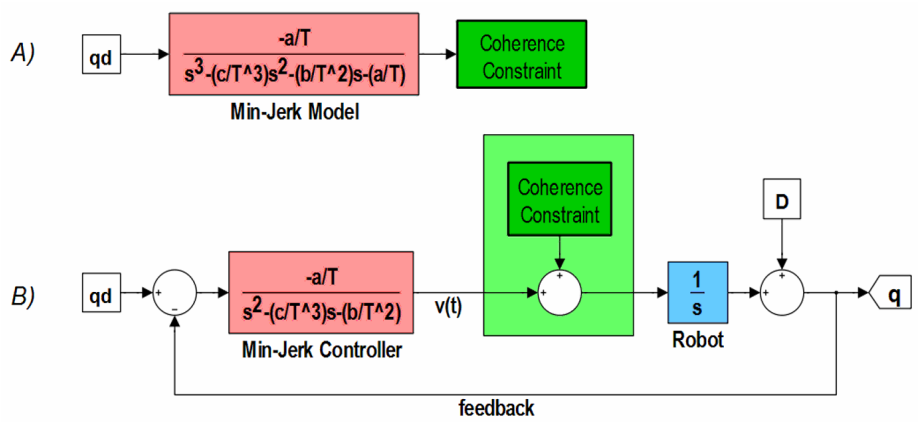

Figure 2-7. *A)*: transfer function of the LTI minimum-jerk model. *B)*: an insight of the joint space minimum-jerk controller implemented in closed-loop form.

In our implementation two main exceptions arise with respect to the aforementioned steps, which are reported below:

1. The need to constantly read the feedback q_{fb} motivated the authors of (Hersch and Billard, Reaching with Multi-Referential Dynamical Systems 2008) to introduce a P controller with the purpose to keep the generation of trajectories and their execution as two separated functionalities, preventing the evolution of the feedback from interfering with the internal state of the *VITE*. This brought about a series of drawbacks we have analyzed in section 2.3.

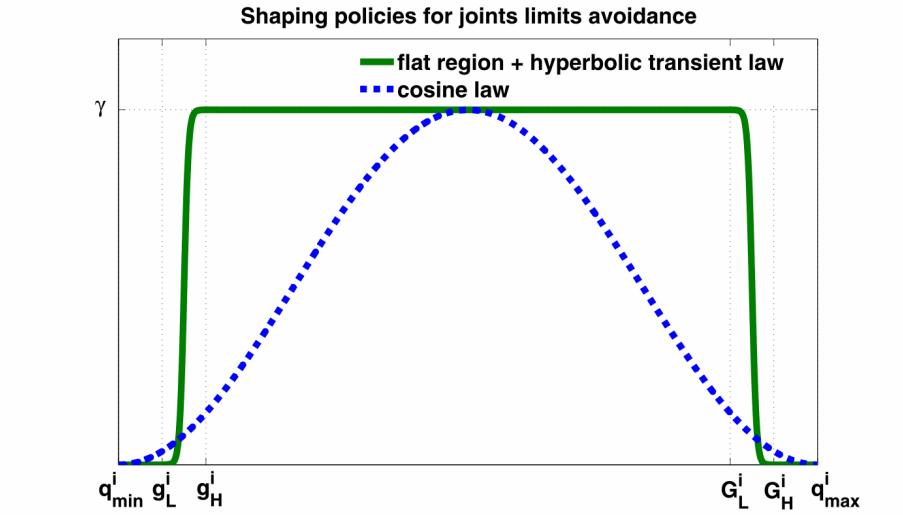

Figure 2-8. The implemented shaping policy for the joints limits avoidance (green) and the original cosine law (blue) used in Hersch et al.

In order to adhere to the original diagram of Figure 2-3, an alternative solution has been explored that transforms the structure of our model as represented in Figure 2-7 (*B*), where for sake of clearness only the joint space minimum-jerk controller is presented: from case *A* that corresponds to the state-space model in (2.14) we migrated to the system *B* that owns exactly the same transfer function $q(s)/q_d(s)$ written in *A*, taking now the actual error between the target and the feedback as input. The pure integrator plays the

role of the mechanical plant that integrates the received command and returns the current feedback. All the remaining unmodeled dynamics and uncertainties are represented by the term D, whose effects are rejected by the closed-loop system. The disturbance introduced in the minimum-jerk controller by the coherence constraint is compensated similarly: as a matter of fact, the signal computed through the Lagrangian multipliers does not act like a feed-forward component, but rather perturbs the controller. To conclude, the closed-loop structure realizes exactly the scheme of the multi-referential approach and ensures that the current robot's position is correctly fed back in the system.

2. The modulation of weighting matrices that appear in the coherence reinforcement and serve for the handling of joints limits avoidance is achieved by imposing a cosine shaping relation. Nonetheless, to better exploit the whole arm workspace it is advisable to assign high priority to the Cartesian controller in a portion of the joint space that is as large as possible. To this end we adopted a different weighting policy made of a flat region connected with hyperbolic tangent functions whose decay rate is much steeper than the original cosine law (see (2.19) and (2.20)), as illustrated in Figure 2-8, showing out its benefits in the execution of quasi-straight Cartesian trajectories for point-to-point motion as demonstrated by experiments.

$$\frac{w_x}{w_q^i} = \begin{cases} \dfrac{1}{2}\gamma \cdot \left(1 + \tanh\left(10 \cdot \dfrac{q^i - \overline{g}^i}{d^i}\right)\right), & g_L^i < q^i < g_H^i \\ \dfrac{1}{2}\gamma \cdot \left(1 - \tanh\left(10 \cdot \dfrac{q^i - \overline{G}^i}{d^i}\right)\right), & G_L^i < q^i < G_H^i \\ \gamma, & g_H^i < q^i < G_L^i \\ 0, & \text{elsewhere} \end{cases} \quad (2.19)$$

with:

$$\begin{cases} d^i = \dfrac{\rho \cdot \left(q^i_{\max} - q^i_{\min} \right)}{4} \\ g^i_L = q^i_{\min} + d^i;\ g^i_H = g^i_L + d^i;\ \overline{g}^i = \dfrac{g^i_L + g^i_H}{2} \\ G^i_H = q^i_{\max} - d^i;\ G^i_L = G^i_H - d^i;\ \overline{G}^i = \dfrac{G^i_L + G^i_H}{2} \end{cases} . \qquad (2.20)$$

2.5 Experimental Results

According to the RobotCub open-source philosophy the whole software developed by the project partners was made available to the wide community of *iCub* users; this facilitates the collaboration and promotes the development of new algorithms. As result, a collection of libraries and modules targeting different fields (vision processing, motor development, machine learning, etc) has been circulating among partners of the RobotCub Consortium who can easily reuse code for their research activities without having to be concerned with the implementation details. In this respect it has been almost immediate and much valuable comparing on the same shared platform the performance of our Cartesian controller[5] with the *VITE*-based system[6] whose modules can be downloaded from public repositories. Additionally, we included in the assessment a further controller as an example of a more conventional strategy making use of the typical Damped Least-Squares (*DLS*) rule (Deo and Walker 1992) coupled with a secondary task that comprises the joints angles limits by means of the gradient projection method (Ho-

[5] The stand-alone application of Minimum-Jerk Cartesian controller used to obtain the presented experimental results is named *iKinArmCtrl* and relies on the *iKin* library. It is available from the repository: https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCub.

[6] A version of the code developed in (Hersch and Billard, Reaching with Multi-Referential Dynamical Systems 2008) that accepts target points expressed in the *iCub* standard reference frame can be downloaded from the repository:
https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/papers/pattacini2010.

Yul, Byung-Ju and Yungjin 2007). This solution employs the third-party package *Orocos*[7], a tool for robot control that implements the *DLS* approach (used by some partners of the RobotCub Consortium) and whose public availability and compliance with real-time constraints justified its adoption as one of the reference controllers.

In the first experiment we put to the test the three selected schemes in a point-to-point motion task wherein the *iCub* arm was actuated in 7-DOF mode and whose end-effector was controlled both in position and orientation. It came out that paths produced by our controller and the *DLS*-based system are well restricted in narrow tubes of confidence intervals and are quite repeatable; conversely the *VITE* is affected by a much higher variability.

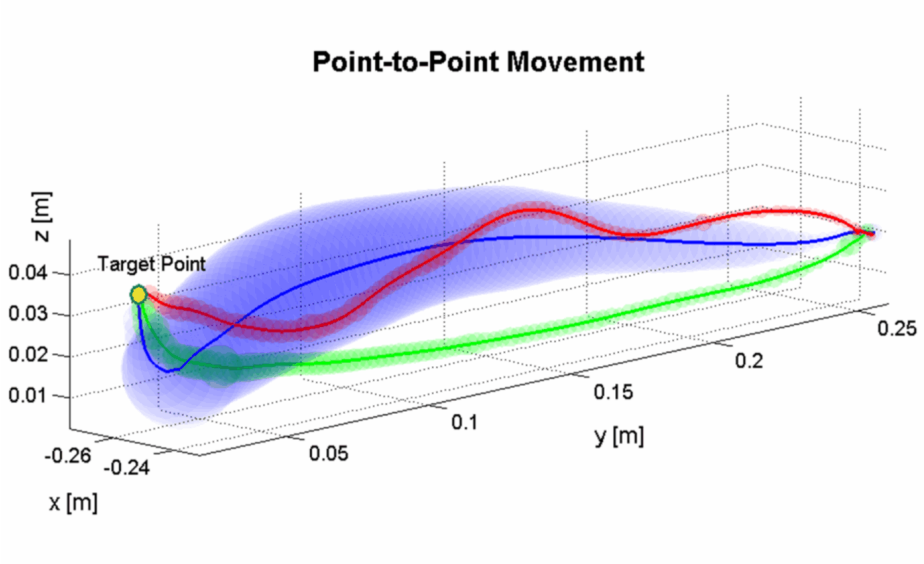

Figure 2-9. Point-to-point Cartesian trajectories executed by the three controllers: the *VITE*-based method produces on average the blue line, the minimum-jerk controller result is in green, the *DLS* system using *Orocos* in red. Bands containing all the measured paths within a confidential interval of 95% are drawn in corresponding colors. Controllers settings are T=2.0 s for the minimum-jerk system, α=0.008, β=0.002, K_P=3 for the *VITE*, and μ=10^{-5} for the damping factor of the *DLS* algorithm.

[7] The kinematic component of *Orocos* project is reachable here: http://www.orocos.org/kdl.

Figure 2-9 highlights what reported for a set of 10 trials of a typical reaching task where the right hand is moved from the rest posture towards a location in front of the *iCub* waist with the palm turned downward; Table 2-2summarizes the measured in-target errors for the three cases: all the controllers behave satisfactory, but the *DLS* achieves lower errors because operates continuously on the current distance from the target x_d, being virtually capable of canceling it at infinite time. On the contrary, strategies based on the interaction with an external solver bind the controller module to close the loop on an approximation $\tilde{x}_d$ of the real target that is determined by the optimization tolerances as in (2.1).

Regarding the analysis of human-likeness, the new proposed Cartesian controller outperforms both the traditional and the *VITE*-based solution thanks to the regulator design – so near to the ideal minimum-jerk model – and also as consequence of the wider working region where the task space module can function due to the replacement of the shaping policy. It is indeed clear from Figure 2-9 how the trajectory commanded by the minimum-jerk controller (green line) approaches much more a quasi-straight path whereas the red and the blue lines oscillate before reaching the target.

Controller	Position Error	Orientation Error	Mean Radius of Trajectory Band
VITE-based	$1.3 \pm 1.4 \cdot 10^{-3}\,\mathrm{mm}$	$0.041 \pm 0.05\mathrm{rad}$	$10 \pm 10.8\mathrm{mm}$
Min-Jerk-based	$3.0 \pm 1.4 \cdot 10^{-3}\,\mathrm{mm}$	$0.048 \pm 0.008\,\mathrm{rad}$	$2.5 \pm 1.5\,\mathrm{mm}$
DLS-based	$1.3 \pm 1.4 \cdot 10^{-3}\,\mathrm{mm}$	$0.016 \pm 0.028\,\mathrm{rad}$	$2.0 \pm 1.36\,\mathrm{mm}$

Table 2-2. Mean errors along with the confidence levels at 95% computed when the target is attained. An average measure of the variability of executed path is also given for the three controllers.

This important aspect becomes evident also once the velocity of the end-effector in the operational space is drawn as shown in Figure 2-10 : the velocity profiles have been

computed in post-processing from the indirect acquisition of the end-effector coordinates by reading the joints encoders' values (i.e. $x(t) = K(q(t))$ and then have been filtered to remove the high-frequency components (with a cutting frequency of 2.5 Hz). The superposition with the curve of the ideal minimum-jerk prototype (sketched black line), identified by knowing the starting and the ending points of the motion, underlines the good level of similarity of our implementation (green line) and, at the same time, the discrepancy of the other two methods which show a very sharp onset and a remarkable asymmetry of the response. Table 2-3 sums up the jerk measures computed in the Cartesian space: a factor of 43.8% is gained with respect to the *VITE* system and even a factor of 69% is achieved against the *DLS*.

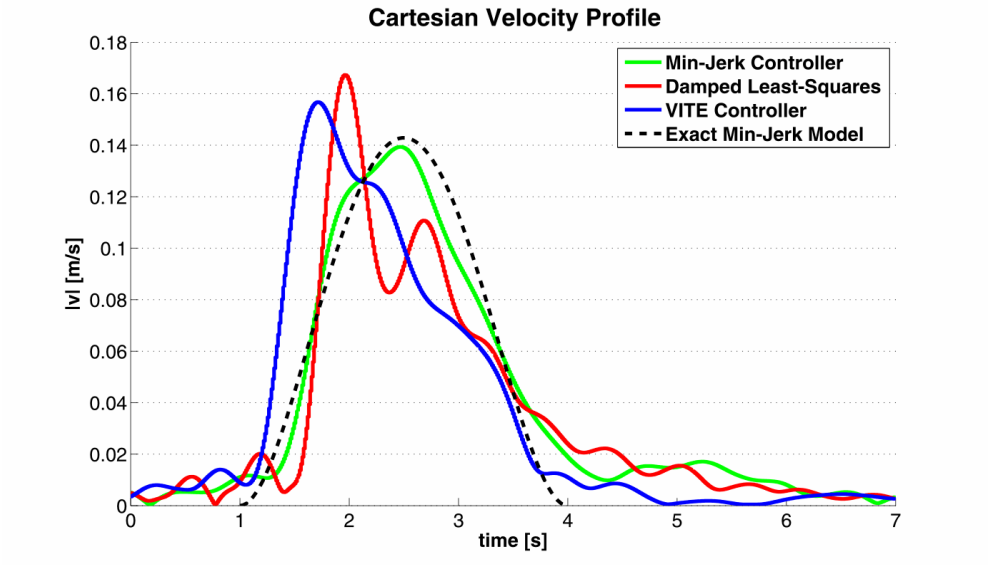

Figure 2-10. Magnitude of the end-effector velocity in the operational space: blue for *VITE*, red for *DLS* and green for minimum-jerk controller. The point-to-point task begins at *t*=1 s.

The second evaluation was concerned with the dynamical characteristics of the *DLS* and minimum-jerk controllers. In particular, we verified their capabilities to track in position a quite fast reference trajectory given in the operational space while keeping the orientation of the end-effector constant; unfortunately, we did not manage to test the

VITE algorithm as we experienced that the implemented *CCD* solver was not fast enough to run in real-time.

Controller	$\int_1^{+\infty} \left(\frac{d^2 v_{\text{cart}}}{dt^2} \right)^2 d\tau \Big/ \int_1^{+\infty} \left(\frac{d^2 v_{\text{exact model}}}{dt^2} \right)^2 d\tau$
VITE-based	71.86
Min-Jerk-based	40.33
DLS-based	131.06

Table 2-3. Relative measures of the jerk in the operational space given as the ratio between the integral of the squared second derivative of the Cartesian velocity for the three controllers and the same quantity computed for the exact minimum-jerk model.

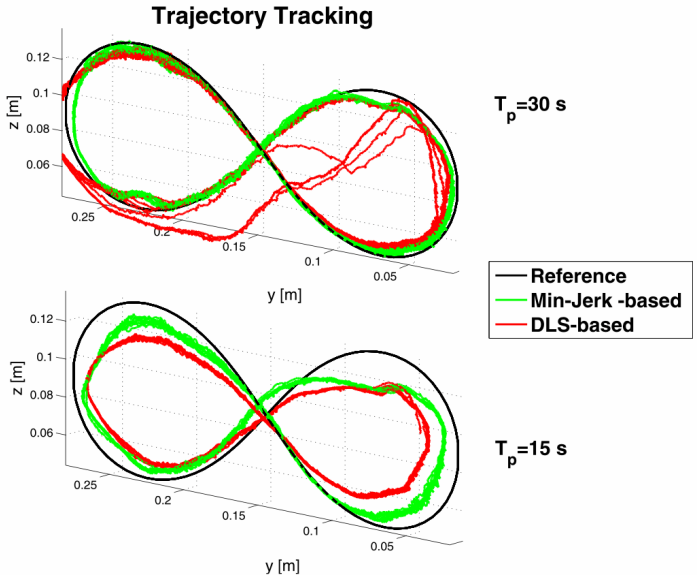

Figure 2-11. Controllers' responses while tracking a lemniscates shape: minimum-jerk controller in green, *DLS* in red. The resulting trajectories of 10 trials are shown for the two time periods T_p.

The target passed to the controllers evolved along a lemniscate-shaped trajectory (Figure 2-11), completing one cycle with two different time periods: T_P in {30, 15}

seconds. In the first experiment ($T_P = 30\,\mathrm{s}$), the minimum-jerk Controller ran with the parameter T set to 2.0 s and accomplished the task considerably well; the *DLS* method deviated somehow from the reference and did not perform with the same accuracy. When the target moved faster ($T_P = 15\,\mathrm{s}$), the minimum-jerk Controller still behaved better, in the sense that the gap between the executed curve and the reference was lower compared to the *DLS* case, as we reduced the parameter T to 1.5 s in order to get a quicker response; on the other side, the *DLS* reactivity remained unchanged lacking of an analogous built-in tuning.

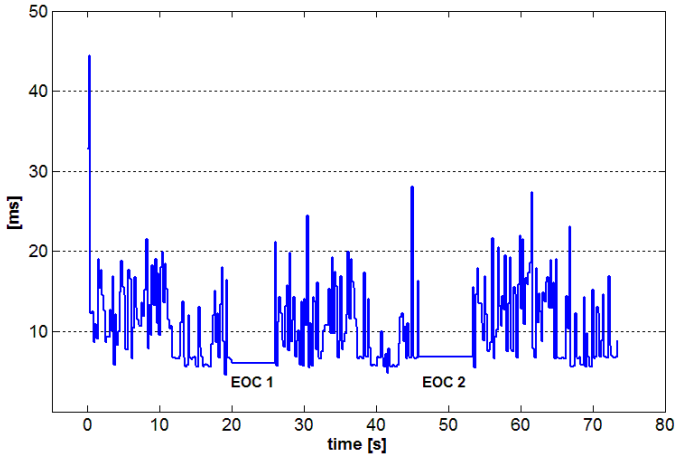

Figure 2-12. A plot of solver computation time during tracking of the lemniscate-shaped trajectory: the end of first and second cycle (respectively EOC1 and EOC2) is visible in the figure.

Notably, it is crucial to mention for this kind of test that *IpOpt* is able to comply with the stringent real-time constraints and eventually allows closing the loop of Figure 2-2. During tracking the average computation time for the case $T_P = 15\,\mathrm{s}$ is below 20 ms (on a multi-core Intel (R) Xeon with 2.27 GHz of clock frequency); this allows performing tracking with a constant solver rate of 33 Hz and a controller rate of 100 Hz. Interestingly, only a small latency is experienced at the beginning of the first cycle when the starting position of the end-effector is far from the target: this is expected since the

optimization algorithm starts from an initial "guess" that is far from the solution and takes a longer convergence time. However, the initial overhead is acceptable (< 50 ms), and does not cause a significant degradation of the real-time performance.

CHAPTER 3
THE GAZE CONTROLLER

3.1 Introduction

The Gaze Controller is a fundamental tool that lies at the basis of any attentive systems designed specifically for the field of humanoid robotics (Ruesch, et al. 2008), (Natale, et al. 2005). Its goal is to allow the robot to focus its gaze on salient objects that might be of some relevance for the task to be executed or just to prompt the robot curiosity towards the exploration of surrounding environment in order to foster learning through interaction and experience. In this context it comes out quite significantly that the coordinated motion of the eyes and the head must be reasonably fluent and fast to deal with rapidly variable cues, such as motion, colors, moving shapes and/or templates, in order to point at the objects whose features are considered central and eventually increase the knowledge of the world collected by the robot.

Lately in literature there has appeared an increasing number of significant works addressing the problem of controlling an anthropomorphic head: notably the proposed implementations are inspired from studies conducted on humans since the final aim is to mimic the coordinated movements of the eye-head system. Therefore (Maini, et al. 2008) designed the controller block diagram directly on the basis of the independent gaze control model originally introduced by (Goossens and Van Opstal 1997) as the outcome of empirical evidences gathered on humans to then confirm the initial hypotheses and validate their experimental achievements by carrying out a comparison with physiological data. By contrast, (Lopes, et al. 2009) applied an approach that resorted more to well-established theory such as optimal control and feedback controller design to realize a system exhibiting the peculiar features of a human gaze.

An interesting method that attempted to walk off from traditional groundings makes use of nonlinear dynamics and chaos theory (Duran, Kuniyoshi and Sandini 2008) with a formalism introduced in the middle of 80's by (Kaneko and Tsuda 2001) that employs the so called Coupled Map Lattices (*CML*), a set of nodes globally coupled in space and whose temporal evolution is guided by the well known logistic map which is responsible for the development of chaotic behaviors. By suitably linking the visual stimuli together with the input of this chaotic field and analogously the output of the lattice straight with the motor torque commands it has been demonstrated the emergence of a surprising self-organized capability of the system in controlling the axes of a simulated eye to perform smooth pursuit in tracking scenarios.

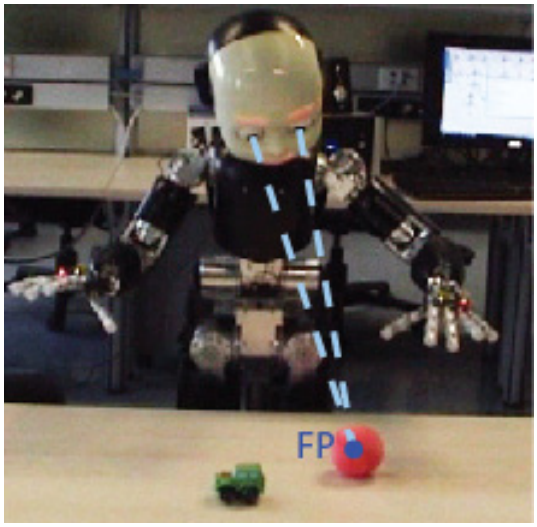

Figure 3-1. *iCub* looks at a ball in the 3D space, having his fixation point (FP) placed right at the center of the target object.

In the context of this thesis, having a somewhat robust framework for coping with control of the robot's limbs in the operational space draws naturally the interest to the possibility of applying the same concepts also to the control of robot's gaze. In fact, as better explained in the remainder of the chapter, one can think of the fixation point, i.e. the point in the Cartesian space where the robot is currently looking at, as a virtual end-

effector that can freely moves according to the values assigned to the head joints (3 for the neck and 3 for the eyes). Hence, this time, the inverse kinematic problem for the gaze consists in finding those values of the joints that allows the robot to fixate a given 3D location in the space (Figure 3-1), whereas the orientation of the head is less meaningful to be given, or even it can be left unspecified to the underneath controller in order to meet the requirement of providing a bio-plausible gaze motion for which basically the eyes move with quick saccades to the target and then counter-rotate to compensate for the slower neck movements.

Consequently, if for the Cartesian controller outlined in Chapter 2 the fundamental contribution was mainly due to the system's robustness and the quality of experimental results compared to other implementations, since the coupled solver-controller structure is not new in the area of reaching researches, here, differently, for the gaze control problem the advancement is represented appropriately by the architecture itself, that actually introduces an element of novelty, being a solution not already pursued in this domain.

Moreover, another important aspect that prominently motivated this work was the need to develop a gaze manager that behaved smoother with respect to the implementation on the *iCub* that was currently available (Lopes, et al. 2009), and further to also rationalize the commands interface exposed at the user level which turned to be rather complex and cumbersome. The last section of the chapter reports indeed the experiments executed on the platform in pretty the same comparative manner as those regarded the Cartesian controller, demonstrating how performance gains have been achieved.

3.2 Gaze Shifts in Humans

It is worth briefly recapping at this point some relevant behaviors that can be easily observed in humans while performing gazing tasks.

To foveate a visual target in the periphery, human gaze shifts are carried out with combined eye and head movements so as it is widely reported in literature: (Guitton and Volle 1987), (Phillips, et al. 1995), (Tweed, Glenn and Vilis 1995), (Freedman and

Sparks 2000), and finally (Goossens and Van Opstal 1997). If described in terms of functional movements gaze shifts typically consist of at least one rapid eye movement (saccade) followed by a slower head movement in the same direction. The gaze is then stabilized after the saccade by a vestibular driven slow phase that is present until the head comes to rest.

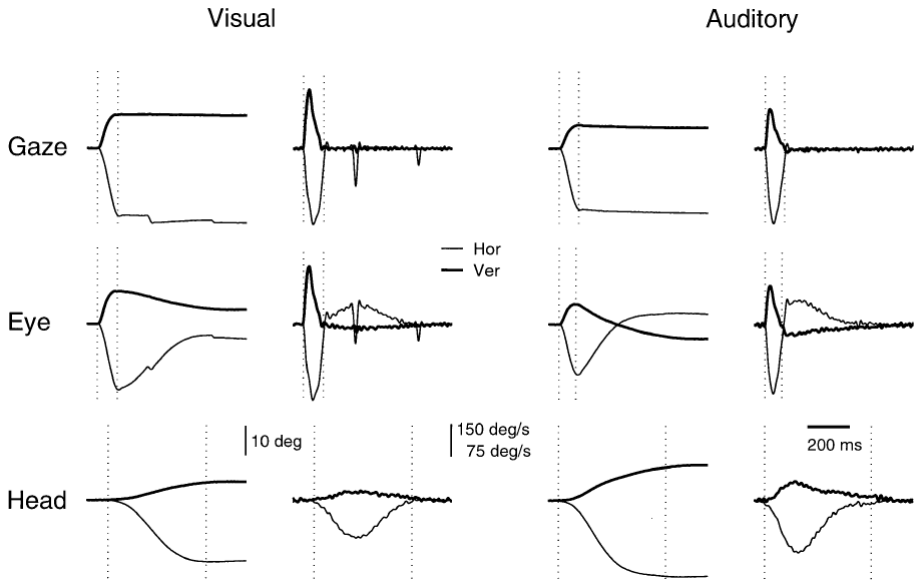

Figure 3-2. Human saccadic responses. Typical examples of a visually evoked (left-hand traces) and an auditory-evoked (right-hand traces) gaze shift towards a target. Both eyes and head were initially aligned with the straight-ahead fixation spot. The position (1st and 3rd columns) and velocity (2nd and 4th columns) traces are aligned with stimulus onset. Note the different scale for head velocities.

