Increasing Perceptual Skills of Robots Through Proximal Force/Torque Sensors:

A Study for the Implementation of Active Compliance on the iCub Humanoid Robot

Matteo Fumagalli

DIST University of Genoa, Italy

RBCS IIT - Italian Institute of Technology

A thesis submitted for the degree of *PhilosophiæDoctor (PhD)*
XXII Doctoral School on Humanoid Technologies
April 2011

Current version: February 18, 2011

Thesis supervisors:

Dr. Francesco Nori
RBCS - Italian Institute of Technology

Prof. Giorgio Metta
RBCS - Italian Institute of Technology
DIST - University of Genova, Italy

Declaration:

This work has been carried out by Matteo Fumagalli during his Ph.D. course in Humanoid technologies, under the joint supervision of Dr. Francesco Nori and Prof. Giorgio Metta, at the Robotics, Brain and Cognitive Sciences Department, Italian Institute of Technology, Genova, Italy, directed by Prof. Giulio Sandini. His Ph.D. has been financially supported by the Italian Ministry of Education, University and Research (MIUR) and the Fondazione Istituto Italiano di Tecnologia (IIT).

1. Reviewer:

2. Reviewer:

Day of the defense:

Signature from head of PhD committee:

...to my family

Contents

CONTENTS

List of Figures

List of Tables

LIST OF TABLES

Introduction

During the last decade, interaction (with humans and with the environment) has become an increasingly interesting topic of research within the field of robotics. Currents trends in robotics foster research in the development of capabilities and skills which can make robots autonomous and safe (i.e. not dangerous). The evolving scenario indeed requires the human and the robot to coexist within a shared unstructured environment, to interact and perform cooperative or independent tasks precisely and safely. Many research fields are addressing these objectives from different perspectives: some examples are rescue robotics, home robotics, medical and rehabilitation robotics, just to cite few. Also in humanoid robotics the goal is to guarantee safe interaction between humans and robots. Moreover, within this field, the main goal is to create an autonomous system, which is capable of adapting to a continuously changing environment. A humanoid robot, in fact, is a mechatronic system which is capable of autonomously adapts to variations in the surrounding and learn from its own experience, without the supervision or guidance of a programmer or user. In this context, the programmer builds software modules that are properly interconnected in order to define the cognitive behaviors. These modules allow to exchange information, that come from the sensori-motor system and from their processing within the modules themself. From experience and its senses, the autonomous robot thus become able learn, adapt and take decisions about the action to perform. These behaviors are the result of the exploration process, which is achieved through the employment of a great variety of sensors. Proprioceptive, visual, aural, haptic sensors are required for the creation of the cognitive system. This thesis focuses on the exploitation of force information with the goal of creating a framework for the exploration process. The main aspects that have been analyzed are the increase of the perceptual capabilities of generic robotic systems which mount distributed force/torque sensors embedded within its links, and

artificial skin, and the exploitation of these information to achieve active compliance for the humanoid robot iCub.

1

The Role of Force Perception and Backdrivability in Robot Interaction

As human started to think at robot, safety was already one of the most important issues to achieve. It was the 1942 when Isaac Asimov introduced the laws of robotics in his *Runaround* story.
It was the 1981 when a 37-year old maintenance engineer at a Japanese Kawasaki plant, while working on a broken robot, died beaten by the robot itself.
Weng *et al.* (2009) reported:

> *In 1981, a 37-year-old factory worker named Kenji Urada entered a restricted safety zone at a Kawasaki manufacturing plant to perform some maintenance on a robot. In his haste, he failed to completely turn it off. The robots powerful hydraulic arm pushed the engineer into some adjacent machinery, thus making Urada the first recorded victim to die at the hands of a robot.*

It is now widely diffused in robotic research the concept that an autonomous robot must be capable of safely interact with the environment and with humans to carry on common duties. Currents trends in robotics foster research in the development of capabilities and skills which can make robots autonomous and safe (i.e. not dangerous). These capabilities cover an incredible variety of behaviors that the robot should perform in order to cope with uncertainties, noises and unpredictable events. The robotic

scenario indeed requires the human and the robot to coexist within a shared unstructured environment, to interact and perform cooperative or independent tasks precisely and safely.

The tasks the autonomous system is required to perform are dependent on the research field that is taken into consideration. Depending on the framework and the scenario the robot is embedded in, the word safety takes different meanings.