In Figure 3-2 it is shown the typical temporal evolution of eye and head displacements along with the resulting gaze as described in (Goossens and Van Opstal 1997): the two phases of the coordinated movement are plainly highlighted, as the achievement of the visual target is firstly obtained with the saccadic movement of the eye and then the gaze stable fixation is maintained for the counter-rotating movement of the eye.

3.3 The Cartesian Formulation of the Gaze Problem

To introduce the reader to the gaze inverse problem, here are resumed some useful quantities normally used to describe the gaze posture for the *iCub* platform whose formulation remains anyhow at a general level and thus can be easily applied to different robots.

Given the kinematic structure of the torso and head parts (see Figure 3-3) it is possible to extract the parametric description of the z-axes outgoing from the cameras planes (the z_8-axes depicted in blue in Figure 3-3), which are generally not co-planar.

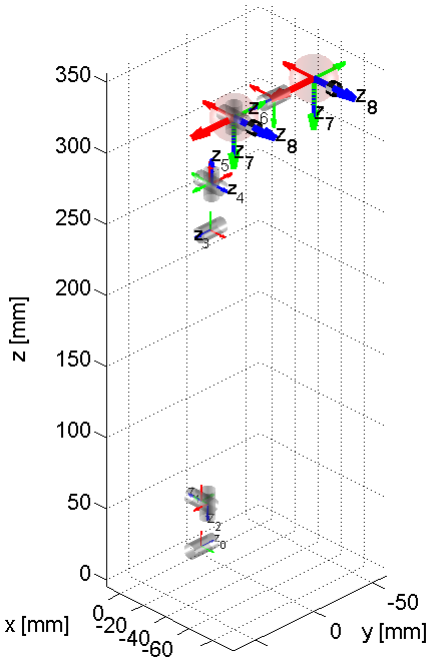

Figure 3-3. Kinematic description of the *iCub* head. Rotation axes attached to the joints are reported in blue.

Therefore, called o_l the center of the left camera plane and z_l the axis perpendicular to the left camera plane, and analogously o_r and z_r for the right camera, the lines $l(t_l)$ and $r(t_r)$ outgoing from the camera planes are in their respective parameters t_l and t_r the following:

$$\begin{aligned} l &: \; o_l + t_l \cdot z_l; \\ r &: \; o_r + t_t \cdot z_r. \end{aligned} \tag{3.1}$$

Being p_l the point of l given for the value t_l^* that minimizes the distance from the line r and, similarly, p_r the point over r given for the specific value t_r^* minimizing the distance from l, then the fixation point FP can be finally found as the mean point between p_l and p_r.

Thus, from the two separate minimizations it simply derives that:

$$\begin{aligned} t_l^* &= \frac{\left(z_l - \left(z_l \bullet z_r\right) z_r\right) \bullet \left(o_l - o_r\right)}{\left(z_l \bullet z_r\right)^2 - 1}; \\ t_r^* &= \frac{\left(\left(z_l \bullet z_r\right) z_l - z_r\right) \bullet \left(o_l - o_r\right)}{\left(z_l \bullet z_r\right)^2 - 1}, \end{aligned} \tag{3.2}$$

and in conclusion:

$$FP = \frac{p_l + p_r}{2} = \frac{o_l + t_l^* z_l + o_r + t_r^* z_r}{2} \tag{3.3}$$

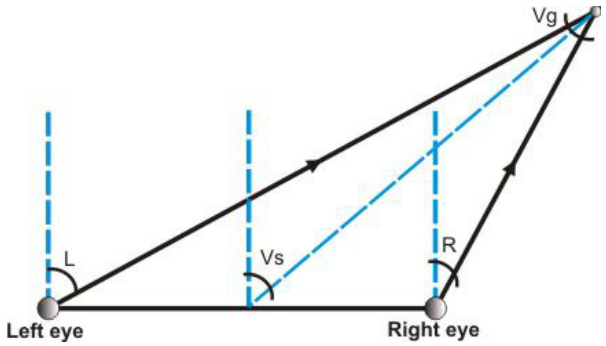

Figure 3-4. Example of a possible gazing configuration of the eyes as projected in the transverse plane: the eye-specific pan L and R are linked to form the couple given by the version V_s and vergence V_g angles. The version angle is computed starting from the middle point of the baseline connecting the two eyes.

Obviously, all the quantities $o_{l,r}, z_{l,r},$ and $t^*_{l,r}$ depend on the current robot joint configuration $\left[q_{\text{torso}}, q_{\text{neck}}, q_{\text{eyes}} \right]^T$. However, concerning the eyes kinematic, this description has to be provided not with regard to the usual Denavit-Hartenberg eyes joints variables $q_{\text{eyes}} \in \mathbb{R}^4$ – representing the tilt and the pan for the left ($\left[q_{\text{eyes}}(1), q_{\text{eyes}}(2) \right]^T$) and right cameras ($\left[q_{\text{eyes}}(3), q_{\text{eyes}}(4) \right]^T$) – but rather in the usual human-like gaze reference frame, since the eyes are actuated in a coupled manner so that their cumulative 4 degrees of freedom reduce to the triplet composed of the common tilt T, the version V_s and the vergence V_g angles, as in Figure 3-4.

The conversion formulas are given by:

$$\begin{cases} \mathrm{T} = q_{\text{eyes}}(1) = q_{\text{eyes}}(3) \\ V_s = \dfrac{q_{\text{eyes}}(2) + q_{\text{eyes}}(4)}{2}, \\ V_g = q_{\text{eyes}}(2) - q_{\text{eyes}}(4) \end{cases} \tag{3.4}$$

where the expression for version V_s – as computed by DSPs at firmware level – holds only for small angles for which the tangent function can be conveniently replaced by its linear approximation.

With these premises the gaze inversion problem can be stated as the procedure that allows finding the suitable vector $q = \left[q_{\text{neck}}, \mathrm{T}, V_s, V_g \right]^T$ that solves the following optimization:

$$q^* = \arg\min_{q \in \mathbb{R}^6} \left\| FP_d - K_{FP}(q) \right\|^2 \tag{3.5}$$

being $FP_d \in \mathbb{R}^3$ a desired fixation point to be attained in the 3D Cartesian space and K_{FP} the forward gaze map that returns the current robot fixation point known the head configuration $q_{\text{neck}} \in \mathbb{R}^3$ comprising the tilt, the pan and the yaw of the neck, and in addition the eyes configuration $\left[\mathrm{T}, V_s, V_g \right]^T$. It is worth recalling here that the

dimensionality of the input space of (3.5) is 6, whereas the dimensionality of the output space (i.e. the task space) is 3; hence the problem turns out to be intrinsically redundant.

Once the (3.5) is solved, a controller is in charge of commanding the motors in velocity mode to steer the system from its initial position to the final joints position ensuring smooth movements that aim at reducing the jerk, resorting basically to the same method discussed in Chapter 2.

For a further simplification, the task (3.5) can be profitably divided in two sub-problems, mirroring somehow the underlying structure that encompasses two sub-systems moving independently, i.e. the head and the eyes. This approach leads to the design choice of having two distinct solvers as explained hereafter.

3.4 The Neck Solver

To redirect the neck in order to expose the forehead as much as possible to the target fixation point it is useful to consider the line that outgoes from the virtual point o_c placed at middle of the baseline connecting the two eyes whose Cartesian position is given by

$$o_c = \frac{o_l + o_r}{2}, \qquad (3.6)$$

and whose direction z_c is parallel to the z_8 -axes in Figure 3-3 when the eyes tilt, version, and vergence angles are all zero, and remains fixed in the frame attached to the head.

Consequently, in the context of head-based control it suffices to solve the following task:

$$q^*_{\text{neck}} = \arg \min_{q_{\text{neck}} \in \mathbb{R}^3} \left(\frac{z_c \bullet \left(o_c - FP_d \right)}{\left\| o_c - FP_d \right\|} \right). \qquad (3.7)$$

It is quite straightforward to recognize that the argument of (3.7) corresponds exactly to

the cosine of the angle θ between the unitary axis z_c and the vector $d = o_c - FP_d$ and assumes the minimum -1 when z_c and d are anti-parallel.

The input space dimensionality of (3.5) boils down to 3 as specified in (3.7), nonetheless the dimensionality of the modified task space changes to 2[8], since the neck tilt and pan are sufficient to exhaustively tackle the (3.7), so that the new problem is still affected by a certain amount of redundancy. To better cope with the exceeding degree of freedom represented by the neck yaw, the (3.7) can be transformed in the following nonlinear optimization and treated relying on the *IpOpt* package:

$$q^*_{\text{neck}} = \arg \min_{q_{\text{neck}} \in \mathbb{R}^3} \left\| q_{\text{neck}_d} - q_{\text{neck}} \right\|^2$$
$$\text{s.t.} \begin{cases} \cos\theta < -1 + \varepsilon \\ q_{\text{neck}_L} < q_{\text{neck}} < q_{\text{neck}_U} \end{cases}, \tag{3.8}$$

with ε a suitable small number in the range $\left[10^{-5}, 10^{-4}\right]$ as in the problem (2.1).

The (3.8) codes the objective of (3.7) as a nonlinear constraint and introduces a new function to minimize that takes into account the distance of the solved configuration from a user-specified set of "resting" neck joints values q_{neck_d}. As already discussed in Chapter 2, a constraint owns clearly higher priority with respect to the objective function and thus plays the role of a primary task: this ensures that the objective function is minimized only after all the given constraints are satisfied (if possible).

Interestingly, the resting configuration can be determined at run-time before starting the *IpOpt* computation according to the current output of the inertial sensor mounted aboard the *iCub* as in Figure 3-5. Through the inertial sensor readouts, indeed, the direction of the gravity can be expressed in the head reference frame and, in case it varies as result for

[8] The problem (3.7) concerns indeed the goal of aligning two vectors in the 3D space, hence the task space of (3.7) is bi-dimensional.

example of the displacements induced by the torso movements, the desired resting configuration can be effectively modified and made dependent on the sensor's output in order to maintain as much as possible the head posture stable with respect to the gravity.

This head stabilization mechanism reveals to be a fairly helpful feature in many practical situations when the robot is required to execute grasping and crawling tasks, or more generally tasks that entail an exploration of the surroundings, since the whole body motion is prone to interfere quite significantly with the appearance of the objects acquired by the cameras that in turn are processed by the vision algorithms.

The final neck joints configuration as computed by solving the problem (3.8) represents the target attractor for the second-stage joints controller described later in section 3.6.

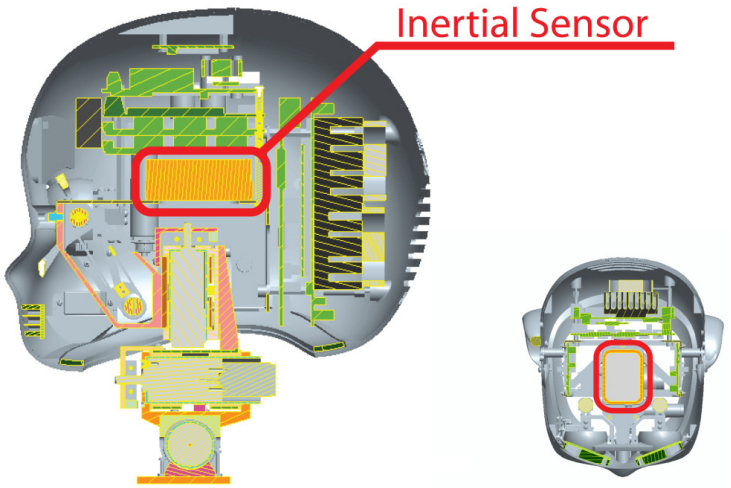

Figure 3-5. Location of the inertial sensor within the *iCub* head.

3.5 The Eyes Solver

It is possible to specialize the equation (3.5) for the eyes solver as follows:

$$q^{*}_{\text{eyes}} = \arg\min_{q_{eyes}} \left\| FP_d - K'_{FP}\left(q_{\text{eyes}}\right) \right\|^2, \qquad (3.9)$$

where K'_{FP} is the forward gaze map computed over the redefined eyes part joints triplet $q_{\text{eyes}} = \left[\text{T}, V_s, V_g\right]^T$ as independent variables.

Because the output space is of dimensionality 3 so as the dimensionality of the input space, the sub-task (3.9) is not redundant. It thus stems that it is convenient to solve the (3.9) by simply resorting to the pseudo-inverse algorithm. In particular, the final joints configuration is determined as the vector where the following iterative time-discrete equation converges:

$$q_{\text{eyes}_{t+1}} = q_{\text{eyes}_t} + \Delta T\left(G \cdot J^{\#} \cdot \left(FP_d - K'_{FP}\left(q_{\text{eyes}_t}\right)\right)\right). \qquad (3.10)$$

with ΔT the integration sample time and G a suitable gain; $J^{\#}$ is the pseudo-inverse of the Jacobian of the gaze forward map $J = \partial K'_{FP}\left(q_{\text{eyes}}\right) \big/ \partial q_{\text{eyes}}$.

The convergence rate of (3.10) is very quick thanks to the low dimensionality of the search space and provides the reference trajectory as it evolves over time to be tracked by the second-stage controller.

Employing the Vestibular-Ocular Reflex (*VOR*)

Notably, retaining the framework of equation (3.10), it is already possible to bring in a term $\dot{q}_c$ to account for the counter-rotation required to compensate the head movement; the complete formulation is thus the following:

$$q_{\text{eyes}_{t+1}} = q_{\text{eyes}_t} + \Delta T\left(G \cdot J^{\#} \cdot \left(FP_d - K'_{FP}\left(q_{\text{eyes}_t}\right)\right) - \dot{q}_c\right). \qquad (3.11)$$

The component $\dot{q}_c$ is estimated considering how through the gyroscope readouts $\left[\omega_x, \omega_y, \omega_z\right]^T$ of the inertial sensors – which moves jointly with the head – a measure of

the velocity of the actual fixation point v_c can be inherited. In this respect it is possible to demonstrate that it holds:

$$v_c = \omega_x x_I \times \left(FP_d - c_I \right) + \omega_y y_I \times \left(FP_d - c_I \right) + \omega_z z_I \times \left(FP_d - c_I \right), \qquad (3.12)$$

being $\left(x_I, y_I, z_I \right)$ the frame attached to the inertial sensors in the location c_I with respect to the root frame. The term $\dot{q}_c$ is finally given by $\dot{q}_c = J^{\#} v_c$ and it represents a sort of Cartesian implementation of the so called vestibular-ocular reflex (*VOR*) present in the human eye-head coordination as for example described in (Rosander and von Hofsten 2000).

3.6 The Neck-Eyes Controller

As anticipated previously in this chapter, the second-stage controller is in charge of delivering the proper velocity profiles to command the robot's motor, given the desired attractor point (found by the optimizer) to be achieved in the neck configuration space as well as the reference trajectory (computed by the pseudo-inverse) to be tracked in the eyes joints space. Concerning the latter, the reason why the reference joint velocity yielded by the pseudo-inverse is integrated and passed to the controller instead of directly fed back into the robot system is that the pseudo-inverse produces exponential trajectories that are inherently jerky and evolve according to the values of parameter G in (3.10) and (3.11) that does not have a direct link to the movement time.

To reduce the jerkiness and obtain a final easy-to-use tuning of the fundamental parameters the controller's baseline analyzed in Chapter 2 is here adopted again and its structure is replicated twice, one to deal with the control of the neck, one to control the eyes as in the following:

$$a)\ \frac{\dot{q}_{\text{neck}}}{q_{\text{neck}_d} - q_{\text{neck}}} = \frac{-a/T_{\text{neck}}}{\text{s}^2 - \left(c/T_{\text{neck}}^3\right)\text{s} - b/T_{\text{neck}}^2}$$

$$b)\ \frac{\dot{q}_{\text{eyes}}}{q_{\text{eyes}_d} - q_{\text{eyes}}} = \frac{-a/T_{\text{eyes}}}{\text{s}^2 - \left(c/T_{\text{eyes}}^3\right)\text{s} - b/T_{\text{eyes}}^2} \tag{3.13}$$

The controllers' equations are provided in terms of Laplace transfer functions that return as output the velocity profiles taking as input the actual error in the configuration spaces; the values of coefficients $a, b,$ and c are identical to the values computed for the Cartesian controller in Chapter 2, whereas two different parameters, T_{neck} and T_{eyes}, serve to specify the point-to-point movements time: usually $T_{\text{eyes}} < T_{\text{neck}}$.

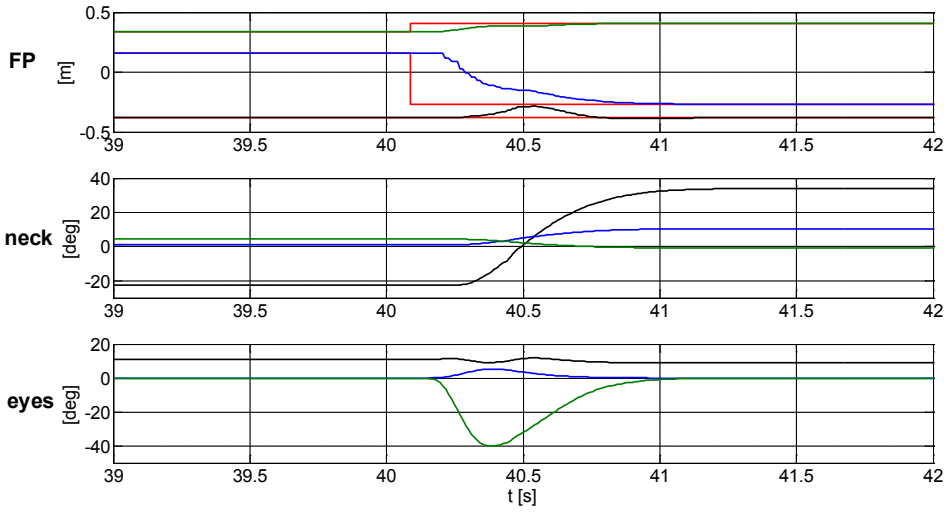

Figure 3-6. Profiles obtained with the gaze controlled to perform a quick saccade starting approximately at t=40.08 s. *Top graph (FP)* – The target Cartesian position is depicted in red for the three components; the current fixation point coordinates x (black), y (blue), z (green) are also shown, being expressed in the root reference frame. *Middle graph (neck)* – The neck joints values are traced to reach the desired configuration as solved by the optimizer: the pan is in black, the tilt in blue, the yaw in green. *Bottom graph (eyes)* – The movements of the eyes are shown: pan in green, tilt in blue and vergence in black.

Differently from the implementation studied in Chapter 2, it does not exist here any explicit resort to a controller devoted to shape the trajectory of the fixation point in the Cartesian space, so that the Multi-Referential approach is not required.

Finally, to handle the counter-rotation of the eyes improving the coordination with the head motion, it is necessary to subtract the feed-forward term $\dot{q}_c$ (due to the *VOR*) to the output $\dot{q}_{\text{eyes}}\left(t\right)$ of the controller (3.13)-b.

Figure 3-6 shows a simulation of how the solvers and controllers succeed in handling a request for a quick saccade: in the first phase (approximately for $40.08\ \text{s} < t < 40.42\,\text{s}$) the eyes move rapidly and the robot's fixation point almost attains the target; at the same time the neck starts the motion at a lower velocity compared to the eyes' velocity. In the last phase ($t > 40.42\,\text{s}$) the neck concludes its motion while the eyes continue to counter-rotate to keep the fixation point on the target.

The Oculo-Collic Reflex (*OCR*) as Recovery for the *VOR*

At this point it has to be mentioned that it is fairly straightforward to device a sort of recovery strategy for the eyes counter-rotation in case of a failure or even the absence of the inertial sensor. In fact, the velocity commands generated by the neck controller can be considered as a further feed-forward signals available at the eyes solver's stage to be employed as a suitable kinematic equivalent of the inertial sensor's readouts. In other words, knowing the neck velocity $\dot{q}_{\text{neck}} = \left[\omega_{\text{tilt}}, \omega_{\text{pan}}, \omega_{\text{yaw}}\right]^T$ as the output of the neck controller and the orientation of the corresponding rotation axes $z_{\text{tilt}}, z_{\text{pan}}$ and z_{yaw} along with the centers of the relative reference frames $o_{\text{tilt}}, o_{\text{pan}}$ and o_{yaw} from the kinematic description of the robot, the formula (3.12) for v_c can be replaced by the following:

$$\begin{aligned} v_c = {} & \omega_{\text{tilt}} z_{\text{tilt}} \times \left(FP_d - o_{\text{tilt}}\right) + \omega_{\text{pan}} z_{\text{pan}} \times \left(FP_d - o_{\text{pan}}\right) + \\ & \omega_{\text{yaw}} z_{\text{yaw}} \times \left(FP_d - o_{\text{yaw}}\right), \end{aligned} \tag{3.14}$$

which analogously embodies the Cartesian implementation of the so called oculo-collic

reflex (*OCR*) present in the human eye-head coordinated behavior.

3.7 Additional Coordinate Systems

It has been shown that the designed Gaze Controller is inherently Cartesian since it drives the system with the aim to attain a specified fixation point in the 3D space. Nevertheless in many cases it reveals to be worthwhile having the possibility to express targets in different coordinate systems, such as the angular system and the image pixel coordinates. To achieve that a simple coordinate's transformation is required, that is capable of extracting from the input target the three Cartesian components to be passed to the underlying controller. Therefore, while retaining the Cartesian structure of the solver-controller couple, further additional coordinate systems detailed hereafter are also implemented as a higher level layer.

Angular Coordinate System

The target has to be given as a combination of angular quantities and specifically: the azimuth α, the elevation φ and the vergence angle V_g. The angular reference frame is head-centered, i.e. it is attached at the middle point between the two cameras (o_c as defined in section 3.4), owning the same orientations of the cameras axes and it can be either *absolute*, when it refers to a fixed position of the torso and head, or *relative* to the current torso and head configuration.

There exist simple transformations from the angular domain to the Cartesian space; for example, considering the absolute head-centered case, the final 3D fixation point FP_d can be retrieved as follows:

$$FP_d = T_r^h R_\alpha R_\varphi T_h^r K_{\text{gaze}}\left(\left[0,\ldots,0,V_g\right]^T\right), \qquad (3.15)$$

where K_{gaze} is the gaze forward map computed in the fixed torso-head position with only

the input vergence applied, T_h^r is the homogeneous matrix that accounts for the position of the output of the K_{gaze} map in the head fixed reference frame, $T_r^h = \left(T_h^r\right)^{-1}$, and finally R_α and R_φ are the matrices that account for the rotation of the given azimuth and elevation angles, respectively, in the head fixed frame.

Image Pixel Coordinate System: the Monocular Case

By knowing the camera intrinsic parameters and relying on the pinhole model of an ideal camera, it is possible to back-project the 2D coordinates of an image pixel provided within one image plane into the corresponding point P in the 3D space with respect to a given reference frame (Ma, et al. 2004).

In fact the 3D point coordinates expressed in the homogeneous form are given by the following operation:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_r = \left(\underbrace{\begin{bmatrix} fs_x & fs_\vartheta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} T_e^r}_{\breve{\Pi}} \right)^{-1} \left(\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \right) \qquad (3.16)$$

where $\left(u, v\right)$ is the pixel coordinates of the point P, λ is the depth of P in the eye's reference frame, T_e^r is the homogeneous transformation from the root frame to the eye's frame, $\left(o_x, o_y\right)$ are the coordinates of the principal point in pixels, $\left(fs_x, fs_y\right)$ is the size of unit length in horizontal and vertical pixels and finally fs_ϑ is the skew of the pixel often close to zero. The overall result of the transformation can be represented by a simple linear application Π^{-1}.

Generally all the parameters are known except for λ, which in turn requires an *a-priori* knowledge of the distance of the physical point from the camera image plane. Consequently, in a framework where the robot is equipped with two cameras and the control relies on a monocular approach, such as the one described above, commands in

this modality will move the gaze so that when in fixation the point P will end up being at the center of the drive camera but still not aligned with the center of the second camera.

Image Pixel Coordinate System: the Stereo Case

Usually in a stereo strategy vision algorithms are able to achieve an inference of the distance of an acquired object by comparing the disparity produced on the two images planes by the object's features. Provided that the two identified salient points $\left(u_l, v_l\right)$ and $\left(u_r, v_r\right)$ within the left and right image plane, respectively, correspond to the same physical point P, it is then possible to overcome the problem of the monocular method by putting in place a continuous closed loop process based on visual cues that converge to the configuration where the pixel coordinates of P will coincide with the center of the image plane both for the left and the right camera.

This convergent mechanism is indeed implemented at each iteration i by the following three sub-steps executed in a row, once for example the left camera is selected to be the drive one:

$$
\begin{aligned}
& i_1) \;\; FP^i = \Pi^{-1} \begin{pmatrix} \lambda^i \cdot u_l^i \\ \lambda^i \cdot v_l^i \\ \lambda^i \end{pmatrix}; \\
& i_2) \;\; \mathrm{lookAtFixationPoint}\left(FP^i\right); \\
& i_3) \;\; \lambda^{i+1} = \lambda^i + \left(K_P + K_I \Delta T\right)\left(u_r^i - o_x\right).
\end{aligned}
\tag{3.17}
$$

It is evident as in the first step i_1 of (3.17) the prediction of the 3D point FP^i at the iteration i is computed based on the monocular formula and the current estimation of the unknown distance λ^i of FP^i from the drive camera (left, in this case). The point FP^i is

then used to command a gaze motion through the Cartesian controller (step i_2): the function *lookAtFixationPoint()* gives indeed the target FP^i to both the eyes and neck solver-controller couples working purely in the Cartesian space in order to generate a new or correct an ongoing gaze motion. Finally the updating third phase i_3 comes into play employing a discrete proportional-integral (PI) controller – with sample time ΔT and gains K_P and K_I – to adjust the estimation of λ for the next iteration $i+1$ according to the current displacement error $e_r^i = u_r^i - o_x$ measured with respect to the center of the image in the non-drive camera plane (right, in this case).

Remarkably, proceeding from iteration i to iteration $i+1$, the visual signals used in the closed loop, i.e. $\left(u_l, v_l, u_r, v_r\right)^{i \to i+1}$, vary as result of the independent motion of the object responsible for originating the cues themselves, as well as the ego-motion of the robot caused by the command in (3.17)-i_2, and thus they have to be provided by reliable vision algorithms not reported in the description of the process.

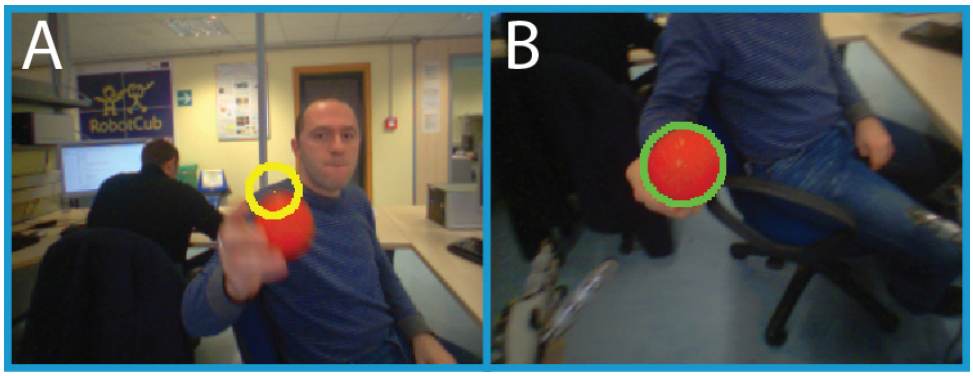

Figure 3-7. The output of the vision algorithm: *A*) the filter is searching for the best occurrence of a red circled object in the scene without actually having found it yet; *B*) the filter finds the object (highlighted by a green circle) and starts to track it.

As far as the vision cues are consistently guaranteed, the process (3.17) is conveniently repeated until the error in pixels in both the camera image planes goes under a given threshold relying on the robust convergence ensured by the PI controller. In this respect, indeed, a positive error $e_r^i > 0$ (i.e. $u_r^i > o_x$) corresponds to the situation where the

object of interest is located farther to the cameras than the current fixation point FP^i, correctly eliciting an increment on the estimation of λ^{i+1} according to (3.17)-i_3; conversely, a negative error $e_r^i < 0$ yields a decrement of λ^{i+1} which turns out to be in the correct direction since for $u_r^i < o_x$ the object of interest lies closer to the cameras with respect to the current fixation point. On the other hand, when the right camera is chosen to be the drive one, in order to keep the values of K_P and K_I positive and more importantly to still enforce the correspondence between the distance of the object of interest from the cameras and the sign of the error, the displacement error on the left image plane can be appropriately defined as $e_l^i = o_x - u_l^i$.

To sum up, the main difference of the stereo approach compared to the monocular method is that whereas the latter can operate just with one single command (executing actually a saccade), the former requires a continuous stream of information to run in closed loop: consequently, in stereo mode a quick initial saccade is followed by a smooth pursuit behavior managed by the PI controller, ensuring the convergence and guiding the gaze towards the target.

3.8 Experimental Results

In this section experimental results of the presented Gaze Controller[9] (hereinafter referred as GC1) conducted on the *iCub* are reported with particular reference to a comparison with the Gaze Controller[10] developed by (Lopes, et al. 2009) (hereinafter referred as GC2) which exploits a full-state feedback structure with a linear gain matrix in order to realize a human-like motion with neck-eye coordination.

[9] The software module implementing the Gaze Controller designed during this thesis work is available on line from within the *iCub* main repository:
https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCub/main/src/modules/iKinGazeCtrl.

[10] The software code from (Lopes, et al. 2009) is available from within the *iCub* contribution repository:
https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCub/contrib/src/controlGaze.

Case Study I: Tracking a Moving 3D Target

The first case study gives an account of how well GC1 performs tracking of a moving Cartesian target on the real robot.

Besides GC1, the software setup comprises also a vision module (Taiana, et al. 2010) capable of detecting objects of specific color and shape features (e.g. red ball as in Figure 3-7) relying on a particle filter that in turn accurately and quickly predicts the center position within the image plane and estimates the 3D location of the object from geometric considerations (knowing the radius of the ball and how it is projected on the image plane allows to retrieve the distance of the ball from the camera).

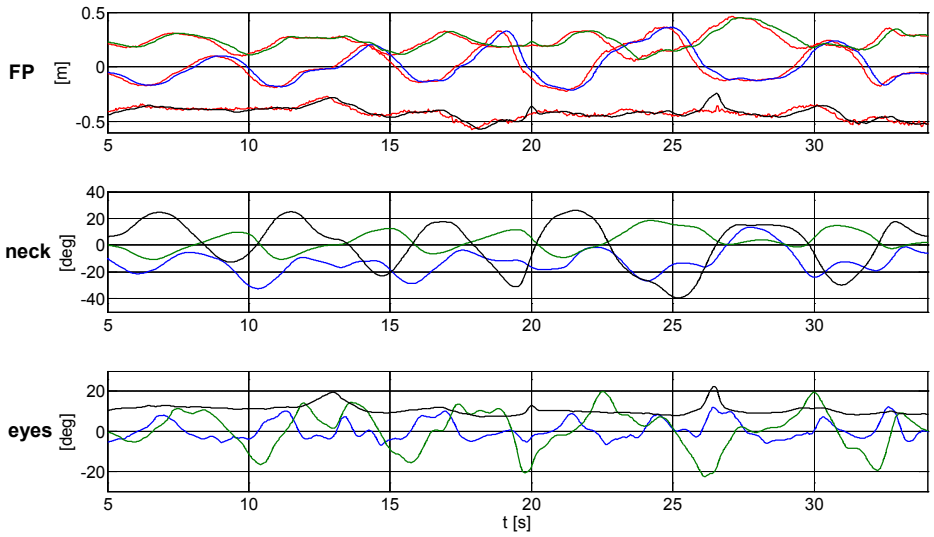

Figure 3-8. Profiles generated by GC1 while tracking the red ball in the 3D space. *Top graph (FP)* – Plot of the 3D Cartesian positions: target (red), x (black), y (blue), z (green) with respect the waist-based reference frame. *Middle graph (neck)* – Neck joints trajectories: tilt in blue, pan in black, yaw in green. *Bottom graph (eyes)* – Eyes joints trajectories: tilt in blue, pan in green, vergence in black.

The predicted 3D output Cartesian position of the particle filter while the object of interest is being moved by an operator is directly fed to GC1, producing the results

reported in Figure 3-8.

By analyzing the logged data it derives that GC1 is capable of perfectly tracking the Cartesian point as determined by the visual filter with an average error of less than $1\,\mathrm{cm}$ (it has to be consider that the reference trajectory is affected by noise, especially on the x component which mainly accounts for the distance from the eyes) and with a delay of approximately $200\,\mathrm{ms}$.

Remarkably, this case study does not apply to GC2 due to the fact that GC2 does not handle Cartesian inputs.

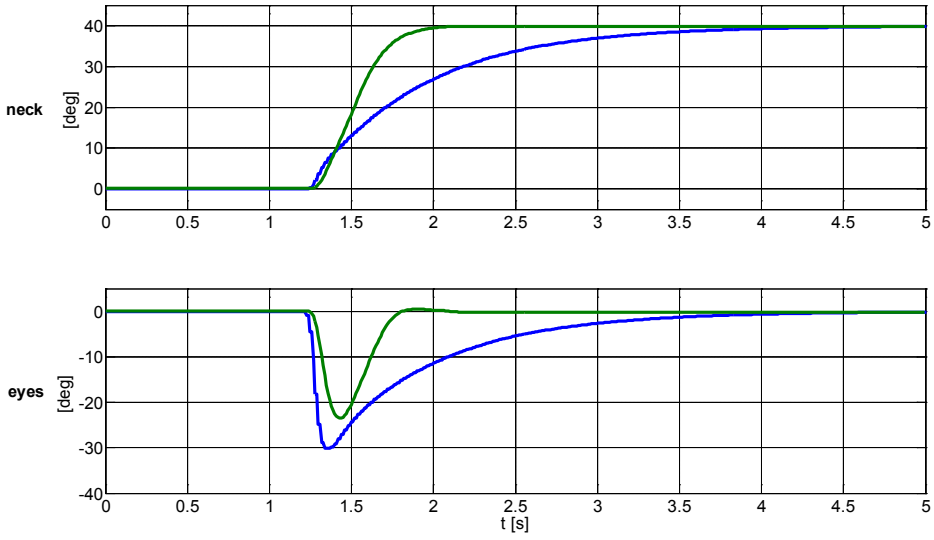

Figure 3-9. Neck pan (top) and eyes pan (bottom) for the saccadic test. Trajectories yielded by GC1 are shown in green, whilst profiles generated by GC2 are in blue.

Case Study II: Comparison GC1 vs. GC2 on Saccadic Movements

The first comparative analysis between GC1 and GC2 regards the behavior of the two controllers dealing with simple saccadic movements commanded in angular reference frame, where an azimuth of 40 degrees is requested to the robot starting from the conventional resting position (i.e. all head joints zeroed). The controllers' parameters are tuned in order to achieve their best working conditions.

Clearly from the plot in Figure 3-9 GC1 performs better than GC2 both in terms of velocity to reach for the target and in terms of smoothness of the trajectory, since the profiles generated by GC1 approximate minimum-jerk trajectories whereas the curves produced by GC2 resemble an exponential behavior.

Interestingly, the eyes movement yielded by GC2 (the peak is reached after 140 *ms*) is slightly faster than the corresponding eyes movement of GC1 (the peak is reached after 180 *ms*): this can be explained by the fact that GC2 commands the saccade in position at high level software control, thus not requiring a continuous reading of the feedback, whereas GC1 is a pure velocity controller that needs the position feedback to attain the target. On the other hand, the difference in timing results does not turn to be significant in terms of human-likeness given that the rapid human saccades last at maximum 200 *ms*.

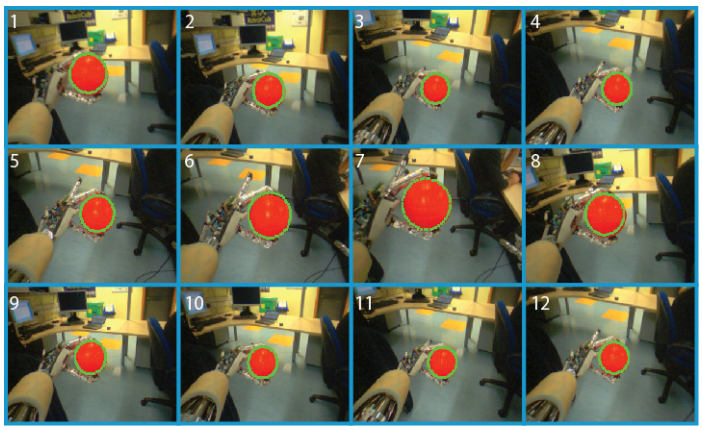

Figure 3-10. From top-left to bottom-right: an excerpt of a complete images sequence logged during the experiment of case study III. *iCub* gazes at the target ball held in its left hand while the left arm is moving following a predefined trajectory.

Case Study III: Comparison GC1 vs. GC2 on Tracking a Moving Object in the Image Planes

The experimental setup is here marginally modified with respect to the previous case where an operator moves randomly the target in front of the robot, since now it is required to have a somewhat repeatable input target profiles in order to perform

consistent comparisons between the controllers.

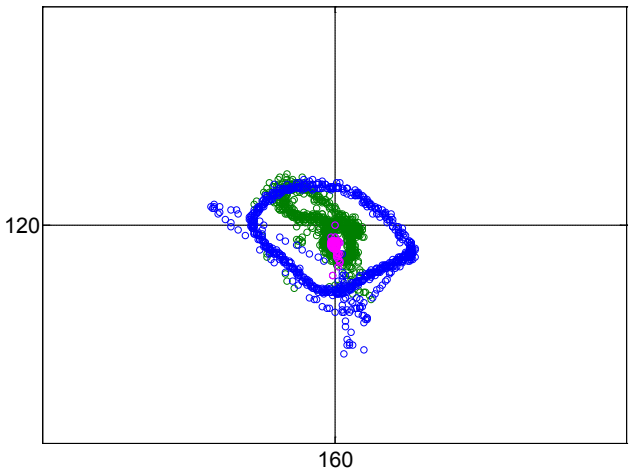

Figure 3-11. The trace of the red ball centroid as acquired within the drive image (with resolution 320x240) throughout the motion: in green is reported the trace of GC1, in blue the trace of GC2; finally in magenta are shown the trailing points of GC2 trace approaching the center at the end of trajectory.

The adopted solution is to have *iCub* holding the red ball in its hand (Figure 3-10) and then to rely on the arm Cartesian controller capable of following a suitable trajectory in the operational space, to finally let the gaze controllers track the projection of the ball in the cameras. The 3D trajectory appositely identified to be tracked by the left arm controller (with a point-to-point movement time of 1.25 *s*) is the following:

$$\begin{cases} x_r = -0.3 + 0.03\cos\left(\dfrac{2\pi t}{T}\right) \\ y_r = -0.15 + 0.07\cos\left(\dfrac{2\pi t}{T}\right), \\ z_r = 0.15 + 0.05\sin\left(\dfrac{2\pi t}{T}\right) \end{cases} \tag{3.18}$$

where $P \equiv \left(x_r, y_r, z_r\right)$ is the point moving over time t on the specified path expressed with respect the standard root reference frame of the robot, with length units given in meters, and T=3.33 s the trajectory period. The orientation of the hand is kept constant by the arm controller in such a way that the target ball stays visible during the entire trajectory. Moreover, in this phase the arm controller is allowed to command only the arm and not the torso to not hinder the gaze movements.

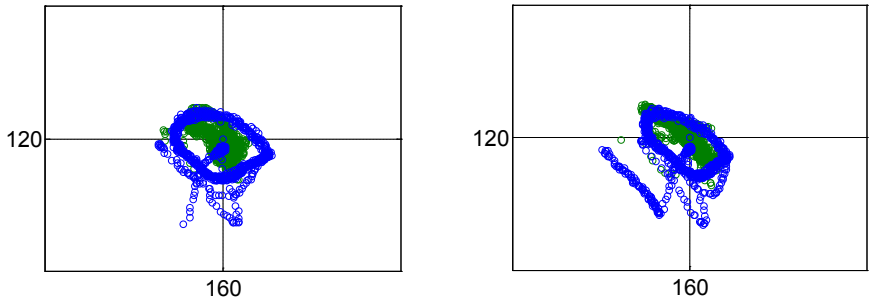

Figure 3-12. Left (left) and right (right) traces for the stereo tracking in the case study III: GC1 results are shown in green, whereas GC2 results are reported in blue.

The plot of Figure 3-11 represents the image plane of the drive camera with a resolution of 320x240 pixels in the case of tracking with monocular approach. The graph shows the centroid of the red ball given in pixels as it is acquired by the particle filter in the case of GC1 (green) and GC2 (blue): obviously the less scattered is the plot around the center of the image, the more effective is the tracking. It results that the average values with the corresponding confidence intervals at 95% for the distribution produced by GC1 are (157.7 ± 25, 117 ± 23.5) pixels, whereas GC2 distribution verifies (159.7 ± 54.1, 107.7 ± 39.24) pixels; that is, the GC2 results are almost twice spread compared to the GC1 case.

Particularly, unlike GC1, the GC2 performance are not sufficient to keep the centroid of the object in fovea for a trajectory period T=3.33 s, since, as it appears from Figure 3-11, the resulting blue profile in the image plane encompasses the center without actually achieving it, except at the end of the experiment when the target ball comes to the rest state being easily approached by the GC2 trace (as shown by the trailing points colored in magenta).

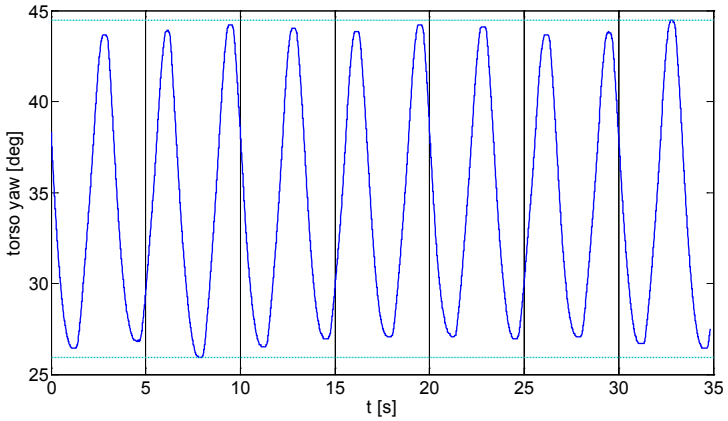

Figure 3-13. The torso yaw profile as logged during experiment for the case study IV: it is seen as an external disturbance by GC1.

Analogously, the experiment can be run for the stereo case, where two instances of the particle filters operate on the two different images; their outputs are thus sent to the controllers obtaining the results depicted in Figure 3-12, which tend to be quite similar to the behaviors recorded for the monocular test, highlighting once more the higher performance achieved by GC1against GC2.

Case Study IV: Robustness against External Perturbations to the Gaze Motion

In order to verify whether GC1 is robust against perturbations induced on the gaze motion, a simple variation to the experiment described in case study III has been put in place that foresees the use of the torso yaw joint by the arm controller. The trajectory to

be tracked by the arm end-effector is the same as (3.18), but this time the torso movement is enabled and produces unwanted rotations on the eye-head structure that need to be compensated.

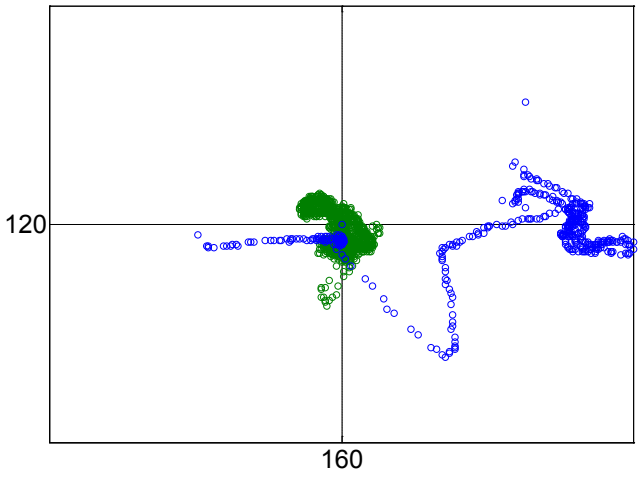

Figure 3-14. The trace of GC1 (green) in monocular visual tracking task when the torso yaw is also actuated. Trace in blue highlights how GC2 is not able to cope with external perturbation in the gaze control problem.

Figure 3-13 depicts the perturbation signal caused by the torso movement on the gaze as it evolves during the arm tracking task (for a span of the torso yaw of approximately 20 degrees), whereas Figure 3-14 demonstrates how GC1 is capable of coping with the external disturbance in the monocular visual tracking task, being still able of achieving small errors within the image plane of the camera.

Unfortunately it was not possible to put GC2 to the same test as it eventually comes out that the source of stimulated disturbance brings GC2 into a region where it somehow ceases to work.

CHAPTER 4

SOFTWARE TOOLS DEVELOPMENT

4.1 Introduction

This chapter illustrates the design carried out to provide proper *YARP*-based tools such as C++ libraries and interfaces to handle the direct and inverse kinematic problems as well as provide a well structured environment to address the task space control of the robot limbs and gaze.

Throughout the period of the research object of this thesis a great effort and emphasis have been put in the software development with the goal of achieving a mature product characterized by a high-level of reliability and performance.

4.2 The *iKin* Library: Tools for Serial-Link Kinematic Chains

A library called *iKin* has been developed in order to provide a general framework for the solution of both forward and inverse kinematic tasks of generic serial-link chains.

iKin is a C++ *YARP*-based package available on the public repository, conceived to be as independent as possible from the mathematical routines *YARP* relies on, enforcing the concept of modularity and providing the interface with external software in the easiest way.

A great effort has been put into organizing *iKin* as a wide, reliable and flexible set of kinematic primitives fully documented[11] to be used to deal with any serial-link chains

[11] The documentation of iKin classes is available on line here:
http://eris.liralab.it/iCub/main/dox/html/group__iKin.html

(provided the description in standard Denavit-Hartenberg convention) with the target of becoming a major reference for the software community. In this respect, a number of European laboratories hosting an *iCub* have been already reported for being using *iKin* for research purposes; just to cite a few: the *Synthetic Perceptive, Emotive and Cognitive Systems* (*SPECS*) in Barcelona[12], the *Computer Vision Lab* (*VisLab*) at Istituto Superior Técnico in Lisbon[13], the U846 laboratory at *Stem-cell and Brain Research Institute* (*INSERM*) in Lyon[14], the *Learning Algorithms and Systems Laboratory* (*LASA*) at École Polytechnique Fédérale in Lausanne[15], the *Laboratory of Autonomous Robotics and Artificial Life* (*LARAL*) at the Institute of Cognitive Sciences and Technologies of the National Research Council in Rome[16], the *Centre for Robotics and Neural Systems* (*CRNS*) at the University of Plymouth[17], the *Adaptive Systems Research Group* at the University of Hertfordshire[18].

Moreover, the property of being a general purpose tool for forward and inverse kinematics allows *iKin* to be employed by researchers who study robotics with platforms different from *iCub*, such as the *BERT2* at *Bristol Robotics Laboratory* (*BRL*)[19]; in the near future the goal is to use *iKin* also with *Jido* and *HRP2* robots hosted by the *Laboratoire d'Analyse et d'Architecture des Systèmes* (*LAAS*) in Toulouse[20].

iKin consists of the following four separated modules:

---

[12] http://specs.upf.edu

[13] http://www.isr.ist.utl.pt/vislab

[14] http://www.sbri.fr (partner of the *CHRIS* project)

[15] http://lasa.epfl.ch

[16] http://laral.istc.cnt.it

[17] http://tech.plym.ac.uk/SoCCE/CRNS

[18] http://adapsys.feis.herts.ac.uk

[19] http://www.brl.ac.uk (partner of the *CHRIS* project)

[20] http://www.laas.fr (partner of the *CHRIS* project)

1) **iKinFwd**: comprises the definitions of the main C++ classes (link, chain, limb) devoted to the computation of forward kinematics, along with the required methods for adding and removing links, blocking and unblocking chain's degrees of freedom, manipulating joints constraints, querying for end-effector positions and analytical or geometric Jacobian relying on selectable notations for the orientation: the Euler angles or axis-angle representation.

2) **iKinInv**: collects a wide range of algorithms for finding the joints configuration which achieve a given end-effector pose to cope with the inverse kinematics problem; poses with only the translational part defined or poses with also specified orientation are both handled.

   Hereafter a list of the implemented algorithms is given:

   a) Steepest descent gradient with fixed gain;
   b) Steepest descent gradient with variable gain;
   c) Levenberg-Marquardt (*LM*) algorithm also known as the Damped Least Squares (Deo and Walker 1992);
   d) Levenberg-Marquardt algorithm with boundary functions implemented through a gradient projected method (Ho-Yul, Byung-Ju and Yungjin 2007).
   e) Some strategies based on the GNU Scientific Library (*GSL*), such as the Nelder-Mead simplex (Nelder and Mead 1965) and the Broyden-Fletcher-Goldfarb-Shanno (*BFGS*) method(Fletcher 1970).
   f) The Multi-Referential dynamical systems approach (see Chapter 2) addressing the control problem is provided within this module.

3) **iKinIpOpt**: contains the C++ classes required to interface with *IpOpt* to allow describing the inversion problem as a nonlinear constrained optimization task as detailed in Chapter 2.

4) **iKinSlv**: exports C++ classes for on-line solution of the inverse kinematic problem specifically tailored for the *iCub* limbs (i.e. the arms and the legs). On-line solvers run as daemons which connect to the robot to retrieve information on the current joints configuration (along with their bounds) and by requesting a desired pose with queries forwarded through *YARP* ports return the corresponding target configuration in the joint space.

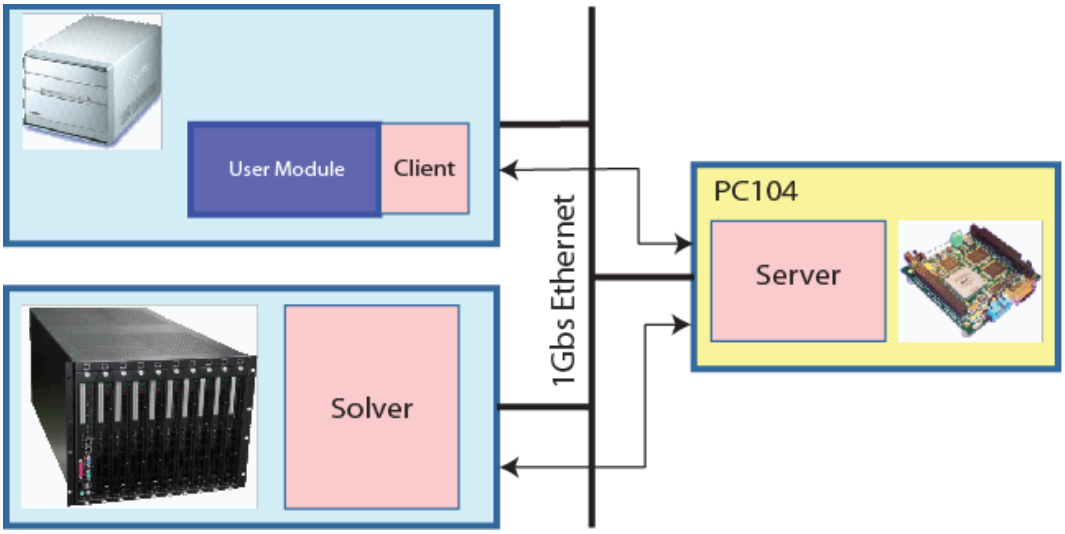

Figure 4-1. The modular Cartesian Interface architecture. Arrows are depicted representing the information flows exchanged between the three components of the interface, namely the client, the server and the solver.