If we consider the case of an autonomous mobile robot, fault-tolerant methods (see Lussier *et al.* (2005)) have the goal to detect and limit the consequences of hardware and software problems (see Sisbot *et al.* (2006)). Danger index can be used to define the motion of the robot, to avoid collisions or dangerous situations Kulic & Croft (2005). Collision avoidance solutions, i.e. control strategies where commonly the end-effector trajectory or the manipulator configuration is changed during motion so as to avoid collisions with the surrounding or the self have been extensively studied in literature, but they still represent an open issue in robotics (see Kulic & Croft (2007); Minguez *et al.* (2008); Sisbot *et al.* (2010)).

More in general, the autonomous system relies on its sensor system and it becomes of primary importance to provide to the robot as many information as possible. Through perception the robot creates its own knowledge of the surrounding. The sensor system of robots can in general be constituted by sensors that are exploited to define high level control (we classify these types of sensors as *high-level sensor system*), and other which give a measurement of the state of the robot (which we classify as *low level sensor system*). The firsts allow to obtain a wide representation of the surrounding environment. Their measurement is very informative, but the interpretation is complicated and computationally expensive. Part of this category are cameras, laser, proximity sensors, radar. On the other side, low level sensor give a detailed measurement of the state of the robot. Their information is typically localized but can be easily processed. Force/Torque sensors, joint torque sensors, artificial skin, proprioceptive sensors are classical example of this kind of sensor system.

In general, all the possible sources of information (cameras, proximity sensors, proprioceptive sensors, etc.) must be included to improve the representation of the robot workspace, in order to fulfill its tasks while preserving safety. Uncertainties and noises may reduce the reliability of its perceptual representation, and lead to misbehaviors. In

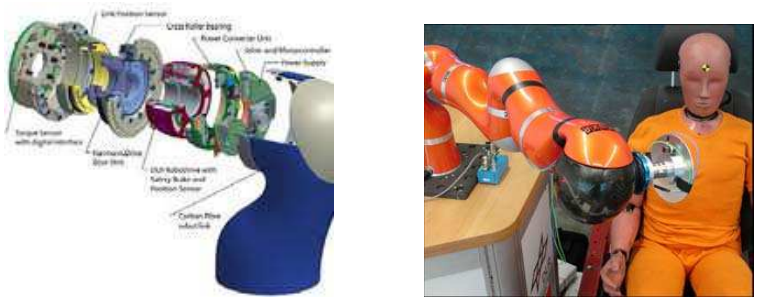
these cases, unsought and potentially dangerous contacts might occur between the human and the robot. An interesting analysis of the effects of possible impacts of robotic manipulators on humans can be found in Haddadin *et al.* (2008a,b).
In this framework, new generation mechanical design, novel actuators but, in particular, force information become of primary importance. Within this opening chapter, a brief overview of these mechanical choice that allow to reduce the risks that might consequence from the failure in the representation of the interaction scenario which rely on the *high level sensor system*, will be presented. Particular attention will be given to back-drivable systems and finally, on the mechanical choice of the sensors that allow to retrieve the information about the physical interaction of the robot with its surrounding. It will be shown the necessity to have the wider representation of the interaction and one possible solution that achieve this issue.

1.1 State of the Art on Physical Human Robot Interaction: Mechanical Solutions

1.1.1 Light Weight Design

It has been shown in Desantis *et al.* (2008) and Alami *et al.* (2006) that among the possible criterion that limit injuries that can be caused by collisions, weight reduction and lower inertia of the moving parts is fundamental. In Haddadin *et al.* (2007) an evaluation of the risks and the damages that can be produced to a human in case of crash with a light weight robot are shown. The DLR-III Lightweight robot is composed of light but stiff materials. Its motor gearboxes are harmonic drives which are compact but with high reduction ratio and efficiency. Torque sensors are placed at joint level, as will be shown in next section. It has been shown that lightweight robot have the intrinsic capability to reduce the moving inertia, but the high reduction ratio of the gearboxes, which is required to reduce the overall weight of the motors, introduces the problematic of the reflected inertia. The overall inertia sensed on the load side, and which is due to the inertia of the load, and to the inertia of the motor, whose energy is transmitted by the gearbox to the load side, is infact:

$$J_{all} = n^2 J_m + J_l \tag{1.1.1}$$



(a) (b)