As ending remark, it is worth mentioning that the code implementing the Gaze Controller structure exposed in Chapter 3 has been also developed based on the *iKin* package.

4.3 *YARP* Interfaces

To simplify use (and re-use) of the methods designed in Chapter 2 and Chapter 3,

YARP has been extended to support dedicated interfaces to the realized Cartesian and Gaze controllers. This extension consisted in widening the existing joint level motor control interfaces to include the task space control of the arm as well as the control of the gaze. The purpose was twofold: (1) achieve better modularity and (2) hide the implementation details of the controllers behind a set of simple interfaces methods.

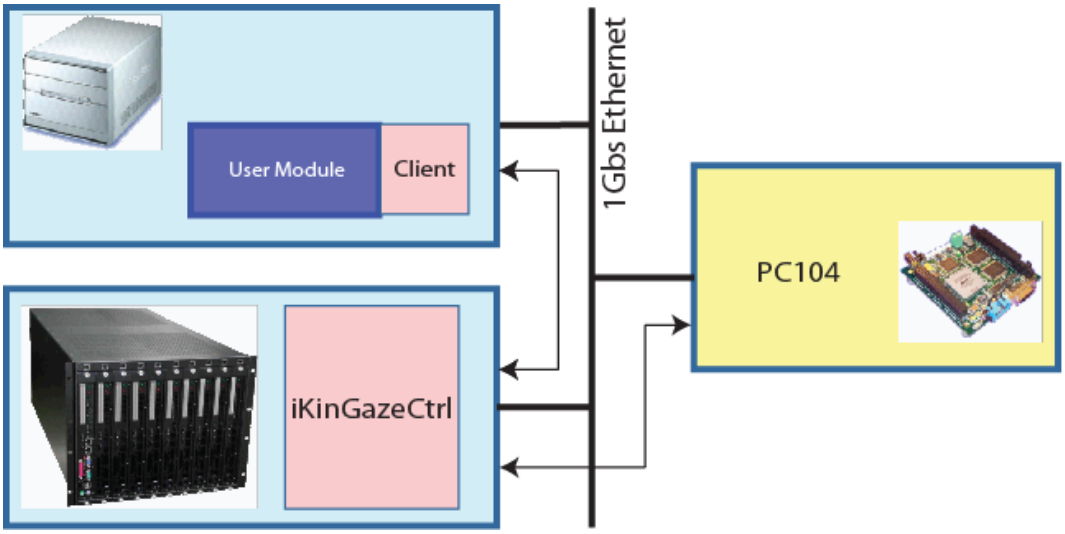

Figure 4-2. The Gaze Interface architecture. Arrows are depicted representing the information flows.

Concerning the Cartesian Interface[21] for the task space arm control, by separating the trajectory generator from the solver it is possible to run the controller part directly on the robot (i.e. on the PC104 hub) and reduce latencies associated with the network. We thus obtained a considerable gain in controller performance. Running the controller on board of the PC104 allowed attaining a factor of about 50% performance increase with respect to the standard *YARP* module accessing the iCub through the network. We adopted a

[21] The documentation of the Cartesian Interface is available on line here: http://eris.liralab.it/iCub/main/dox/html/icub_cartesian_interface.html.

client/server architecture (Figure 4-1). The client exports the Cartesian Interface as it is seen by the user. Accessing the client at this level is as easy as creating a C++ object and calling its methods. Through *YARP* the client implements the request of the user by dispatching them to the Server and, hidden to the user, the Solver object. The Server runs locally on the robot and implements the core of the control based on the Multi-Referential approach (since the controller sends velocity commands to the low-level hardware, latencies at this point are crucial). The hidden part is represented by the *S*olver which computes the set points for the Server using *IpOpt*; since this process is computationally expensive, it is advisable to run the Solver on a separated, powerful machine (as the output of the Solvers are set-points, latencies at this level are less critical).

Likewise, the Gaze Interface[22] has been devised in order to enable the user to control the robot gaze resorting to simple C++ methods calls, which in turn hide the protocol details and the information marshalling inside the implementation. Differently with respect to the Cartesian Interface architecture, the Gaze Interface does not need a distinction between the Solver and the Server parts, having the two components encapsulated in the same module *iKinGazeCtrl* (see Figure 4-2) running on the cluster. This is motivated by the fact that the resources aboard the PC104 are somewhat limited and to a certain extent at least half of the mechanical inertias the gaze controller has to deal with – e.g. the *iCub* eyes – do not require stringent closed-loop reactivity in velocity mode.

Nevertheless, by virtue of *YARP* modularity approach, the module *iKinGazeCtrl* can be launched on the robot hub with no additional design effort, when for example due to constant improvements in the technology the PC104 will be replaced by more powerful boards.

[22] The documentation of the Gaze Interface is available on line here:
http://eris.liralab.it/iCub/main/dox/html/icub_gaze_interface.html

4.4 Grasping: The Action Primitives Library

To tackle the problem of grasping we focused on a typical scenario described in the context of the *CHRIS* project: a human and a robot performing cooperative actions on objects placed on a table (Figure 4-4). The design of the grasping architecture has been broken down in two main parts: A *reaching* module, whose responsibility is to bring the hand of the robot close to the object, and a *grasping* module that controls the movement of the fingers and detects if and when the fingers apply enough force to the object surface.

We have thus developed a library that relies on the *YARP* Cartesian Interface and realizes a further abstraction layer that expose to the user a collection of action primitives (such as *reach*(), *grasp*(), *tap*(), …) along with an easy way to combine them together to form higher level actions and eventually execute more sophisticated tasks.

Central to the Action Primitives library[23] is the concept of *action*. An action is a "request" for execution of three tasks (as depicted in Figure 4-3):

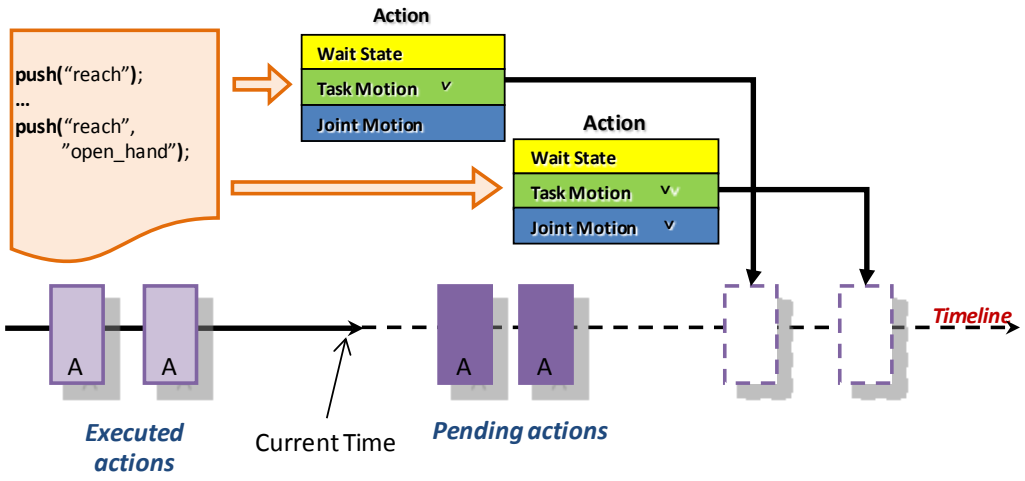

Figure 4-3. A diagram explaining the core characteristics of the Action Primitives library.

[23] The documentation of Action Primitives library is available on line here:

http://eris.liralab.it/iCub/main/dox/html/group__ActionPrimitives.html

1. It can ask to steer the arm to a specified pose, hence performing a motion in the task space;
2. It can command the execution of some predefined finger sequences in the joint space and identify it with a tag;
3. It can ask the system to wait for a specified time interval;

Evidently, the arm motion in the task space (task of type 1) is executed exploiting the Cartesian controller devised in Chapter 2.

Besides, the library offers the possibility to specify if the action involves the execution of a task of type 1 simultaneously with a task of type 2. Action requests are stored in an *action queue* and served with a First In First out policy. The idea is to provide a mechanism to program and execute sequences of actions as combinations of movements of the arm (type 1), movements of the fingers (type 2) or coordinated movement of both arm and fingers (type1+type2).

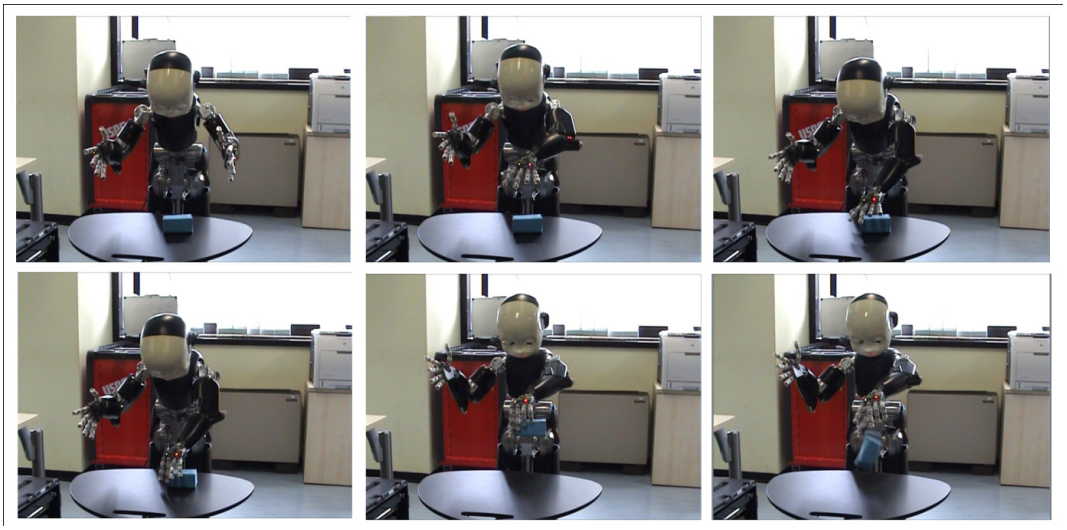

Figure 4-4. A grasping sequence. From top-left to bottom-right, an object is placed on the table, the robot moves the hand above the object and closes the hand. When a successful grasp is detected the robot lifts the object and eventually drops it.

An important aspect to point out is that at the moment the finger positions that realize the grasp actions should be known (and tuned) in advance. However we are planning to include a learning stage that eventually will allow to automatically adapting the motion of the fingers to novel objects. In Figure 4-4 we report a sequence that shows the robot grasping an object.

The Detection Problem

In absence of tactile feedback Nori (Schmitz, et al. 2010) designed an algorithm that performs contact detection using the springs mounted in the distal phalanges of the hand. Due to the elastic coupling between the phalanges of the fingers[24], the fingers passively adapt when they encounter an obstacle (i.e. when they touch the surface of an object). The amount of "adaptation" can be indirectly estimated from the encoders on the joints of the fingers. In other words, the idea is to measure the discrepancy between the finger motion in presence of external obstacles (e.g. objects or the other fingers) and the one that would result in normal operation (in absence of obstacles/free movement). In a calibration phase we estimate the (linear) relationship between the joints of the fingers in absence of contact. In normal operation, we detect contact by comparing how much this model fits the current encoder readings.

Calibration Phase

The purpose of the calibration is to fit a linear model to the encoders of the joints that are mechanically coupled. In order to be as general as possible, it is assumed that the data are generic n dimensional vectors, here indicated as $q \in \mathbb{R}^n$. The dimension n depends

[24] On the iCub fingers are made up of three phalanges that we call proximal, middle and distal. With the exception of the ring and little fingers, the middle and distal phalanges are mechanically coupled with an elastic element and actuated by a single motor. In the ring and little fingers all phalanges are mechanically coupled, and a single motor is responsible for the actuation.

on the number of joints actuated by a single motor: for the thumb, index and middle fingers $n = 2$, whereas for the ring and little fingers $n = 6$. The parametric linear model used to fit the data is:

$$r: \quad q = k_0 + k_1 \cdot t, \qquad t \in [t_{\min}, t_{\max}], \qquad (4.1)$$

where t is the free parameter to be chosen in $\left[t_{\min}, t_{\max}\right]$. Another possible representation for this model can be obtained by observing that it represents a line r (i.e. a one dimensional subspace) embedded in a n dimensional vector space. Therefore, it can be represented as the intersection of planes. The general implicit equation of a plane π in $\mathbb{R}^n$ is:

$$\pi: \qquad a_1 q_1 + a_2 q_2 + ... + a_n q_n = c. \qquad (4.2)$$

Since the plane is a $n-1$ dimensional subspace, the intersection of k planes is a $n-k$ dimensional subspace. Therefore, a line r can be represented by intersecting $n-1$ planes:

$$r: \begin{cases} a_{2,1} q_1 + a_{2,2} q_2 + ... + a_{2,n} q_n = c_2 \\ \vdots \\ a_{n,1} q_1 + a_{n,2} q_2 + ... + a_{n,n} q_n = c_n \end{cases}. \qquad (4.3)$$

However, the representation of a line by means of the coefficients in (4.3) is redundant. It can be shown that an equivalent non-redundant implicit representation is the following:

$$r: \begin{cases} a_{2,1} \cdot q_1 + q_2 + 0 \cdot q_3 ... + 0 \cdot q_n = c_2 \\ \vdots \\ a_{n,1} \cdot q_1 + 0 \cdot q_2 + ... + 0 \cdot q_{n-1} + q_n = c_n \end{cases}, \qquad (4.4)$$

which corresponds to the following parametric model:

$$r: \quad q = k_{\mathbf{0}} + k_{\mathbf{1}} \cdot t, \qquad (4.5)$$

with $t = q_1$ and:

$$k_{\mathbf{0}} = \begin{bmatrix} 0 \\ c_{22} \\ \vdots \\ c_n \end{bmatrix}, \; k_{\mathbf{1}} = \begin{bmatrix} 1 \\ -a_{2,1} \\ \vdots \\ -a_{n,1} \end{bmatrix}. \qquad (4.6)$$

Given a set $q^1, \ldots q^N$ of observations, the parameters $a_{i,1}$ and c_i are estimated by solving the following least squares optimization:

$$\min_{\substack{a_{2,1},\ldots,a_{n,1} \\ c_2,\ldots,c_n}} \sum_{i=1}^{N} \sum_{j=2}^{n} \left\| a_{j,1} \cdot q_1^i + q_j^i - c_j \right\|^2. \qquad (4.7)$$

These parameters are then converted to k_0 and k_1 by means of equation (4.6). Moreover, points of the minimum distance are computed as follows:

$$t_j^* = \arg\min_t \left\| k_{\mathbf{0}} + k_{\mathbf{1}} \cdot t - q^j \right\|, \; j = 1 \ldots N, \quad (4.8)$$

which corresponds to:

$$q^{*,j} = k_{\mathbf{0}} + k_1 \cdot t_j^*, \; j = 1 \ldots N. \qquad (4.9)$$

Accordingly, the span of the allowed values for t is determined as follows:

$$t_{\max} = \max_{j=1\ldots N} t_j^*, \; t_{\min} = \min_{j=1\ldots N} t_j^*. \qquad (4.10)$$

Similarly, the maximum and the minimum distance from the model is computed as:

$$q_{\max} = \max_{j=1\ldots N} \left\| q^{*,j} - q^j \right\|, \; q_{\min} = \min_{j=1\ldots N} \left\| q^{*,j} - q^j \right\|. \qquad (4.11)$$

Figure 4-5 shows the result of an example calibration carried out on the two distal joints of the *iCub* thumb. The two joints are represented by blue dots: clearly, they roughly lie on a linear manifold represented by the red line, whilst the distances from this manifold are depicted with green lines. Therefore by measuring the deviation from the linear manifold and by applying proper thresholds, this method allows to detect contacts with objects during grasping tasks and to safely stop the fingers preventing damage.

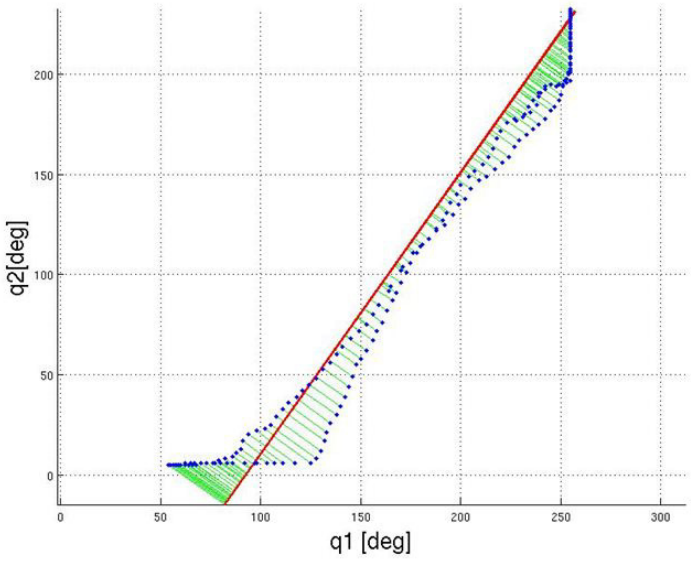

Figure 4-5. The linear manifold that models the coupled distal joints of the thumb.

CHAPTER 5

TACKLING THE ADAPTION PROBLEM WITH MACHINE LEARNING: PRELIMINARY RESULTS

5.1 Introduction

The Cartesian Interface presented in Chapter 4 and conceived to control the *iCub* limbs embeds a structure that relies deeply on a model-based methodology taking advantage of the *a priori* knowledge of the body schema. Likewise, an analogous interface has been designed to cope with the control of the robot gaze, making use of the same kind of approach. As result of such good representation of the mechanical parts, the produced movements are fast, precise, reliable and repeatable over time, and perhaps more importantly we are able to take into account a number of constraints specific to the platform such as the mutual bounds of the shoulder's tendons lengths and the self-collision issue that let the robot interact with the surrounding environment safely, accomplishing complex visuo-haptic tasks.

The attained performances are undoubtedly higher than the results obtained by applying pure learning-oriented algorithms both in its off-line and on-line implementation, considering robustness, repeatability and reliability issues. On the other hand, any method that exploits only an assigned and fixed description of the reality cannot compensate for unmodeled dynamics and unknown offsets that may affect the computation causing sensible degradation in how precise the task is executed in real circumstances. A meaningful example of such scenario refers to the light mechanical misalignment of the relative position of the iCub eyes with respect to nominal configuration that is responsible along with the uncalibrated stereo cameras for generating heavy perturbations in tasks where the robot is requested to reach for a target whose

position in the operational space is computed by the vision system.

Many other references can be provided to comparable situations for which techniques resorting to conventional control theories do not suffice to achieve the objectives satisfyingly. For this reason, given the model-based layer that ensures good performances, the ultimate goal would be to put on top a further machine learning (*ML*) oriented structure capable of adapting the behavior to unknown quantities and changes in the environment.

A profitable example of such interaction between the model design approach and the *ML* world is the autonomous learning of eyes-hand coordination that would improve the reaching accuracy by building a map which relates the points in the cameras image planes and the proper offsets required to compensate for the final target as the arm moves in the 3D space. The learning of the map can be effectively performed on-line by the robot in an autonomous manner (e.g. with Support Vector Machines (Gijsbert, Metta and Rothkrantz 2010)) by following its own hand using for example reliable cues such as motion, without actually retaining any markers as, conversely, it is usually done in the literature (Hersch, Sauser and Billard 2008), (Nori, Natale, et al. 2007). This desirable feature that surely enriches the robot capabilities with a preliminary self-adaptive behavior is the matter of this chapter together with the investigation of a further task that has been explored in the same direction since it well addresses the set of scenarios envisaged within the *CHRIS* project, and specifically: the possibility to construct on-line a simple 2D map to adjust the errors the robot naturally commits while reaching for a target placed on a table, exploiting at the same time the available force control routines to realize a sort of coaching phase that an operator carries out directly on the robot with the aim to teach it the correct position by simply exerting forces on the limbs.

As evident from the title of the chapter and highlighted previously in this paragraph, the results attained hereafter, even with their relevance, have to be considered as a very first attempts towards an emerging perspective where model-based and learning-oriented policies can be usefully integrated and combined in robotics, paving the way in long term for promising future works.

Nevertheless, prior to going deeply through the descriptions of these two "blending" experiments it is necessary to provide a brief outline of the available force sensing features employed in the *iCub* for the human-robot teaching exercise and, rather, to give more details about the solution developed for the motion detector which has been extensively used for the autonomous eyes-hand markerless coordination.

5.2 Force/Torque Sensing

The objective of force/torque sensing within the current scope is to implement in the robot the *active force control* that aims at modifying software wise the compliance of the manipulator to satisfy the requirements of different interaction tasks, such as the contact detection and the coaching.

To achieve active compliance standard industrial approaches employ an F/T sensor located at the end effector of the manipulator. The obvious assumption in this case is that the robot interaction with the environment only occurs at the tool level.

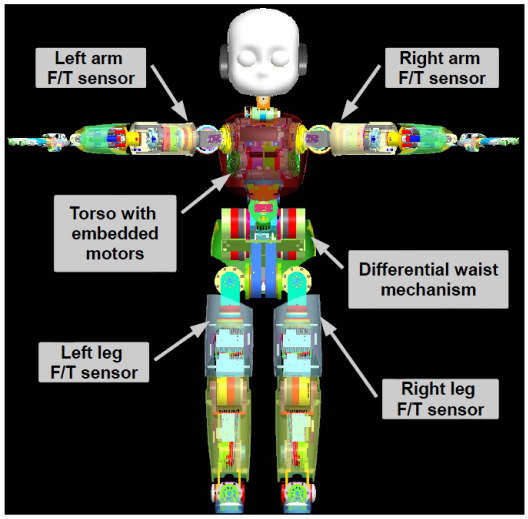

Figure 5-1. A schematic representation of the *iCub* showing the location of the force/torque sensors.

The *iCub* robot instead is arranged with a solution that embodies both the benefits of F/T and joint torque sensing (Metta, Vernon, et al. 2008). It mounts indeed a 6-axes F/T sensor at the beginning of the kinematic chain of each limb as it is visible in Figure 5-1; this makes the robot sensitive also to external forces occurring at different level (not only at the end-effector) and also gives information about the real torque acting on the joint, due to the limb dynamic. The drawback is that, if not compensated, the internal dynamic forces are detected as external forces (something that does not happen when the F/T sensor is placed at the end-effector). Moreover, to properly balance external forces acting on the whole arm, the knowledge of their point of application is required. The latter information is not available without tactile feedback but due to the nature of the experiments described in the following it is plausible to assume that all external forces are applied at the end-effector.

The method that provides the estimation of contact forces detection is explained in (Fumagalli, Randazzo, et al. 2010) and is briefly detailed hereafter.

The wrench F_s measured at the sensor reference frame denoted <*s*>, is the sum of two terms, one due to the limb internal dynamics, the other due to an external wrench F_e (see Figure 5-2). As previously pointed out, the assumption here is that F_e is the unique external perturbation and that this perturbation is applied in a known reference frame, denoted <*e*>, attached at the end-effector. Expressing all these quantities in <*s*>, we can conclude that the measurement $F_s \in \mathbb{R}^6$ can be decomposed as follows:

$$F_s^s = F_e^s + F_i^s, \qquad (5.1)$$

where $F_i^s \in \mathbb{R}^6$ is the contribution of the manipulator internal dynamics to our measurements F_s^s. Moreover, it can be shown that (Sciavicco and Siciliano 2005):

$$F_i^s = M_F\left(q\right)\ddot{q} + C_F\left(q,\dot{q}\right)\dot{q} + g_F\left(q\right), \qquad (5.2)$$

where $q \in \mathbb{R}^n$ is the generalized coordinate's vector describing the configuration of the

limb, here assumed composed of n DOF. The internal force F_i^s depends on joint position q, velocities $\dot{q}$ and acceleration $\ddot{q}$ and can be decomposed in the sum of an inertial term $M_F\left(q\right) \in \mathbb{R}^{6\times n}$, a gravitational term $g_F\left(q\right)$ and a Corioli's term $C_F\left(q,\dot{q}\right)$.

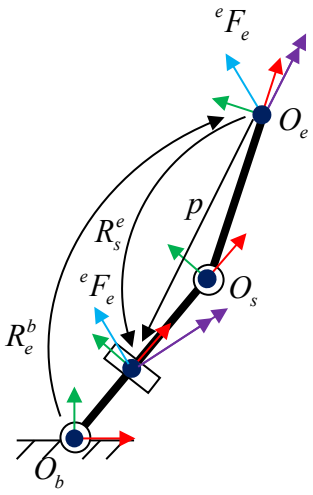

Figure 5-2. A sketch depicting the wrench measured at F/T sensor's location along with its projection at the end-effector. The sensor <*s*>, base <*b*> and end-effector <*e*> frames are also reported.

In (Fumagalli, Gijsberts, et al. 2010) a method to approximate the internal dynamic equation (5.2) is provided relying on a parameter estimation technique, so that the measure F_e^s is simply given by:

$$F_e^s = F_s^s - F_i^s\left(q,\dot{q},\ddot{q}\right). \quad (5.3)$$

The Action Primitives library conveniently encapsulates this force control layer making available to the user the possibility to sense contact with external elements of the environment.

5.3 Independent Motion Detection

The *independent motion detection* is the discipline concerned with the automatic identification of independently moving objects within a scene acquired by a moving camera. This is the typical scenario the robot has to tackle when it gazes at its moving hand for the task described in the first paragraph: a motor babbling of the end-effector generates motion cues within the pair of the equipped cameras that in turn can be exploited to relate the location of the projected points in the image planes with the 3D position of the hand in the Cartesian space. Since the head is supposed to widely moves while keeping the hand in fixation it is necessary to rely on a robust algorithm capable of eliminating the effect of the egomotion in the scene.

Presently in literature (and interesting survey has been conducted by (Irani and Anandan 1998)) not many algorithms cope with the stringent requisite of being real-time compliant and, more eminently, their implementation are not currently accessible.

The proposed method stems from an analysis of the optical flow problem with a particular emphasis on the approach exploited by the Lucas-Kanade algorithm (Lucas and Kanade 1981). In the following an exhaustive overview of this classic technique is presented, followed by the idea which led to the definition of our framework.

It is important to underline that coauthor of the design and the development of the framework along with all the analyses reported in this section was Carlo Ciliberto, a PhD student researching at *RBCS* department of *IIT*, with whom a constant, tight and fruitful collaboration has come about.

Lucas-Kanade Optical Flow

The study of optical flow deals with the problem of evaluating motion across streams of images. Such motion is usually induced on the image plane by the actual dynamics interesting the elements in the observed scene. A typical example illustrating this situation is a robot exploring its surroundings and consequently generating apparent motion in the image stream acquired from the embarked cameras. In these contexts it can

result necessary to understand exactly how the observed motion takes place. This can be partially inferred from the analysis of the optical flow, which is defined as the field of instantaneous velocities on the image plane reference frame.

Under the reasonable assumption that image sampling is performed at sufficiently high frequency, it can be accepted that the *appearance continuity* property holds. In other words, it is legitimate to expect that within small time intervals, the appearance of elements in the scene does not vary dramatically and it is thus possible to track points across subsequent frames based exclusively on their visual appearance. This property can be expressed more formally through the equation

$$I\left(p\left(t\right),t\right) = I\left(p\left(t+\delta t\right),t+\delta t\right) \text{ with } \delta t \ll 1, \quad (5.4)$$

where $I\left(\cdot,t\right)$ is the image function at time t (usually representing pixel brightness) and $p\left(t\right)$ is the projection on the image plane at time t of a point in the scene.

Deriving the image function $I\left(p\left(t\right),t\right)$ with respect to time, we obtain:

$$\nabla_p I \cdot \dot{p} + \frac{\partial I}{\partial t} = 0, \quad (5.5)$$

where ∇_p is the vector of the derivatives with respect to the u and v main directions of the image plane and $\dot{p}$ represents the planar velocity of point p. As can be noticed, equation (5.5) has an infinite number of solutions, implying that the sole punctual information is insufficient to determine the exact velocity of any point on the image plane. However, it can be assumed that locally points behave similarly and thus that in sufficiently small neighborhoods, instant velocities are almost identical. Velocity $\dot{p}$ of a point p can hence be approximated by setting a window W around it and then solving the least-squares minimization problem:

$$\hat{\dot{p}} = \min_{w\in\mathbb{R}^2} \left\| I_p w - I_t \right\|^2, \quad (5.6)$$

where I_p and I_t are matrices whose rows are respectively the gradient $\nabla_p I$ and the derivative $\partial I / \partial t$ computed on each point w in W.

When the left pseudoinverse $I_p^\dagger$ of I_p exists, equation (5.6) is solved by taking $\hat{\dot{p}} = I_p^\dagger I_t$. Existence of $I_p^\dagger$ is not always guaranteed depending on the spatial appearance of the neighbors of p. In particular, it can be shown that in order for $I_p^\dagger$ to exist, both partial derivatives of the image along the u and v axes need to be different from zero in some point of the nearness of p.

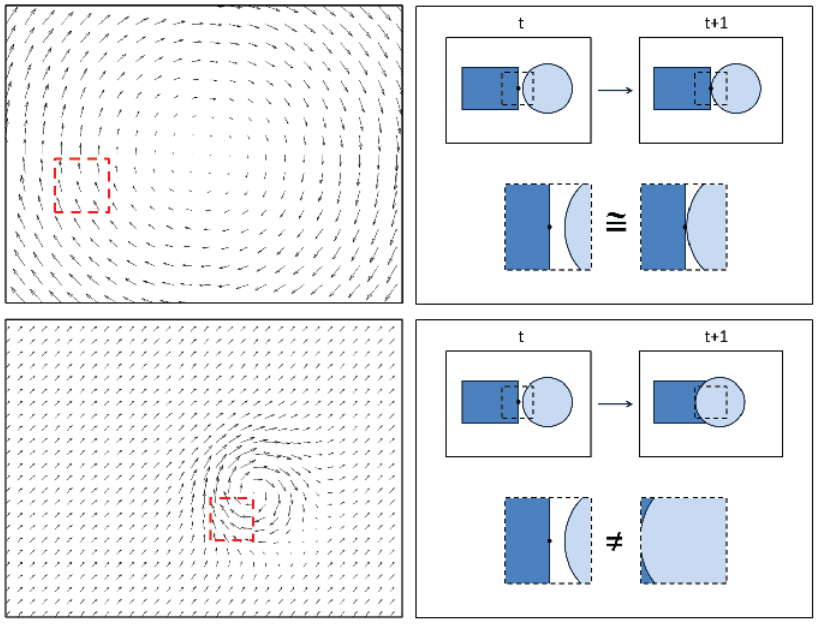

Figure 5-3. Illustration of typical failure conditions in the optical flow computation: (a) Egomotion rotations (top) usually induce homogeneous and uniform flows on the image stream while independent rotations (bottom) frequently cause the Lucas-Kanade assumption of local constant velocity to not usually hold (the red square windows encompass a detail of such behavior). (b) Failures due to occlusion depend on the relation between image sampling frequency and the speed at which such event happens: if the occluding object is moving slowly (top), differences in the neighborhood of the occlusion are smaller than those exhibited in the case of higher velocities (bottom).

Failure Analysis

From the discussion above, corners appear the most suitable points over which compute optical flow with the Lucas-Kanade algorithm. However in practice, even corner tracking is not always successful. As a general rule, in order to verify that the optical flow of a point has been correctly estimated, the original image patch around that point is compared to the patch in the new image where the point is supposed to have been moved. A similarity measure is then used to evaluate whether tracking is correctly performed or not. It is thus interesting to analyze in general when the Lucas-Kanade algorithm fails and why. Conclusions from this investigation will lead directly to the method we are proposing to perform independent motion detection.

The main circumstances in which errors in the evaluation process of the optical flow arise are three:

1. The instantaneous velocity of the point is too large with respect to the window where motion is being considered. Hence I_t loses its meaning of temporal derivative.
2. The motion around the point has a strong rotational component and thus, even locally, the assumption regarding the similarity of velocities falls.
3. The point is occluded by another entity and obviously it is impossible to track it in the subsequent frame.

The first type of tracking failure depends exclusively on the scale of the neighborhood where optical flow is computed. This issue is usually solved by the so-called *pyramidal approach* (Bouguet 2004) which applies the Lucas-Kanade method at multiple image scales. This allows evaluating iteratively first larger velocities and then smaller ones.

The second kind of tracking failure is generally a consequence of motion independent from the observer. Figure 5-3(a) depicts a comparison between the typical rotational effects on the image generated respectively by egomotion (upper image) and independent motion (lower image). As can be noticed, egomotion has a global effect on the image

stream, producing optical flows that result smooth and locally almost constant. On the other hand, independent motion affects only specific areas of the image plane, introducing local distortions when it is the case of rotations.

The third situation in which Lucas-Kanade fails is caused by an object moving (actively by independent motion or passively by egomotion) between the observer and the target point. In this context the main role in determining whether optical flow has been successfully computed is played by the speed at which such occlusion takes place. As shown in Figure 5-3(b), if the occluding object moves slowly with respect to the image sampling frequency, the window around the target point remains almost unaltered even when the point disappears behind the object. However, as the event happens faster, larger portions of the neighborhood get covered, eventually causing the similarity measure between original and tracked window to decrease.

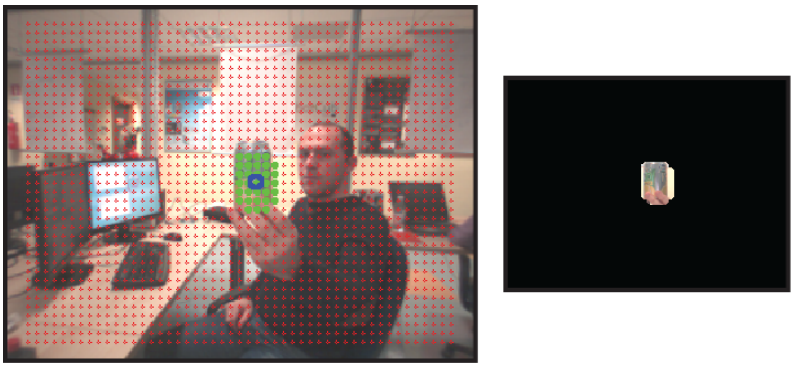

Figure 5-4. (*Left*) The uniform static grid of nodes depicted in red where computing the standard Lucas-Kanade algorithm according to the new proposed method. The operator waves a can generating independent motion cue that in turn stimulates the superimposed nodes resulting in green; the centroid of the detected blob is also automatically extracted and highlighted in blue. (*Right*) The output of *motionCUT* can drive a coarse segmentation of the moving object.

It has to be noted that also in this context, errors in the optical flow evaluation are usually caused by independent motion. This depends on the fact that aside from

pathological cases, elements in the scene lie generally sufficiently far from the observer so that the process of occlusion due to egomotion happens too slowly to allow the similarity measure to go under the predefined threshold.

The Cover-Uncover Trick

We propose a method to detect independent motion which derives directly from our observations on optical flow[25]. As a matter of fact, the fundamental idea underlying our approach is to take into consideration the points in the image where the Lucas-Kanade algorithm fails, as they are likely to identify image patches where independent motion is actually occurring, rather than those over which it succeeds.

The main concern regarding this analysis is the selection of those points over which try to compute the optical flow. While corners are the best choice for the standard Lucas-Kanade algorithm thanks to their structure, in our context we look for points over which tracking is likely to fail as soon as one of the conditions discussed previously is met, i.e. the flow inconsistencies due to rotations or occlusions. With these premises we named our method *motionCUT*, being *CUT* the acronym for *Cover-Uncover Trick*, a concise yet evocative definition that captures the essence of the procedure revealing its simplicity: independent moving agents in the scene can be easily recognized by inspecting the effect of their motion at the frontier with fixed elements of the environment (see Figure 5-3(b)) where episodes of occlusion of parts of the background (cover) and their corresponding re-emergence (uncover) mostly appear.

To this end, we consider a uniform grid of points placed on the image plane and compute the optical flow with these grid nodes as pivots (Figure 5-4). In this framework point selection is done regardless of their appearance and thus, the majority of them will not be corners but rather weaker points in the sense of Lucas-Kanade tracking.

[25] We actually came across a promising indication of the benefit of using a static grid for motion detection while we were struggling with the ordinary Lucas-Kanade method during nightly experiments.

Ideally we could consider each image pixel as the node of a dense grid. However, as experimental results on independent motion detection show similar results for relatively sparser grids, it is often wiser to ease the computational effort by choosing not too dense grids.

Nonetheless, the process of taking the elements of the grid depending only on their position expose the system to the risk of having points where the pseudoinverse of I_p does not exist and it is therefore impossible to compute the optical flow. However, as mentioned earlier, this happens for points whose neighborhood have a specific shape: regarding pure edges, by virtue of Equation (5.5) we can obtain the patch's velocity along the direction normal to the edge itself and try to approximately track the point using this information. For uniform image patches though, motion cannot be estimated and it is just assumed to be zero.

As noticed, the points of the regular grid are quite susceptible to motion inconsistencies, responding positively - in the sense that the optical flow computation fails over them - when they lie near to an image patch where independent motion is occurring. In order to eliminate false positives we take in consideration only areas of the image plane where clusters of nodes respond contemporaneously. This implies that if a single point is responding but no other point near it is, that point is considered a false positive and it is eliminated from the list of possible locations where independent motion is occurring.

Our independent motion detection procedure can be finally described by the following steps:

- A uniform grid is placed over the image and for every couple of consecutive frames the Lucas-Kanade optical flow is computed over each grid element.
- The nodes over which Lucas-Kanade fails are taken as potential independent motion locations. In order to eliminate temporary false positives, only points which are located nearby at least n other positive responding points (where n is a parameter chosen *a priori*) are considered locations of the image over which independent motion has occurred.

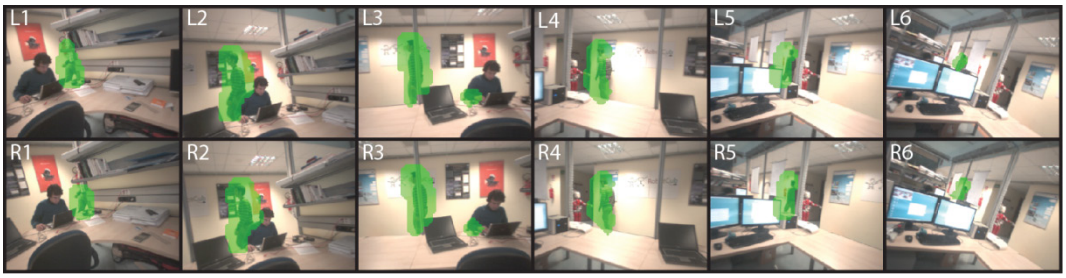

Figure 5-5. A strip of images recorded during a real time stereo tracking of a walker: six images are shown in their temporal sequence from L1 to L6 as taken from the left camera, whereas images from R1 to R6 represent the corresponding acquisitions from the right camera. The walking person is highlighted with a green blob using the result of *motionCUT* detection.

The utmost plainness of the *motionCUT* concept is very well suited for one of the main goals of this work that is to achieve independent motion detection in a real-time architecture. Accomplishing Lucas-Kanade computation in a traditional fashion foresees indeed the dynamic identification of the corner points and the successive least-squares solution of Equation (5.6); by contrast, our method discards precisely the first critical searching phase by adopting a static grid of nodes over which the least-squares computation is performed. As result of this further improvement, *motionCUT* can effectively run in real-time[26] processing input images at the same frame rate of the cameras acquisition, that is 30 Hz, as reported in Figure 5-5 for a traditional stereo tracking task of a walking person in a highly cluttered background.

5.4 Adaption to Reaching Errors in *CHRIS* Scenarios

The typical scenario foreseen within the *CHRIS* project where the robot is requested to

[26] A multi-core Intel (R) Xeon machine with 2.27 GHz of clock frequency has been employed for the experiments. Remarkably, on this machine we measured an average processing time for a single image of 15 ms, meaning that *motionCUT* is capable of running up to 66 Hz.

reach for objects and cooperate with humans for a shared goal is vividly depicted in Figure 5-6: small toys such as colored cars, sponge balls and boxes with handle are placed on top of a table to then let the robot grasp them in order to change their location or execute higher level actions as for instance the cover/uncover game.

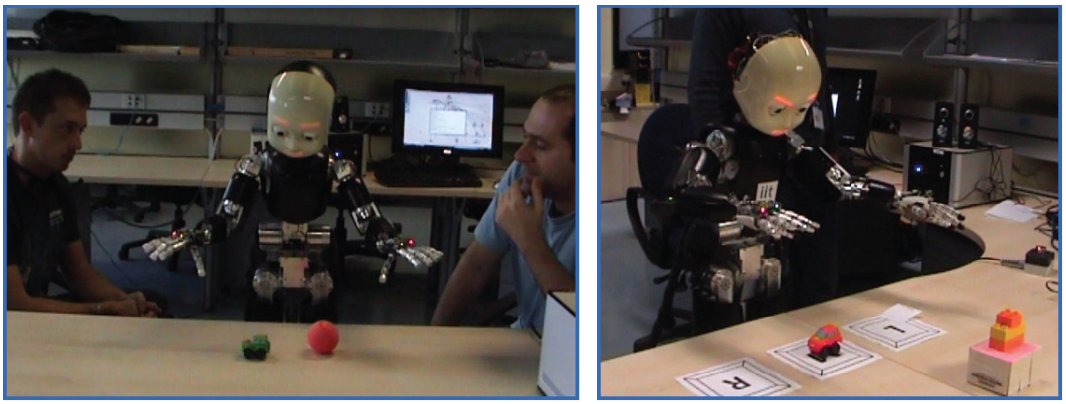

Figure 5-6. (*Left*) A snapshot showing *iCub* interacting with small toys and humans in the context of *CHRIS* scenario. (*Right*) A scene of the cover/uncover game is captured where *iCub* has to detect the toy car to be covered with the box present in the bottom-right corner of the image.

A requisite to allow the robot interact with the environment is to acquire and learn the visual appearance of the objects of interest. Interestingly, in this regard, the *motionCUT* can be profitably adopted to carry out a rough segmentation based on the motion cue, which then can guide the successive acquisition of the object template through the commercial vision system *SpikeNet*™ (Thorpe, et al. 2004), a real-time package that uses a spiking neural network technology to provide fast recognition of objects in an image. As exemplified in Figure 5-7, in fact, the operator waves the toy in the view field of the robot in order to catch its attention; the output of *motionCUT* is capable of defining a suitable area encompassing the object that is finally refined by a conventional *Canny* algorithm for edge-detecting (Canny 1986), exploiting the fact that the background appearance is fairly uniform. The result of this process is directly fed into the *SpikeNet*™ network.

Figure 5-7. The process of object's template acquisition is represented in these three successive pictures. (*A*) The operator captures the robot's attention generating motion cue with the object. (*B*) The output of motion detection is employed for a coarse segmentation of the object. (*C*) The template is refined and acquired by *SpikeNet*™.

Contact Detection with Environment

Notably, under the assumption that the robot manipulates objects on a flat table, it is convenient to employ a homographic projection to estimate the 3D Cartesian coordinates of the target to be attained from the 2D coordinates of the template's centroid as provided by a monocular vision perception system[27]. To this end, it is necessary to know the relative height of the table with respect to the root reference frame of the robot.

Clearly, having an embodied platform, it would be preferable and valuable to avoid any static calibration procedure aimed at finding this relative distance which is obviously subject to the contingent components used for the apparatus: e.g. the type of the table and the configuration of the robot, particularly if it stands on a pole or not. Conversely, the *iCub* should be able to autonomously explore its "playground", adapting to the unknown height of the said table surface. Figure 5-8 reports how the most intuitive way can be engaged to proactively discover this quantity: *iCub* simply touches the table being capable of stopping its motion as soon as a contact is detected by the force sensing

[27] Differently from the learning of eye-hand coordination described in the following, within the *CHRIS* project it has been decided to pursue a monocular approach.

routine embedded within the Action Primitives library that in turn monitors whether the current value of the external force estimated at the end-effector has overcome a safety threshold. The relative height of the table is thus elicited relying directly on the actual position of the end-effector when in contact.

Intriguingly, the basic yet practical solution proposed for the table height estimation permits to face also the contact detection problem relative to objects of interest the robot needs to grab, as demonstrated by the images sequence in Figure 5-9. Right after the detection of the contact with the object, the robot can lift its hand of a given suitable quantity to then operate opportunely the grasp closure with the fingers in the correct position with respect to the object.

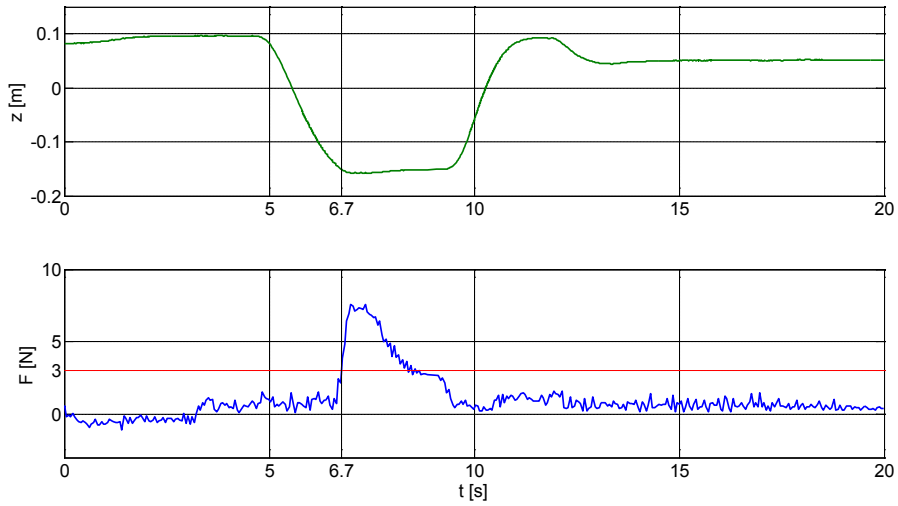

Figure 5-8. Table contact detection. (*Top*) The component of the end-effector Cartesian position accounting for the height with respect to the ground (z axis) is shown. (*Bottom*) The magnitude of the external force acting on the end-effector is depicted together with the safety threshold fixed at 3 N. The exploration starts at t=5 s, whereas the contact event happens at t=6.7 s, causing the system to recognize the table location at approximately -0.15 m along the z axis.

The Coaching Stage

Despite the initial exploration *iCub* carries out, knowing just the table height usually do not suffice to extract an optimal 3D target location for executing a successful grasp. This is due to two basic reasons: (1) one main account regards how the errors in the model of the kinematic parts as well as the pinhole camera affect the final result; (2) moreover, the fact that the height of the objects placed on the table remains unknown to the vision perception system (before completing the first exploration) definitely introduces further uncertainty in the homographic projection.

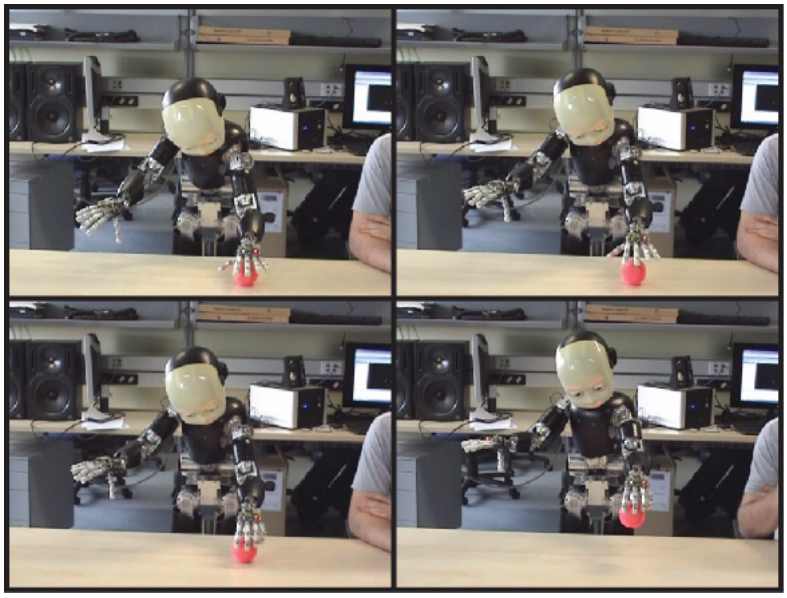

Figure 5-9. From top-left to bottom-right: *iCub* employs the same principle underlying the table height estimation in order to sense contacts with the pink sponge ball. Right after the detection with the toy, *iCub* lifts a bit its hand to suitably perform a grasp.

To sidestep this inconvenient and increase the chance of success in accomplishing the task, the following online coaching procedure has been beneficially exploited:

1. Through a dedicated command the robot enters the learning stage and the position of the identified object to be reached is stored in memory for a

successive recall.

2. The "teacher" starts the demonstration by exerting an external force directly on the robot's end-effector with the intent to drive it towards the position he reckons is the best for the grasping task (Figure 5-10).

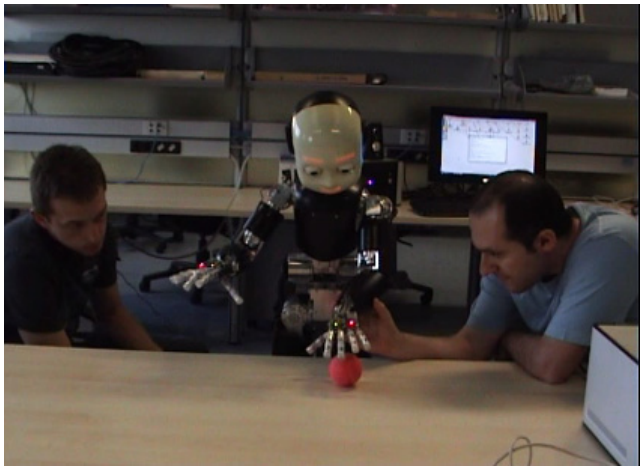

Figure 5-10. An excerpt of the coaching phase where the operator teaches the robot the correct Cartesian position for the object to grasp.

3. At the same time the robot perceives the external force $F = F\left(x, y, z\right)$ and moves the limb contextually in the direction of F by implementing an admittance control in the operational space, for which the end-effector velocity is guided by the external forcing term (Figure 5-11). More particularly, the admittance control whose outcome is the commanded task velocity $v = v\left(x, y, z\right)$ obeys the following law[28], with m and β suitable inertia and damping parameters, respectively:

[28] Integrating twice the equation (5.7) leads to the computation of the trajectory $x\left(t\right)$ of a virtual point in the operational space; this trajectory serves as the reference for the Cartesian control in a traditional position tracking task.

$$m\dot{v} = F - \beta v. \qquad (5.7)$$

4. As result of the combined actions of step 2 and 3, the robot hand is finally moved on top of the object, compensating the unmodeled errors with the help of human operator. Therefore, it is possible to update an internal 2D Cartesian map that associates the compensation offset to the object position as originally provided by the vision system in step 1.
5. The learning phase can be terminated at this point with a specific command.

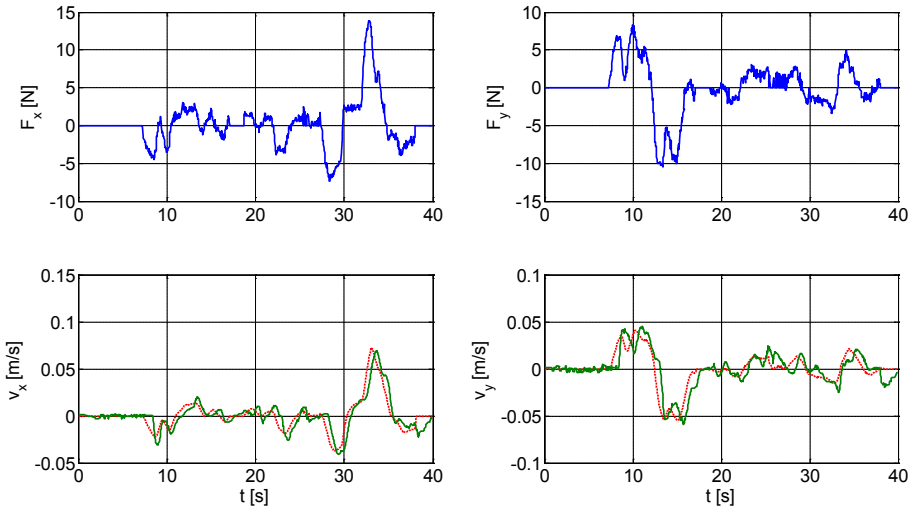

Figure 5-11. The admittance control in the task space reported for the two meaningful Cartesian components *x* (left) and *y* (right) that span the table surface. (*Top*) The external force imposed by the human teacher during coaching is perceived by the embarked F/T sensor and projected with a model-based approach at the end-effector level. (*Bottom*) Based on the external force a velocity reference is generated in the task space (dash red) and then tracked by the Cartesian controller (green).

A careful reader would have probably realized how in step 3 the three components of the external wrench referring to the momentum at the end-effector so as the three rotational components of the commanded velocity in the operational space are discarded

not being considered in the computation. This was done on purpose for sake of a first easy implementation, pondering also the fact that the orientation errors at the end-effector due to uncertainties in the kinematic model are much lower than corresponding errors in positions.

By iterating steps from 1 to 5 many times and with many objects of approximately the same size the goal of exploring the two-dimensional space of the table can be achieved and the map can be populated with the learnt offsets in a way that reasonable covers a wide range of possible objects locations.

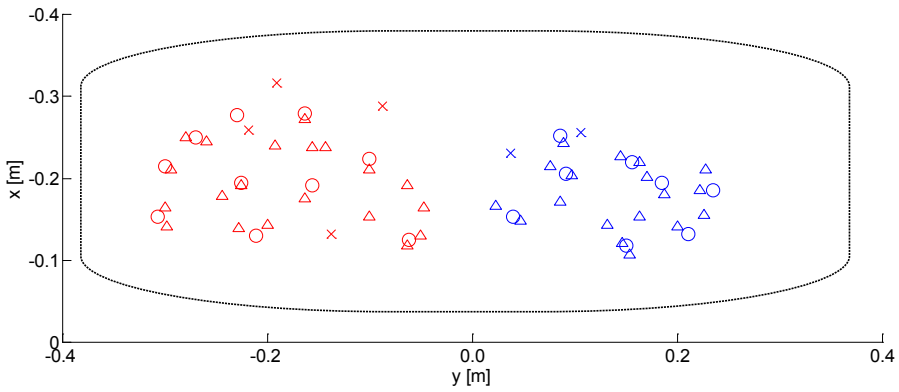

Figure 5-12. A figurative representation of the flat table (top view) is given where nodes used to build the internal offsets map through demonstrations are reported with the circled symbol (red for the left hand, blue for the right hand). Successful grasp events are then depicted with the triangle, whereas failure events are shown with the cross.

Finally, during nominal operation, when a novel target position is identified by *Spikenet*™, it is designed to employ a k -nearest neighbors algorithm that simply averages the offsets contained within the internal map on the basis of the distances of the current input from the k nearest nodes where the teaching-by-demonstrations was performed, in order to retrieve a prediction of the Cartesian offset to apply to the target. Figure 5-12 conveniently resumes this online procedure.

Discussion

Concluding this section, it is worth pointing out how the envisaged learning-by-demonstrations mechanism can adequately assist the reaching behavior with a reasonably high probability of success just by resorting to a very simple 2D map that even does not take into account any kind of specialization with respect to the particular object under attention, merely acting as a pure feed-forward term to compensate deviations from the model. Possible future improvements would be thus to diversify maps for reaching offsets and bias them with top-down stimuli collected by the visual perception and/or the tactile feedback of known and novel objects. Leveling up, learning maps for more complex affordances should also become attainable in a similar fashion.

However, this preliminary experiment served mainly to demonstrate the efficiency and the robustness of the "blending" principle that sees in this context the model-based Cartesian controller and the very basic machine learning technique as both fundamental ingredients to deal with such demanding task. On one hand, the Cartesian controller delivers the required grade of speed performance, movement accuracy and flexibility of the architecture that a pure state-of-the-art learning method would find indisputably burdensome to realize, lacking of the same amount of *a priori* knowledge; on the other hand, the adaption to the unmodeled perturbations is ensured by the (elementary) learning tool that counterbalances the drawback of a model when it has to deal with unstructured environment.

5.5 Learning Eye-Hand Coordination Through Motion[29]

As previously stated in the introduction to this chapter, the task refers to the aim of learning the offsets required to compensate the errors the robot naturally does when reaches positions in the Cartesian space due to uncertainties in the model. The goal here is therefore to design a procedure that allows enforcing a certain level of eye-hand

[29] So as for the study on *motionCUT* the results presented for the eye-hand coordination are the outcome of a profitable teamwork conducted with Carlo Ciliberto.

coordination such that, whenever the robot reliably localizes the target in both the cameras mounted in the eyes, it is able to extract the information needed to guide the hand towards it. In this respect, the role of stereo vision has received an increasing attention in literature as reported in (Kagami, et al. 2003) and (Gaskett and Cheng 2003).

Thus, it intuitively suffices to let somehow the robot build an internal mapping capable of linking the visual position of the hand within the image planes and the corresponding position in the operational space. Remarkably, the method is still not extended to the case of the hand orientations to keep the approach initially manageable and mostly because, unlike hand positions, the hand postures need pretty complex visual inference algorithms to be robustly acquired.

The first duty is to define suitable input and output spaces within the given framework.

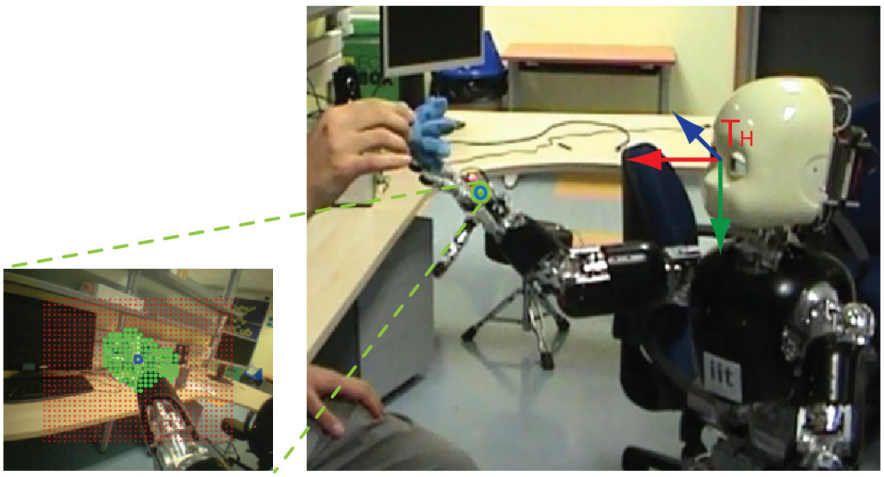

Figure 5-13. The framework of learning reaching task. The head centered frame $< T_H >$ is visible.

The Output Space

Examining the output space, one possibility suggested in literature and widely employed by studies in developmental robotics aims at coding the limb configuration

achieving the task in terms of its joints encoders. On one side, it undoubtedly evokes the least structured set of variables to describe a point in the output space, in the sense that it does not resort to any theoretical description axiomatically plugged into the architecture, as for instance the DH-based forward kinematic law[30]. On the other side, though, it exhibits two major practical shortcomings: (1) it exposes the system to the ill-posed inverse kinematics problem that entails multiple output configurations for a single input point so that it may frequently happen that points close in norm in the input space will correspond to configurations dramatically distant in the output domain, causing the learning to strive and eventually fail; (2) further, depending on the number of degrees of freedom used for reaching, the dimensionality of the output space can become inherently high (e.g. up to 10 if torso is employed) and, as consequence, the convergence time of the learning algorithms can increase exponentially.

Therefore, to cope with the two aforementioned weaknesses, it has been decided to rely on the three positional coordinates of the end-effector expressed in the Cartesian space to span the whole output domain. To achieve that, the forward kinematic map provided by the *iKin* library has to be necessarily exploited, contrasting thereby with the lessons somehow currently accepted in developmental approaches (Metta, Sandini and Konczak 1999) that rather tend to reject the *a priori* knowledge of the forward law in favor of its online estimation (Nori, Natale, et al. 2007). However, this model-based "backbone" layer addresses again the need to comply with stringent reliability and performance criteria, guaranteeing an intrinsically more robust design from an engineering standpoint and ensuring at the same time the possibility to perform learning at higher stage.

The Input Space

Once the output space is given, defining the input space turns to be straightforward. In fact, in order to uniquely relate the position of the end-effector in the stereo camera

[30] This necessity is motivated by the inspiration that developmental robotics dominantly inherits from the physiological findings in primates and humans.

planes (i.e. $\left(u_l, v_l\right)$ for left camera and $\left(u_r, v_r\right)$ for right camera) with its position in the Cartesian domain it is necessary and sufficient to augment the visual information with the current robot configuration including all the joints variables that serve to describe how the camera planes are oriented with respect to the frame adopted as reference. This detail clearly highlights the importance of selecting a suitable base frame that brings about the lowest number of independent variables; thus, the most obvious preference for the conventional fixed frame attached to the robot's waist – namely $<T_0>$ – is probably not the recommended solution, since it comprises the entire set of joints of the torso (3 joints), the head (3 joints) and the eyes parts (3 joints), that corresponds to an input space of dimension 13, where 4 components out of 13, as said, are due to the visual cues. Conversely, it turns out beneficial considering in lieu of $<T_0>$ the frame $<T_H>$ attached to the head at the middle point of the baseline connecting the two eyes (Figure 5-13). The primary justification is that taking $<T_H>$ as reference allows being concerned with only the actual values of the eyes encoders (i.e. tilt, pan, vergence) as meaningful quantities that determine the orientation of both cameras in $<T_H>$, boiling down the overall number of independent input variables from 13 to 7. Besides, it also holds that generating in $<T_H>$ random 3D points to be probed by the hand grants a boost to the exploration because the learning process is speeded up compared to the case where the target points are given in $<T_0>$. This expedient substantially reduces the chance that novel data will lie in regions of the input space that were already covered.

To sum up, having defined the input and the output space, the map M to be learnt is the following:

$$\left(x, y, z\right)_H = M\left(u_l, v_l, u_r, v_r, \mathrm{T}, V_s, V_g\right), \qquad (5.8)$$

where $\left(u_l, v_l, u_r, v_r\right) \in \mathbb{R}^4$ represent the visual input gathering the position of the hand within the stereo cameras, whereas $\left(\mathrm{T}, V_s, V_g\right) \in \mathbb{R}^3$ accounts for the proprioceptive part of the input designating the tilt, the pan and the vergence of the eyes; finally, $\left(x, y, z\right) \in \mathbb{R}^3$ is the Cartesian position of the hand we want to learn as expressed in the

head centered frame.

The Method

With these premises, the approach we applied for learning the eye-hand coordination map consists of two distinct phases: (1) one exploration stage that is devoted to the data acquisition, (2) a second off-line procedure that employs a standard Levenberg-Marquardt algorithm (Hagan and Menhaj 1999) to train a feed-forward neural network.

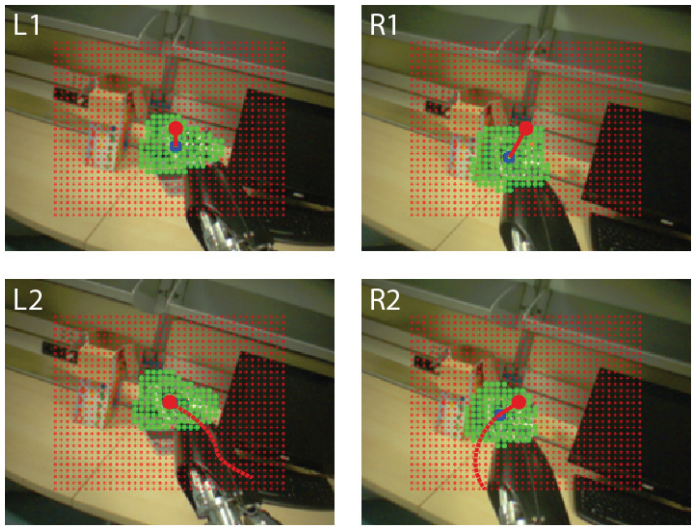

Figure 5-14. The data acquisition stage. L1 and R1 pictures report a typical acquisition of the same scene performed by *motionCUT* on the left and the right camera respectively: the actual position errors driving the Gaze controller and computed between the current hand centroids (blue circles) and the images centers are shown with red lines. L2 and R2 contain the epoch plot elucidating the history of input points used then in the off-line learning.

Bearing in mind that we avoid relying on markers for the hand detection, the data acquisition makes centrally use of the *motionCUT* device by virtue of its ability to be robust against the disruption of the optical flow caused by the egomotion which pervasively floods the scene since the robot is instructed to move its head continuously while tracking the hand.

Hence, for each epoch of the logging phase, a 3D point is randomly picked up in the head centered frame $< T_H >$ and then used as target to command a reaching with the Cartesian controller[31]. While reaching, the wrist is steered constantly back and forth in order to amplify the motion cue induced in the optical flow. Once the robot is in position with its arm, it keeps waving the hand until the gaze has approached it.

Meantime, as sketched in Figure 5-14, the induced independent motion of the hand is captured by the *motionCUT* modules that process the incoming image streams acquired through the cameras to then transmit the identified blobs centroids to the Gaze controller. The robot gaze is thus handled in a stereo closed-loop fashion as described in Chapter 3 in order to converge to the state where the projections of the end-effector centroid are situated in the close proximity of the centers of the two image planes.

Throughout the complete trajectory traced by the hand in the operational space as well as by the blobs centroids in the images, it is typically possible to collect online for each epoch about a hundred of data couples $\left(I^i, O^i \right)$, where $I^i = \left(u_l, v_l, u_r, v_r, \mathrm{T}, V_s, V_g \right)^i$ is the input and $O^i = \left(x, y, z \right)^i$ is the output of the map at recording instant i.

The training phase is carried out off-line by mean of MATLAB™ Neural Network Toolbox and runs on a feed-forward neural network with 7 nodes in the linear input layer, 50 nodes for the hidden layer implemented with the ordinary hyperbolic tangent function and 3 nodes in the linear output layer: an overall number of 15000 samples of $\left(I, O \right)$ pairs has been employed for training and validation, whereas 5000 samples have been used for test.

As reported in Figure 5-15, the neural network provides a very attractive estimation of the real underlying map. Notably, as expected, the z component results to be the most affected by noise since it accounts principally for the distance of the hand from the head, a value that is not directly elicited from the visual signals but rather implicitly deduced by the disparity between the two eyes views. The inspection of the very small mean and the

[31] The Cartesian controller works by default with target given in $< T_0 >$: anyways, knowing the kinematic relations among limbs the conversion from $< T_H >$ to $< T_0 >$ is immediate.

standard deviation error in Table 5-1 supports once more the quality of the net prediction.

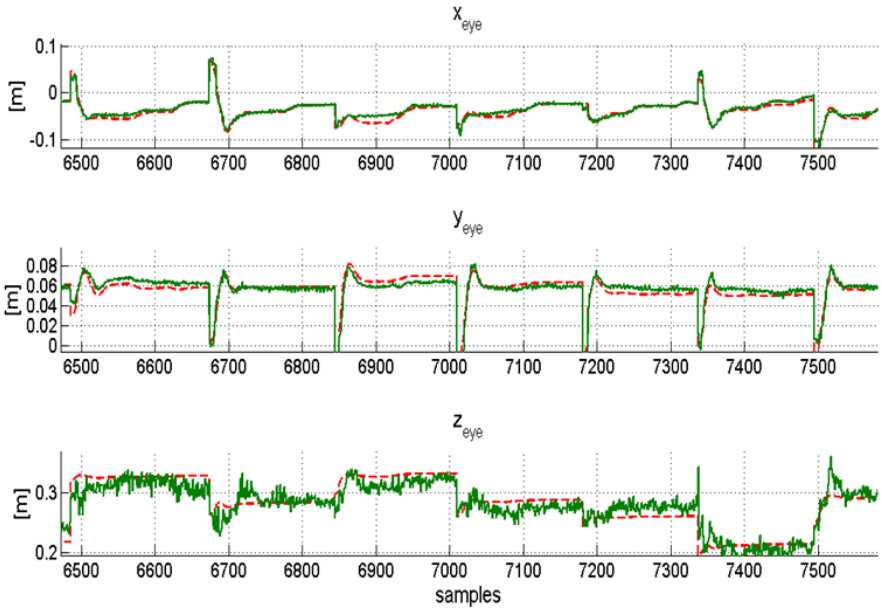

Figure 5-15. The desired target (dashed red) and the corresponding outputs of the neural network (green) for the three Cartesian coordinates in the head centered frame $< T_H >$.

	μ [m]	σ [m]
x	-0.00031	0.0055
y	0.00034	0.0056
z	0.00084	0.0194

Table 5-1. The mean and standard deviation error for the three Cartesian components in the head centered frame.

To obtain a rough measure of the goodness of the neural map from a more theoretical point of view, it might be convenient to compare the results of the learnt map with the response of a parametric model fitted with the same recorded data used for the training of the network.

To this end, let us consider the simplest conditions where the robot is fixating straight

at the hand at the conclusion of each epoch, over increasing distances from the head; thus, the visual stimuli $\left(u_{l,r}, v_{l,r}\right)$ the tilt T together with pan V_s of the eyes remain always constant, and specifically $u_{l,r} = 160, v_{l,r} = 120$ for a 320×240 image size and $\mathrm{T} = V_s = 0$ degrees. Only the vergence V_g varies according to the distance z_{eye} from the target.

Mathematically, the following model relating the two latter quantities holds at first order of approximation:

$$z_{\mathrm{eye}} = \frac{b}{2} \cdot \cot\left(\frac{V_g}{2}\right), \qquad (5.9)$$

where b represents the length of the baseline between the two eyes. In addition to the baseline length, the model can be augmented with a second parameter accounting for an unknown offset and then tuned on the basis of the logged data.

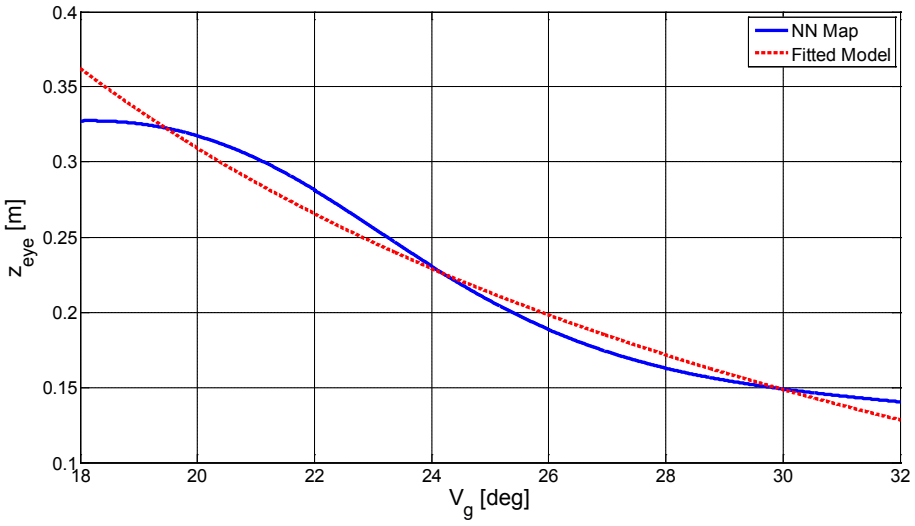

Figure 5-16. Comparison between the responses of the physical model (dashed red) and the prediction of the neural map (blue) when the target is in fovea and the robot straightly gazes at it.

Figure 5-16 shows the good resulting correspondence between the physical model and the prediction of the network within the range where the net was trained.

Experiments that take advantage of such map recruiting a visuo-motor reaching functionality are still ongoing but already provide promising yet preliminary results. The *iCub* indeed is able to determine the 3D position of any object presented to it and waved in order to generate motion that in turn is captured by *motionCUT*: the robust motion cues permit to discard the visual features that refer to the appearance of the object (e.g. colors, corners, size, etc) and that are generally difficult to be detected in different environmental conditions; as result, it happens that the robot can touch the target in the majority of the cases.

As concluding remark, it is relevant to outline here that an upcoming activity has been planned with the purpose to replace the off-line training phase with a fully online version that resorts to random features as in (Gijsberts and Metta 2011) and will eventually make the robot learn the eye-hand coordination completely autonomously.

CHAPTER 6

TRAJECTORY ENCODING: TOWARDS HIGH-LEVEL FEED-FORWARD CONTROL

6.1 Introduction

The performances of the Cartesian controller designed in Chapter 2 are somehow limited in terms of reactivity by the fact that the regulator responsible for the generation of velocity commands has to be placed at high-level in the system architecture (i.e. in the cluster or aboard the robot hub) and needs to run in a closed loop fashion reading the actual joints values in order to compensate for external disturbances and discrepancies in the kinematic model. Therefore the control loop is handled in near real-time (~ 10 ms) and to avoid unwanted overshoots while converging to the target, the controller's parameter T specifying the point-to-point movement time cannot be set under a minimum threshold of approximately 0.5 s (this values is affected by some variability depending on the task); this fact that induces an obvious restriction in how fast movements can be reproduced.

Nonetheless the limiting factor of near real-time conditions is not easily solvable by resorting merely to the Cartesian controller as it is designed due to two basic reasons: (1) it is impracticable to make the controller run aboard the low-level boards where the hard real-time is available because the DSPs are not powerful enough to accomplish the required kinematic computations and moreover the embedded boards are local to the motors they control, whereas a central unit managing the motion of the whole structure as well as the motion of the end-effector is a mandatory prerequisite; (2) the velocity control paradigm adopted by the central regulator imposes itself the constraint on the overall performance and cannot be superseded in favor of a pure position control method, since

the position-based commands are executed at low-level as minimum-jerk trajectories that do not suffice to describe all the possible joint profiles required for human-like movements in the operational space.

A remedy to address the need for increasing controller's performance is to envisage and develop a method which allows reproducing a much richer set of joint reference trajectories (not only minimum-jerk), by representing them within appropriate functional bases in order to eventually make the low-level position control usable; the final goal is thus to set the high-level controllers free from the constraint of closed loop by exploiting the fast reactivity of a feed-forward model whose inaccuracies can be in turn corrected by local DPSs controllers.

An inquiry has been conducted on techniques available from literature that enable to suitably code an arbitrary profile as a collection of coefficients. The investigation and assessment of these techniques in terms of number of required encoding coefficients computed for specific bio-mimetic trajectories as well as the quality of approximation is the objective of this chapter as detailed in the following.

6.2 Identification of Functions Basis for Trajectory Encoding

In a framework where the function to be coded $F(t)$ is sampled at discrete time instants t_i it is worth introducing the measure E_{rel} in order to estimate how good the final encoded representation $\tilde{F}(t)$ describes the original profile:

$$E_{\text{rel}} = \frac{\sum_{i=1}^{M}\left(F(t_i) - \tilde{F}(t_i)\right)^2}{\sum_{i=1}^{M}F(t_i)^2}. \qquad (6.1)$$

This measure will be usefully employed in section 6.3 to carry out comparison tests over the candidate methods that have been identified and reported hereafter.

1. Cubic Splines

This is a well-established approach easy to be implemented which describes a generic curve in terms of piecewise third-order polynomials.

If the entire time period T of the trajectory is partioned in N intervals, then for each interval $I_{k=1,\ldots,N}$ the approximation is locally given by:

$$\begin{cases} f_k\left(t\right) = a_k t^3 + b_k t^2 + c_k t + d_k, & t \in I_k \\ f_k\left(t\right) = 0, & t \notin I_k \end{cases}. \quad (6.2)$$

The unified encoded trajectory is thus represented by:

$$\tilde{F}_N\left(t\right) = \bigcup_{k=1}^{N} f_k\left(t\right). \quad (6.3)$$

The coefficients $\left(a_k, b_k, c_k, d_k\right)$ are determined so that the values of piecewise polynomials f_k match exactly the samples of trajectories (*knots*) at the boundaries between intervals and also to establish a condition of smoothness on the final approximation by preserving the continuity of its first derivative along the whole period $\left[0, T\right]$. Moreover, it is possible to specify that the first derivative is zero in $t = 0$ and $t = T$ which are reasonable starting and ending conditions since the movements we aim to code are supposed to start from and come to the rest state.

Clearly the number of resulting coefficients required to code any generic profile given the number N of intervals is equal to $4N$. Conversely, given a particular merit E_{rel}^{*} that the representation $\tilde{F}\left(t\right)$ is requested to achieve, an iterative process can take place starting from low values of N that entail course representations and then increasing step by step the number of intervals up to the point where the resulting E_{rel} attains the target.

The same iterative procedure applies also to the other methods described in the following.

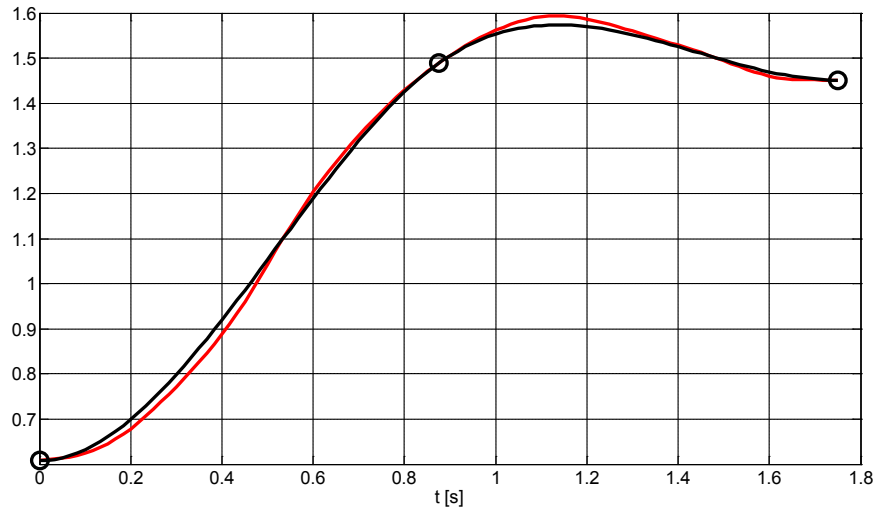

Figure 6-1. Example of encoding an arbitrary trajectory (red) with cubic-spline approximation (black). The two intervals used to create the piecewise third order polynomials, corresponding to 8 coefficients in total, are visible in the picture. E_{rel} is $6.1 \cdot 10^{-5}$.

2. Bio-Mimetic Nonlinear Differential Systems

The properties of two second-order nonlinear coupled systems (Ijspeert, Nakanishi and Schaal, Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots 2002) of reproducing a human-like trajectory are here exploited. The proposed control policies (*CPs*) are capable of learning any arbitrary demonstration through a set of N parametric Gaussians trained with a locally weighted regression technique.

In particular, a control policy is defined by the following $\left(z, y\right)$ dynamics which specifies the attractor landscape of the policy for a trajectory $y-$ that is the encoded representation $\tilde{F}$ – towards a goal $g = F\left(T\right)$:

$$\begin{cases} \dot{z} = \alpha_z \left(\beta_z \left(g - y \right) - z \right) \\ \dot{y} = z + \dfrac{\sum_{i=1}^{N} \Psi_i w_i}{\sum_{i=1}^{N} \Psi_i} \cdot v \end{cases} . \tag{6.4}$$

This is basically a simple second-order system with the exception that its velocity is modified by a nonlinear term which depends on internal states. These two internal states (v, x) have in turn the following second-order linear dynamics[32]:

$$\begin{cases} \dot{v} = \alpha_v \left(\beta_v \left(g - x \right) - v \right) \\ \dot{x} = v \end{cases} . \tag{6.5}$$

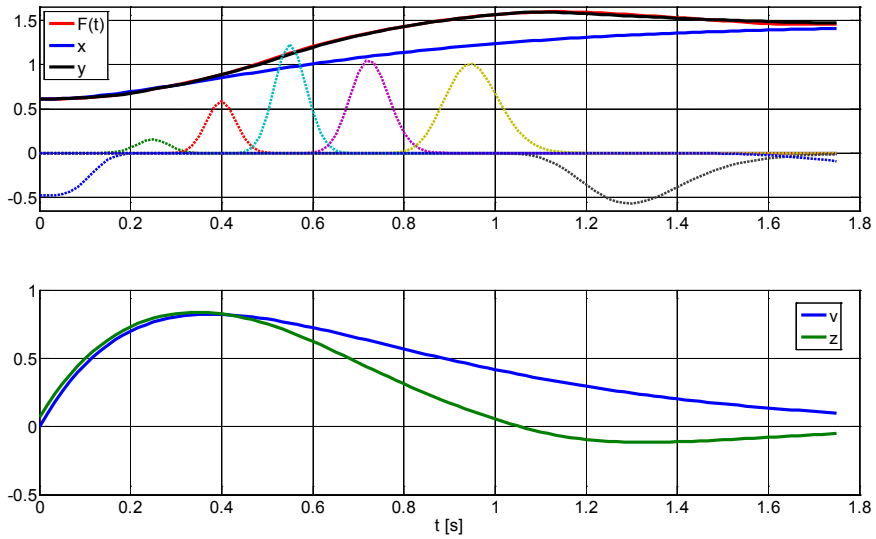

Figure 6-2. An arbitrary trajectory (red) is encoded resorting to Gaussian kernels. The encoded representation is depicted in black in the upper diagram; further, the evolution of the coupled dynamical systems is shown in both graphs. A number of 8 kernels $w_i \cdot \Psi_i$ shown in dashed lines in the top graph are employed for a resulting E_{rel} of $3 \cdot 10^{-5}$.

The system is further determined by the positive constants $\alpha_v, \alpha_z, \beta_v,$ and β_z and by a set of N Gaussian kernel functions Ψ_i

[32] It is straightforward to recognize that the quantities z and v are regulated by a *VITE*-like dynamic.

$$\Psi_i = \exp\left(-\frac{1}{2\sigma_i^2}\left(\overline{x} - c_i\right)^2\right), \tag{6.6}$$

where $\overline{x} = \left(x - x_0\right)/\left(g - x_0\right)$ and x_0 is the value of x at the beginning of the trajectory, i.e. $F\left(0\right)$. As anticipated, the attractor landscape of the policy can be adjusted in order to represent the path $F\left(t\right)$ to be coded by learning the parameters w_i using locally weighted regression (Schaal and Atkenson 1998). Note that this dynamical system has a unique equilibrium point at $\left(z, y, v, x\right) = \left(0, g, 0, g\right)$, and has been proven to be stable and robust against perturbations.

Figure 6-2 shows how the Gaussian kernels perform function encoding; interestingly for a preliminary comparison, the function to be coded is the same as Figure 6-1.

3. Jacobi polynomials

The trajectories are represented by a uniquely-defined sequence of expansion coefficients within a proper basis of square-integrable function space made of the so-called Jacobi polynomials (Biess, Nagurka and Flash 2006). Thanks to the orthogonality property of this special basis the coefficients computation turn out to be straightforward and do not need any training stage, unlike the Gaussian kernels approach.

In formulas, the expression for the representation $\tilde{F}\left(t\right)$ is:

$$\tilde{F}\left(\tau\right) = \sum_{k=0}^{2m-1} p_k \tau^k + \sum_{k=0}^{N} c_k \varphi_k^{(m)}\left(\tau\right), \tag{6.7}$$

where, being $\tau = t/T$ the normalized time, the first term of the sum is a polynomial used to verify the $2\left(m-1\right)$ inhomogeneous conditions at the boundaries of the interval $\left[0, T\right]$ on $F\left(t\right)$ and its $\left(m-1\right)$ derivatives – and we employ herein the minimum-jerk polynomial – whereas the N Jacobi polynomials $\varphi_k^{(m)}$ satisfy the homogeneous boundary conditions.

It holds:

$$\phi_k^{(m)}(\tau) = \frac{2^{2m}\tau^m(1-\tau)^m}{\left(h_k^{(2m,2m)}\right)^{\frac{1}{2}}} P_k^{(2m,2m)}(2\tau-1),$$

$$P_k^{(\alpha,\beta)}(x) = \frac{1}{2^k}\sum_{i=0}^{k}\binom{k+\alpha}{i}\binom{k+\beta}{k-i}(x-1)^{k-i}(x+1)^i, \quad (6.8)$$

$$h_k^{(\alpha,\beta)} = \frac{2^{\alpha+\beta+1}}{2k+\alpha+\beta+1}\frac{\Gamma(k+\alpha+1)\Gamma(k+\beta+1)}{k!\Gamma(k+\alpha+\beta+1)},$$

with Gamma function Γ.

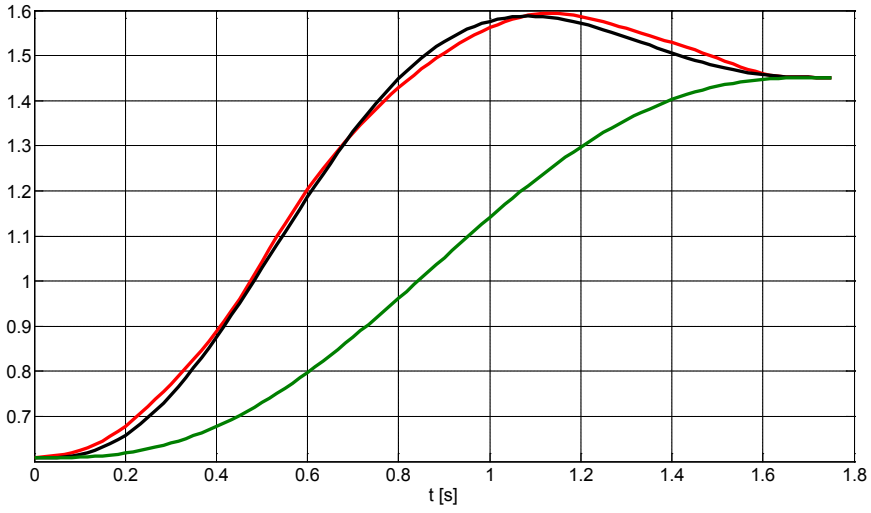

Figure 6-3. Encoding of arbitrary function (red) achieved through Jacobi polynomials based on a minimum-jerk realization (green) for satisfying inhomogeneous boundary conditions; the final encoded representation is displayed in black. Two expansion coefficients are used for a merit E_{rel} of $7.4\cdot10^{-5}$.

The choice of the minimum-jerk polynomial to express the first term of (6.7) naturally

inherits from the fact that it is the standard coding method used at low-level by DSPs.

As it is evident, the computation of $\varphi_k^{(m)}$ entails a large number of operations; on the other hand, the N expansion coefficients of the representation are simply provided by integration:

$$c_k = 2\int_0^1 \left(F(\tau) - \sum_{i=0}^{2m-1} p_i \tau^i \right) \cdot \phi_k^{(m)}(\tau)\, d\tau. \qquad (6.9)$$

To obtain the(6.9), it has been exploited the orthogonality property of the function basis, that is:

$$\int_0^1 \varphi_i^{(m)}(\tau) \cdot \varphi_j^{(m)}(\tau)\, d\tau = \frac{1}{2}\delta_{i,j}, \qquad (6.10)$$

being $\delta_{i,j}$ the Kronecker-delta.

4. Multiresolution approximation with Wavelets

Multiresolution (Mallat 1989) is a method to decompose any trajectory as a sum of an approximation and a detail signal at a given resolution by convolving the original curve with a basis built of special functions known as scaling functions or father wavelets which satisfy some useful properties such as orthogonality and similarity. Multiresolution can be seen as an extension of Fourier Transform since the normal sine and cosine kernel functions are replaced with a highly spatial-localized set of wavelets, resulting in a more effective representation in terms of number of parameters.

Essentially the Multiresolution approach makes use of linear operators A_{2^j} that when applied to the function we want to encode $F(t)$ – belonging to the vector space of square-integrable one-dimensional functions $\mathbf{L}^2(\mathbb{R})$ – returns its approximation at resolution 2^j:

$$A_{2^j}\left[F(t)\right] = 2^{-j}\sum_{n=-\infty}^{+\infty}\left\langle F(u) * \phi_{2^j}\left(u - 2^{-j}n\right)\right\rangle \cdot \phi_{2^j}\left(t - 2^{-j}n\right), \quad (6.11)$$

being

$$\left\langle F(u) * \phi_{2^j}\left(u - 2^{-j}n\right)\right\rangle = \int_{-\infty}^{+\infty} F(u)\phi_{2^j}\left(u - 2^{-j}n\right)du \quad (6.12)$$

the inner product within the space $\mathbf{L}^2(\mathbb{R})$ evaluated in the point $2^{-j}n$, that can be interpreted as an operation of low-pass filtering followed by a uniform sampling at rate 2^{-j}. Therefore, as the expansion based on the Jacobi polynomials, the computation of wavelet coefficients is accomplished easily by integrating the original profile with copies of a single father wavelet $\phi(t)$ properly scaled and translated in time. The father wavelet is chosen according to the specific application: in our case the Daubechies type 4 has been selected whose profile is shown in Figure 6-4.

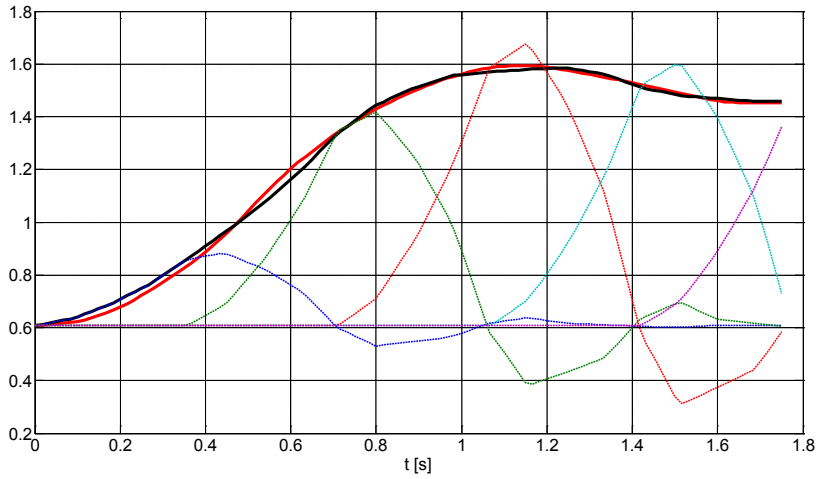

Figure 6-4. Representation of a function (red) in the wavelet vector space. In dashed lines the father wavelets properly scaled and translated to compose the final approximation (black) are visible. Five coefficients have been employed with a measure E_{rel} of $8.8\cdot10^{-5}$.

Provided the number of expansion coefficients N to be used by the representation $\tilde{F}(t) = A_{2^j}[F(t)]$, the resolution is given by $2^j = N$.

6.3 Function Basis Assessment

In order to assess which among the aforementioned strategies turns to be the most suitable one in a real environment conditions, an experimental setup has been put in place based on the *Vicon MX Motion Capture System* (Figure 6-5) in order to acquire human trajectories during reaching tasks to be encoded using the methods under analysis.

The recorded markers' position profiles comprise both outward movements, that steer the arm far from the body, and inward movements, that drive the arm back to the rest position.

Figure 6-5. A picture of the *Vicon* setup available at RBCS Department used for the experiments.

Ten markers to record human reaching movements have been attached to the arm of the subject as in Figure 6-6, whilst a snapshot of the acquired displacements is show in Figure 6-7and Figure 6-8.

Figure 6-6. Location of the markers on the subject to record the arm reaching movements.

Therefore three trials of the same reaching movement have been logged.

The acquired human wrist position in the task space x_{wrist} is then suitably scaled and projected within the *iCub* workspace in order to be used as the reference target position of the robot's wrist in the following minimization problem:

$$\begin{aligned} & q = \arg\min_{q} \left(\left\| x_{\text{wrist}} - K_{\text{wrist}}(q) \right\|^2 + \left\| w \cdot (q_d - q) \right\|^2 \right); \\ & \text{with } q = \left(q_1, q_2, q_3, q_4\right)^T, w = \left(0,0,0,1\right)^T, \end{aligned} \tag{6.13}$$

where q_1, q_2 and q_3 are the three joint angles of the *iCub* shoulder, q_4 is the *iCub* elbow joint angle, q_d is the acquired elbow angle of the human trajectory and finally K_{wrist} is the forward kinematic map of the *iCub* wrist.

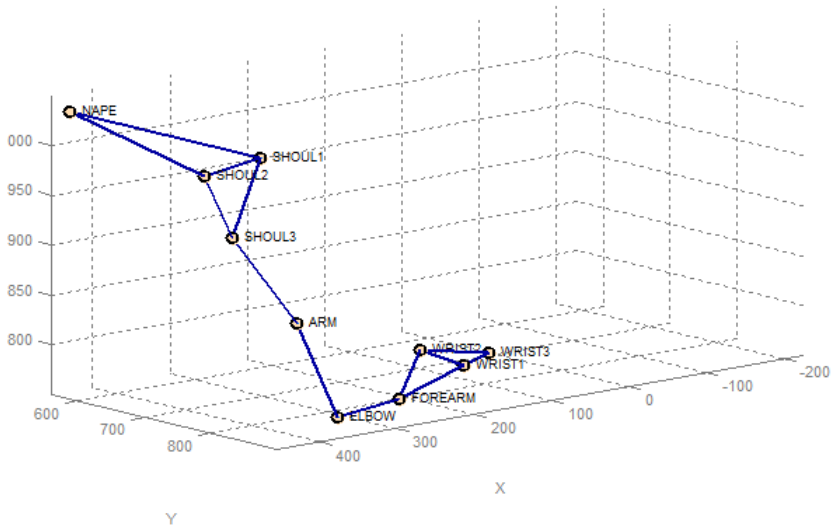

Figure 6-7. A sketch representing markers' location at the onset of the trajectory as acquired through the *Vicon* system.

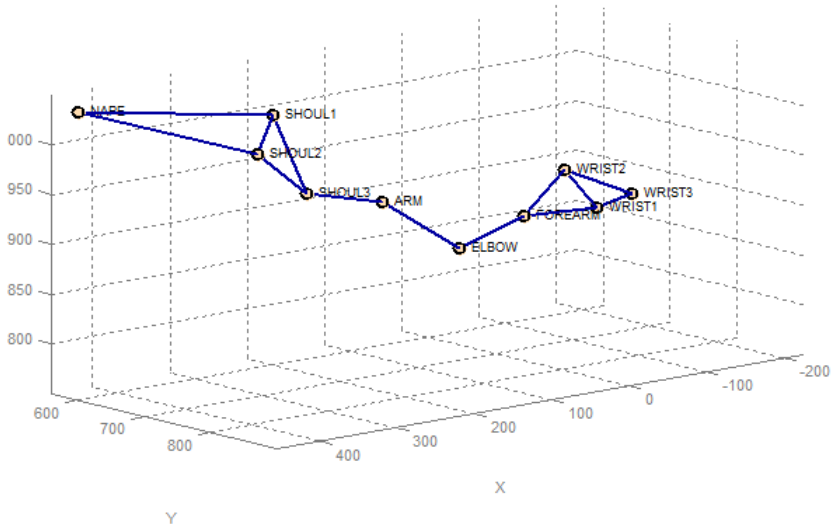

Figure 6-8. The markers' location as recorded in the ending configuration of the reaching phase.

By solving (6.13) point by point with a nonlinear constrained optimizer the *iCub* joint angles trajectories are obtained, which can be finally encoded through the four algorithms whose results have been evaluated taking into account the following parameters:

1. Minimization of required encoding coefficients (the **basis rank**) in order to save low-level bus bandwidth.
2. Grade of **computational load** aboard DSP in terms of number of additions and products and more complex operations such as exponential (where present).
3. **Ability to classify** multiple trials $F_i\left(t\right)$ and $F_j\left(t\right)$ of the same reaching movement defined as the variance of the following quantity $n_{ij}, \forall i \neq j$:

$$n_{ij} = \frac{\left\| c_i - c_j \right\|}{\sqrt{\int_T \left(F_i\left(t\right) - F_j\left(t\right) \right)^2 dt}} \tag{6.14}$$

where c_i and c_j are expansion coefficients – in the corresponding base – of $F_i\left(t\right)$ and $F_j\left(t\right)$, respectively, whereas the denominator of (6.14) accounts for the distance of the realizations in $\mathbf{L}^2\left(\mathbb{R}\right)$ space and serves to be able to compare results stemming from different strategies.

The assessment index $\mathrm{var}\left(n_{ij}\right)$ is meant to convey somehow the capability of a specific function basis to reproduce profiles which are close in a given integrable measure – i.e. they do not change too much, such as the trials recorded in the reaching experiment – with the less variable set of coefficients. This interesting property could turn out to be beneficial in subsequent studies where such novel implementation of low-level motion generation could pave the way for applying machine learning algorithms that can be trained on a number of demonstrations and would be eventually capable of achieving a task just by combining basis functions properly.

Hereafter are reported all the graphs that resume the values of the introduced parameters assumed for each of the four joints involved in the inward and outward reaching movements. Figure 6-9, Figure 6-10, Figure 6-11, and Figure 6-12 give a view on how the different techniques behave in terms of their minimum basis ranks required to attain a value of E_{rel} lower than 10^{-4} together with the computational burden that the DSPs have to sustain per real-time instance to decode the trajectory; Figure 6-13 depicts the quantity $\mathrm{var}\left(n_{ij}\right)$ as it varies for each encoding algorithm on all the joints.

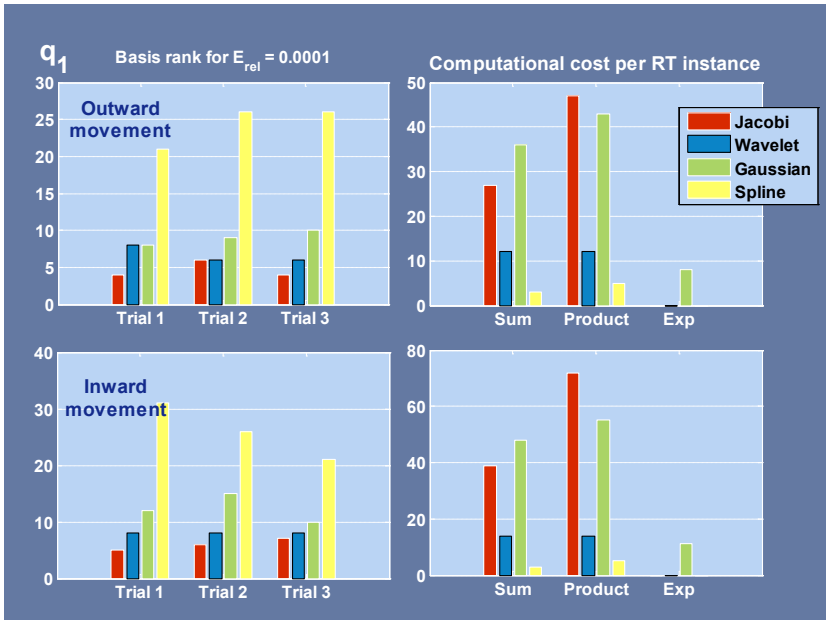

Figure 6-9. Behavior of basis ranks and computational costs for the outward and inward trials of joint q_1.

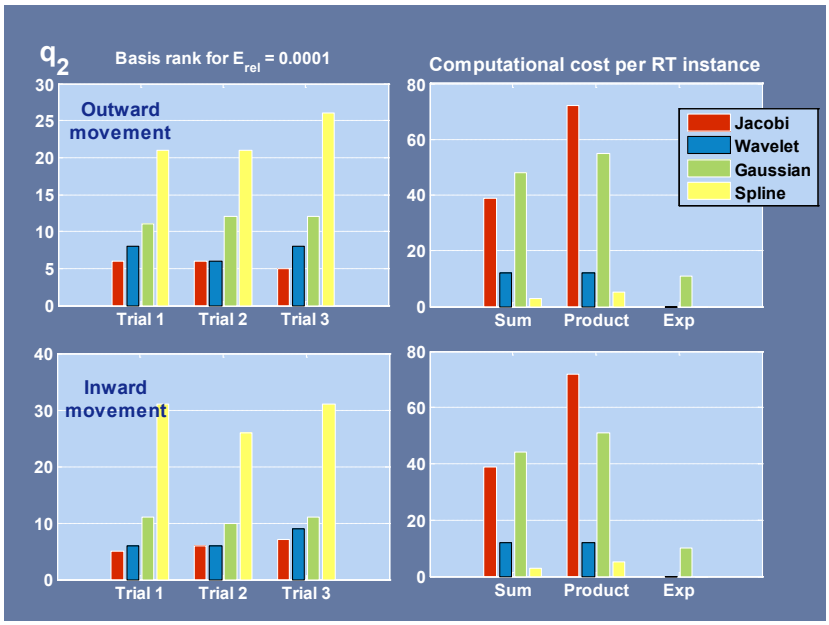

Figure 6-10. Behavior of basis ranks and computational costs for the outward and inward trials of joint q_2.

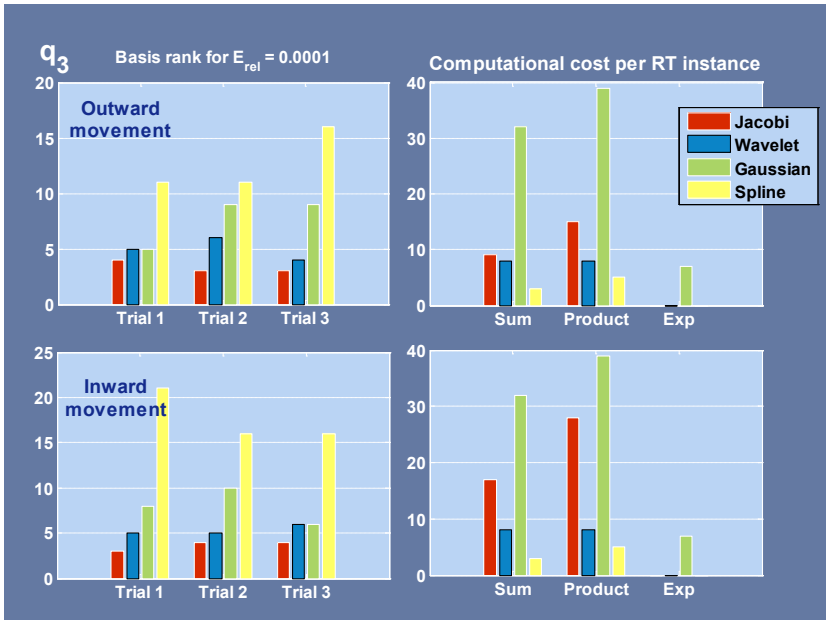

Figure 6-11. Behavior of basis ranks and computational costs for the outward and inward trials of joint q_3.

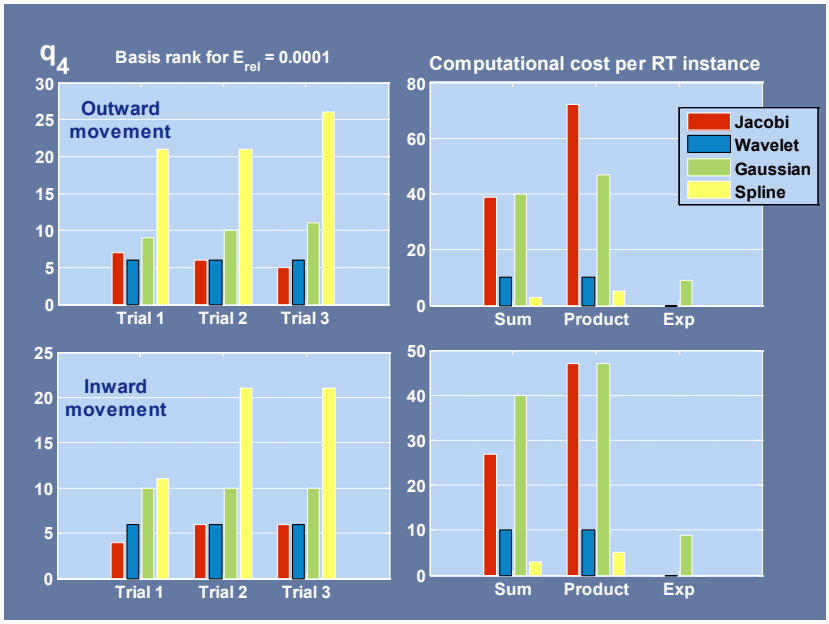

Figure 6-12. Behavior of basis ranks and computational costs for the outward and inward trials of joint q_4.

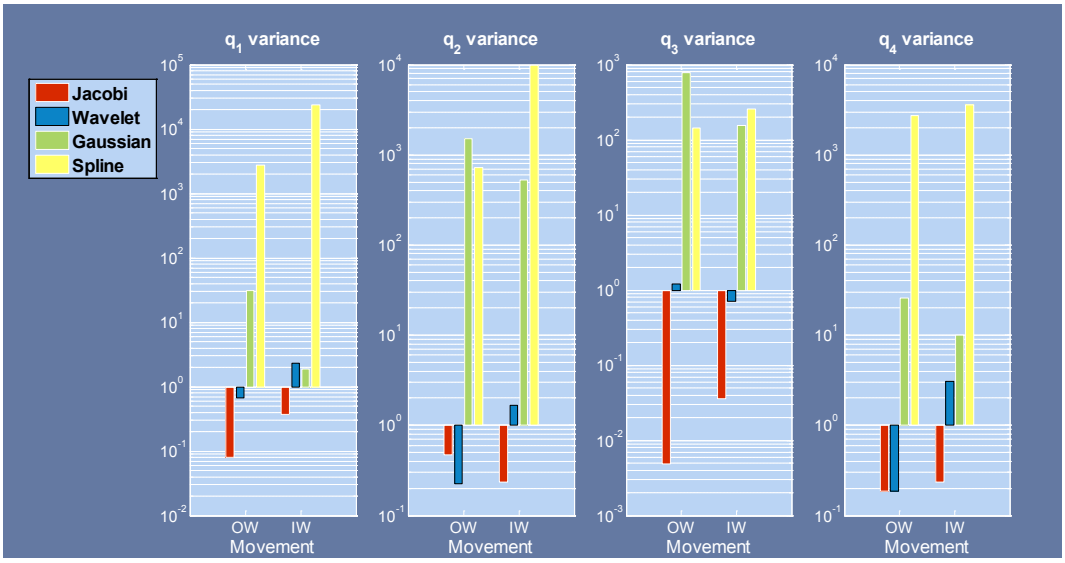

Figure 6-13. Behavior of the *var* $\left(n_{ij} \right)$ measure expressed in logarithmic scale.

From the results shown in the previous pictures it can be derived that the Multiresolution representation based on the wavelets expansion turn to be the most effective way of encoding joint trajectories generated by human-like reaching movements since the number of needed vector components is comparable to one of Jacobi expansion but with the substantial benefit that a much smaller computational effort is required per real-time instance aboard DSPs.

Concerning the trials classification ability, Jacobi polynomials clearly outperforms remaining approaches, but the wavelet-based strategy still remains appealing as its variance approaches the Jacobi baseline, being superior with respect the splines and Gaussian kernels.

Table 6-1 details a summary of the pros and cons for each evaluated strategy.

6.4 Model Development

As outcome of the evaluation stage it has been decided to develop the DSP firmware implementing the new wavelets-based trajectory generator exploiting the powerful feature

of *Real Time Workshop Embedded Coder* MATLAB toolbox to automatically produce C code from a *Simulink* model of the generator itself, which is surely easier to design and already to test under several working conditions.

A sketch of the model as it appears from within the designing environment is shown in Figure 6-14: the model is capable of generating four independent trajectories as foreseen by the control board, each consisting of the modules detailed hereafter.

Method	Benefits	Drawbacks
Splines	Fast coefficients identification; Light computational load.	4 parameters/interval needed; 5÷6 intervals for $E_{\text{rel}} < 10^{-4}$; Non-autonomous movement representation.
Gaussian kernels	No time indexing; Protection against over-fitting;	Exponentials aboard DSPs; Target g reached at infinite time; Critical tuning of parameters; Required derivative of $F(t)$; Regression is required.
Jacobi polynomials	Easy coefficients computation; Small number of coefficients; Increasing order N does not affect coefficients for $N-1$.	Heavy computational load aboard DSPs.
Wavelet	Easy coefficients computation; Small number of coefficients; Light computational load;	Final position is attained with non-null error (very small).

Table 6-1. Summary of pros and cons for the assessed encoding strategies.

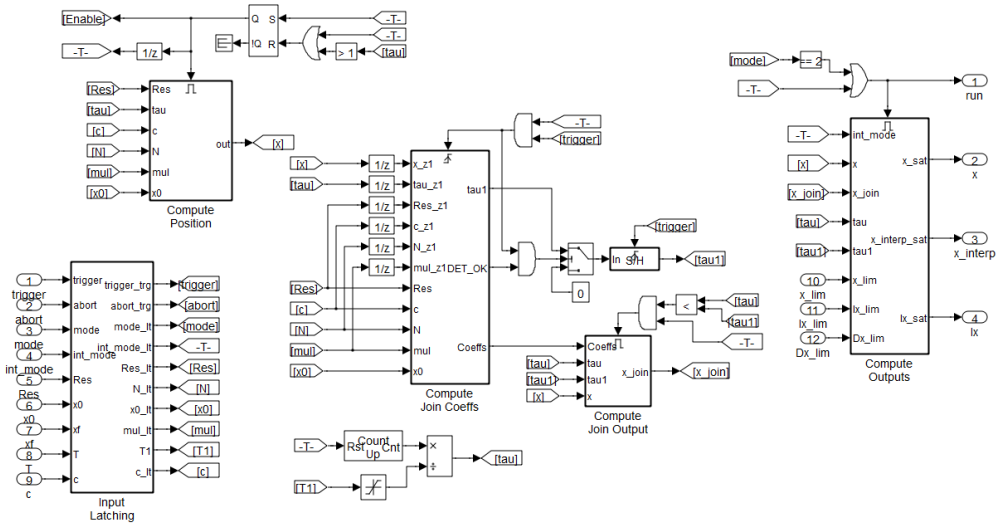

Figure 6-14. Sketch of the model for trajectory decoding aboard DSPs as designed within *Simulink* environment.

The input latching module

This module latches all the model inputs (coefficients, required resolution, joint angle final position, movement's duration, trigger and abort commands, and mode selectors). The purpose is to prevent the user from changing inputs while trajectory is being produced without having properly notified it through the trigger command.

The trajectory computing module

This module computes the actual trajectory based on the coefficients vector provided by the user just combining them in a proper entry point to a look-up table which interpolates the basic father wavelet function (Daubechies type 4 is used).

The smooth-joining module

This module is in charge of computing a smooth joining between the current trajectory

and a new one in case the latter is commanded before the former has ended. This smooth transition is needed to avoid discontinuities in velocity and it is implemented through a third order interpolating polynomial which is applied for a time duration depending on the difference between the slopes of the two curves at the intersection point in order to preserve the value of the first derivatives. On top of this joining strategy a further level to improve the smoothness is provided by applying the following formula:

$$\begin{aligned} x(t) &= \rho(t)\cdot p(t) + (1-\rho(t))\cdot x_2(t), \\ \rho(t) &= \left(\frac{\tau}{\tau_1} - 1\right)^2, \\ p(t) &= a_3 t^3 + a_2 t^2 + a_1 t + a_0 \quad \text{s.t.} \begin{cases} p(0) = x_1(0) \\ \dot{p}(0) = \dot{x}_1(0) \\ p(T_1) = x_2(0) \\ \dot{p}(T_1) = \dot{x}_2(0) \end{cases}, \end{aligned} \tag{6.15}$$

where $x(t)$ is the resulting joined trajectory, $x_1(t)$ is the still on-going commanded trajectory, $x_2(t)$ is the new commanded trajectory, $p(t)$ is the third order polynomial as specified by the first smoothing strategy and $\rho(t)$ is a time varying parameter which further shapes the output, depending on the overall time duration of the smooth transition τ_1, being τ the normalized time t/T, with T_1 equal to switching instant and T equal to the total movement time duration. Note that the expression for $x(t)$ is obviously valid only for $\tau < \tau_1$.

In Figure 6-15 the behavior of the model is shown for a condition in which a new trajectory is commanded (time instant $t_2 = 2\,\text{s}$) while the generation of an already commanded trajectory is still on-going. The effect of the trajectory harmonizer is evident (the red curve in the plot), being capable of preserving the first derivative at the new trigger instant and moreover to steer the final trajectory towards the newest one in a really short time.

The output module

This module is ultimately responsible for the generated output, providing to limit its rate within a given set of constraints as well as to produce the integral waveform of the trajectory according to the state of a mode selector, whenever this quantity is required, e.g. in velocity control mode.

Finally the C code has been straightforward produced by the toolbox properly configured in order to target the DSP mounted on the control board (Freescale 56F807). It shall be pointed out that in this fast-prototyping stage the powerful feature of automatic fixed-point code production, which is a huge time demanding issue for a hand-written code, has been extensively exploited.

The resulting code has been integrated successfully with a small effort within the pre-existing firmware and it is currently under test.

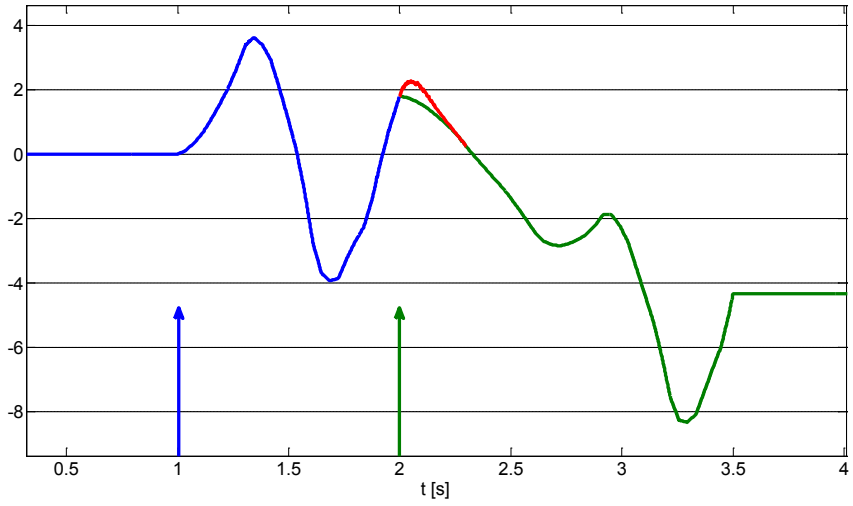

Figure 6-15. How the system harmonizes the first trajectory started at t=1 s in blue with the second trajectory in green started at t=2 s when the former has not finished yet: the result of the joining strategy is visible in red.

CONCLUSION AND FUTURE WORKS

The three years study resumed in this thesis work concerned the realization of two central components acting in the operational space of the robot in a bio-mimetic fashion: one devoted to the control of the arms for reaching tasks, a second to let the robot gazes at a point as provided for example by the attention system. We have seen how the main specifications that oriented the design were the robustness and the performance requirements of the response and the portability of the architecture to robotic platforms different from the *iCub*, along with its scalability with respect to the number of degrees of freedom. The choice of a combined structure composed of a nonlinear real-time optimizer coupled with a minimum-jerk controller has been demonstrated to outperform classical approaches resorting to gradient projected method and damped least-squares solutions of the inverse kinematic problem.

It has been then confirmed how having a solid layer for coping with the motor control turns to be significantly beneficial for the adaption to uncertainties that are not foreseen by the model: a paradigm that definitely suggests to exploit as much as possible the acquired knowledge of the system by virtue of the wide set of well-established "old fashion" policies available from literature guaranteeing the performance achievements, whereas the resort to machine learning techniques has to be pursued in order to compensate for the unknown quantities that naturally emerge from the interaction of the robot with the environment. Preliminary yet very appealing results of this approach that blends modeling with learning are reported for the cases of reaching offsets compensation and robot eye-hand markerless coordination. Thus, in the same direction some additional efforts would be required in the near future to improve the grade of errors compensation in visuo-haptic tasks. One meaningful example of such kind of research would be the autonomous learning of a stereo gaze map that allows building online a look-up table containing the 3D positions of the target once its projections on both the camera image

planes are given: providing an accurate feed-forward prediction of the learnt saccades on the basis of a feedback oriented exploration, the map will certainly permit to replace the stereo method described in Chapter 3 which exhibits slow converging time due to the need of acquiring the feedback signal. Furthermore, the learning of robot eye-hand coordination definitely requires to be carried out online in an autonomous fashion, by employing incremental algorithms such as random features or support vector machines for regression.

Regarding the wavelet-based trajectory encoding treated in the last chapter, the activity will continue with extensive tests of the production code aboard the real platform along with the development of a new communication protocol that would enable to deliver the encoding parameters from the high-level controllers to the firmware level coping with real-time aspects, regarding especially the synchronization issue that stems from the intrinsic nature of a distributed system. The most important benefit of this major overhaul of the motor control layer is the possibility to execute an arbitrarily complex movement in the Cartesian space relying on a completely feed-forward approach that turns to be much faster if compared to the closed-loop case since it is not constrained by the rate of the feedback readings. In fact, as explained in Chapter 6, the wavelet encoding allows delegating the trajectory execution to the low-level position control instead of resorting to the high-level velocity control as it is presently done. Besides, such novel implementation that describes a movement in terms of coefficients in a given base will also pave the way for applying machine learning algorithms that can be trained on a number of demonstrations and would be eventually capable of achieving a task just by combining wavelets functions properly.

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

Abend, W., E. Bizzi, and P. Morasso. "Human Arm Trajectory Formation." *Brain* 105 (1982): 331-348.

Bienenstock, E., and S. Geman. "Compositionality in Neural Systems." In *The Handbook of Brain Theory and Neural Networks*, by M. A. Arbib, 223-226. Cambridge: MIT Press, 1993.

Biess, A., M. Nagurka, and T. Flash. "Simulate Discrete and Rhythmic Multi-Joint Human Arm Movements by Optimization of Nonlinear Performance Indices." *Biological Cybernetics* 95 (2006): 31-53.

Bouguet, J. Y. *Pyramidal Implementation of the Lucas-Kanade Feature Tracker.* 2004.

Bullock, D., and S. Grossberg. "Neural Dynamics of Planned Arm Movements: Emergent Invariants and Speed-Accuracy Properties During Trajectory Formation." *Psychological Review* 95, no. 1 (1988): 49-90.

Canny, J. "A Computational Approach to Edge Detection." *IEEE Transactions on Pattern Analysis and Machine Leraning Intelligence* 8, no. 6 (1986): 679-698.

Chiaverini, S., B. Siciliano, and O. Egeland. "Review of the Damped Least-Squares Inverse Kinematics with Experiments on an Industrial Robot Manipulator." *IEEE Transactions on Control Systems Technology*, 1994.

Del Vecchio, D., R. M. Murray, and P. Perona. "Decomposition of Human Motion into

Dynamics-based Primitives with Application to Drawing Tasks." *Automatica* 39 (2003): 2085-2098.

Deo, A. S., and I. D. Walker. "Robot Subtask Performance with Singularity Robustness using Optimal Damped Least-Squares." *IEEE International Conference on Robotics and Automation.* Nice, France, 1992.

D'Souza, A., S. Vijayakumar, and S. Schaal. "Learning Inverse Kinematics." *IEEE/RSJ International Conference on Intelligent Robots and Systems.* Maui, Hawaii, USA, 2001. 298-303.

Duran, B., G. Sandini, and G. Metta. "Emergence of Smooth Pursuit using Chaos." *IEEE International Conference on Self-Adaptive and Self-Organizing Systems.* Boston, MA, USA, 2007.

Duran, B., Y. Kuniyoshi, and G. Sandini. "Eyes-Neck Coordination using Chaos." *Tracts in Advanced Robotics* 44 (2008): 83-92.

Fitzpatrick, P., G. Metta, and L. Natale. "Towards Long-Lived Robot Genes." *Robotics and Autonomous Systems* 56, no. 1 (2007): 29-45.

Flash, T., and N. Hogan. "The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model." *Neuroscience* 5 (1985): 1688-1703.

Fletcher, R. "A New Approach to Variable Metric Algorithms." *Computer Journal* 13 (1970): 317-322.

Freedman, E. G., and D. L. Sparks. "Coordination of the Eyes and Head: Movement Kinematics." *Experimental Brain Research* 131 (2000): 22-32.

Fumagalli, M., et al. *Learning to Exploit Proximal Force Sensing: a Comparison Approach.* Vol. 264, 149-167. Springer-Verlag Berling and Heidelberg GmbH & Co. K, 2010.

Fumagalli, M., M. Randazzo, F. Nori, L. Natale, G. Metta, and G. Sandini. "Exploiting Proximal F/T Measurements for the iCub Active Compliance." *IEEE/RSJ International Conference on Intelligent Robots and Systems.* Taipei, Taiwan, 2010. 1870-1876.

Gaskett, C., and G. Cheng. "Online Learning of a Motor Map for Humanoid Robot Reaching." *International Conference on Computational Intelligence, Robotics and Autonomous Systems.* 2003.

Gijsbert, A., G. Metta, and L. Rothkrantz. "Evolutionary Optimization of Least-Squares Support Vector Machines." *Annals of Information Systems* 8 (2010): 277-297.

Gijsberts, A., and G. Metta. "Incremental Learning of Robot Dynamics using Random Features." *IEEE International Conference on Robotics and Automation.* Shangai, China, 2011. n.a.

Goossens, H. H., and A. J. Van Opstal. "Human Eye–Head Coordination in Two Dimensions Under Different Sensorimotor Conditions." *Experimental Brain Research* 114 (1997): 542-560.

Guitton, D., and M. Volle. "Gaze Control in Humans: Eye–Head Coordination During Orienting Movements to Targets Within and Beyond the Oculomotor Range." *Journal of Neurophysiology* 58 (1987): 496-508.

Hagan, M. T., and M. Menhaj. "Training Feed-Forward Networks with the Marquardt Algorithm." *IEEE Transactions on Neural Networks* 5, no. 6 (1999): 989-993.

Haruno, M., D. M. Wolpert, and M. Kawato. "MOSAIC Model for Sensorimotor Learning Control." *Neural Computation* 13 (2001): 2201-2220.

Hersch, M., and A. G. Billard. "Reaching with Multi-Referential Dynamical Systems." *Autonomous Robots* (Springer-Verlag) 25 (2008): 71-83.

Hersch, M., E. Sauser, and A. Billard. "Online Learning of the Body Schema."

International Journal of Humanoid Robotics 5 (2008): 161-181.

Hoff, B., and M. A. Arbib. "A Model of the Effects of Speed, Accuracy and Perturbation on Visually Guided Reaching." In *Control of Arm Movement in Space: Neurophysiological and Computational Approaches*, by R. Caminiti, P. B. Johnson and Y. Burnod, 285-306. 1992.

Ho-Yul, L., Y. Byung-Ju, and C. Yungjin. "A Realistic Joint Limit Algorithm for Kinematically Redundant Manipulators." *IEEE International Conference on Control, Automation and Systems.* Seoul, 2007.

Ijspeert, A. J., A. Crespi, D. Ryczko, and J. -M. Cabelguen. "From Swimming to Walking with a Salamander Robot Driven by a Spinal Cord Model." *Science* 315, no. 5817 (2007): 1416-1420.

Ijspeert, A. J., J. Nakanishi, and S. Schaal. "Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots." *IEEE Conference on Robotics and Automation.* Washington, DC, USA, 2002. 1398-1403.

Irani, M., and P. Anandan. "A Unified Approach to Moving Object Detection in 2D and 3D Scenes." *IEEE Transactions on Pattern Analysis and Machine Learning* 20, no. 6 (1998): 577-589.

Kagami, S., Jr. J. J. Kuffner, K. Nishiwaki, M. Inaba, and H. Inoue. "Humanoid Arm Motion Planning using Stereo Vision and RRT Search." *Journal of Robotics and Mechatronics* 15, no. 2 (2003): 200-207.

Kaneko, K., and I. Tsuda. *Complex Systems: Chaos and Beyond.* Springer-Verlag, 2001.

Liegeois, A. "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms." *IEEE Transactions on Systems, Man, and Cybernetics* 7, no. 12 (1977): 868-871.

Lopes, M., A. Bernardino, J. Santos-Victor, K. Rosander, and C. von Hofsten. "Biomimetic Eye-Neck Coordination." *IEEE 8th International Conference on Development and Learning*, 2009.

Lucas, B. D., and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision." *Proceedings of Imaging Understanding Workshop.* 1981. 121-130.

Ma, Y., S. Soatto, J. Koseckà, and S. S. Sastry. *An Invitation to 3-D Vision; From Images to Geometric Models.* New York: Springer-Verlag, 2004.

Maini, E. S., L. Manfredi, C. Laschi, and P. Dario. "Bioinspired Velocity Control of Fast Gaze Shifts on a Robotic Anthropomorphic Head." *Autonomous Robots*, 2008: 37-58.

Mallat, S. G. "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11, no. 7 (1989): 674-693.

Metta, G., D. Vernon, L. Natale, F. Nori, and G. Sandini. "The iCub Humanoid Robot: An Open Platform for Research in Embodied Cognition." *IEEE Workshop on Performance Metrics for Intelligent Systems.* Washington, DC, USA, 2008.

Metta, G., G. Sandini, and J. Konczak. "A Developmental Approach to Visually-Guided Reaching in Artificial Systems." *Neural Networks* 12, no. 10 (1999): 1413-1427.

Mussa-Ivaldi, F. A., and E. Bizzi. "Motor Learning through the Combination of Primitives." *Philosophical Transactions of the Royal Society: Biological Sciences* 355 (2000): 1755-1769.

Natale, L., F. Orabona, F. Berton, G. Metta, and G. Sandini. "From Sensorimotor Development to Object Perception." *5th IEEE-RAS International Conference on Humanoids Robots.* Tsukuba, 2005. 226-231.

Nelder, J., and R. Mead. "A Simplex Method for Function Minimization." *Computer Journal* 7 (1965): 308-313.

Nori, F., and R. Frezza. "A Control Theory Approach to the Analysis and Synthesis of the Experimentally Observed Motion Primitives." *Biological Cybernetics* 93 (2005): 323-342.

Nori, F., L. Natale, G. Sandini, and G. Metta. "Autonomous Learning of 3D Reaching in a Humanoid Robot." *IEEE/RSJ International Conference on Intelligent Robots and Systems.* San Diego, CA, USA, 2007. 1142.

Parmiggiani, A., M. Randazzo, L. Natale, G. Metta, and G. Sandini. "Joint Torque Sensing for the Upper-Body of the iCub Humanoid Robots." *IEEE International Conference on Humanoid Robots.* Paris, France, 2009.

Phillips, J. O., L. Ling, A. F. Fuchs, C. Seibold, and J. J. Plorde. "Rapid Horizontal Gaze Movement in the Monkey." *Journal of Neurophysiology* 73 (1995): 1632-1652.

Poggio, T., and S. Smale. "The Mathematics of Learning: Dealing with Data." *Notices of the American Mathematical Society* 50, no. 5 (2003): 537-544.

Riesenhuber, M., and T. Poggio. "Models of Object Recognition." *Nature Neuroscience*, 2000: 1199-1204.

Rolf, M., J. J. Steil, and M. Gienger. "Goal Babbling Permits Direct Learning of Inverse Kinematics." *IEEE Transactions on Autonomous Mental Development* 2, no. 3 (2010): 216-229.

Rosander, K., and C. von Hofsten. "Visual-vestibular interaction in early infancy." *Experimental Brain Research* 133 (2000): 321-333.

Ruesch, J., M. Lopes, A. Bernardino, J. Hornstein, J. Santos-Victor, and R. Pfeifer. "Multimodal Saliency-Based Bottom-Up Attention: A Framework for the Humanoid

Robot iCub." *2008 IEEE International Conference of Robotics and Automation* . Pasadena, CA, USA, 2008.

Sanger, T. D. "Human Arm Movements Described by Low-Dimensional Superposition of Principal Components." *The Journal of Neuroscience* 20, no. 3 (2000): 1066-1072.

Schaal, S., and C. G. Atkenson. "Constructive Incremental Learning from only Local Information." *Neural Computation*, 1998: 2047-2084.

Schaal, S., C. G. Atkenson, and S. Vijayakumar. "Scalable Techniques from Nonparametric Statistics for Real-Time Robot Learning." *Applied Intelligence* 16, no. 1 (2002): 49-60.

Schaal, S., S. Vijayakumar, A. D'Souza, A. Ijspeert, and J. Nakanishi. "Real-Time Statistical Learning for Robotics and Human Augmentation." *International Symposium on Robotics Research.* Academic Press, 2002. 117-124.

Schilling, M. A. "Towards a General Modular systems Theory and its Application to Inter-Firm Product Modularity." *Academy of Management Review* 25 (2000): 312-334.

Schmitz, A., U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini. "Design, Realization and Sensorization of the Dexterous iCub Hand." *IEEE-RAS International Conference on Humanoid Robots.* Nashville, TN, USA, 2010. 186-191.

Sciavicco, L., and B. Siciliano. *Modelling and Control of Robot Manipulators.* 2nd Edition. 2005.

Shadmehr, R., and S. P. Wise. *Computational Neurobiology of Reaching and Pointing.* 2005.

Taiana, M., J. Santos, J. Gaspar, G. Nascimento, A. Bernardino, and P. Lima. "Tracking Objects with Generic Calibrated Sensors: An Algorithm Based on Color and 3D Shape Features." *Robotics and Automation Systems, special issue on Omnidirectional Robot*

Vision 58 (2010): 784-795.

Thorpe, S. J., R. Guyonneau, N. Guilbaud, J. Allegraud, and R. VanRullen. "SpikeNet: Real-Time Visual Processing with One Spike per Neuron." *Neurocomputing* 58 (2004): 857-864.

Tweed, D., B. Glenn, and T. Vilis. "Eye–Head Coordination During Large Gaze Shifts." *Journal of Neurophysiology* 73 (1995): 766-779.

Vijayakumar, S., and S. Schaal. "Fast and Efficient Incremental Learning for High-Dimensional Movement Systems." *IEEE International Conference on Robotics and Automation.* San Francisco, CA, USA, 2000.

Wang, L. C., and C. Chen. "A Combined Optimization Method for Solving the Inverse Kinematics Problems of Mechanical Manipulators." *IEEE Transactions on Robotics and Automation* 7, no. 4 (1991): 489-499.

Wätcher, A., and L. T. Biegler. "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming." *Mathematical Programming* 106, no. 1 (2006): 25-57.

Wolpert, D. M., and M. Kawato. "Multiple Paired Forward and Inverse Models for Motor Control." *Neural Networks* 11 (1998): 1317-1329.

Wrede, S., et al. "Interactive Learning of Inverse Kinematics with Nullspace Constraints using Recurrent Neural Networks." *Workshop on Computational Intelligence.* Dortmund: Fachausschuss Computational Intelligence der VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, 2010.