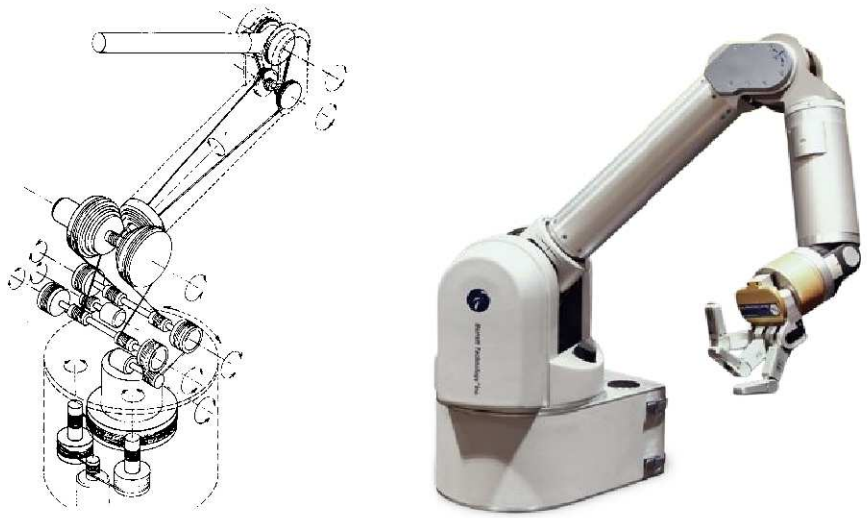
Figure 1.1: Left (1.1(a)): exploded view of the joint of the lightweight DLR-III developed at the German Aerospace Agency (DLR). **Right (1.1(b)): crash test involving the lightweight robot DLR-III**

1.1.2 Tendon Driven System

Another effective solution to reduce the weight of the moving parts is to employ cable transmissions. Motors are typically the major source of weight for robots moved by electro-magnetic actuators. One possibility, whose major example is constituted by the Barret Whole Arm Manipulator (WAM) Inc. (2010), Townsend (1988) and Salisbury *et al.* (1988), is to place the motors at the base of the robot. This solution allows to employ smaller motors, or to reduce the reduction ratio of the gearboxes. Since motor are not be moved together with the links, for they are placed on the robot base, the overall power necessary to move the system is in fact slightly reduced.
This mechanical solution, a part from reducing the moving inertia of the robotic structure, present another advantage that is o primary important in human-robot physical interaction. The employment of small reduction and small motors, have the intrinsic advantage to obtain a robotic mechanism which is also *backdrivable*. It will be shown in Section 1.1.3 the importance meaning and the importance of backdrivability characteristic in robotic mechanisms that interact in unstructured environments.

1.1.3 Backdrivable systems

Backdrivability is a term which refers to the easiness of transmission of movement from the output axis, to the input axis, as a consequence of an externally applied force. Different definition have been given in literature by Townsend (1988), Salisbury *et al.*



(a) (b)

Figure 1.2: Left (1.2(a)): sketch of the tendon based transmission of the lightweight robot from developed by Townsend (1988). **Right** (1.2(b)): Whole Arm Manipulator (WAM) from Barret Technology (Inc. (2010))

(1988), but also by Ishida & Takanishi (2006) in their study on the development of improved actuators for the Sony SDR robot.
When the robot operates in unstructured environment, where also human are present, backdrivability is essential for safe robotic-arm operation. With backdrivable manipulators the stability of the system during interaction control becomes less critical.
The advantage of nonbackdrivability instead is that it allows the motors to be de-energized when the robot is in a fixed position for a long periods. Nevertheless, on the contrary with respect to backdrivable systems, stability issues become more critical when non backdrivable mechanisms are employed. Frictional problems increase the difficulty in the control, for example, of Cartesian forces. The control of non backdrivable systems do not allow to rise the gains of the controller, because of the limited bandwidth that digital control can achieve. Moreover, there are passivity related problems which further limit the gains of the controllers, as shown by Colgate (1988) and Hogan & Buerger (2004). In this cases, a possible solution to obtain good performances of the controlled system, is to implement feed-forward model based control.

1.1.4 Series Elastic Actuators and VIA

Another possibility to reduce injury risks in physical human robot interaction is to add compliance. Compliance can be added to the exterior part of the robot (e.g. soft covers) to limit the effect of impact with rigid surfaces, but this aspect does not solve the issue of reducing the problems related with the reflected inertia at impacts.
One solution that allows to mechanically decouple the large reflected inertia of the motors from those of the links. Compliant transmission may ensure safe interaction, since the intrinsic elasticity stores the energy of impacts into the springs that allow to convert the kinetic energy of the moving links into potential energy of springs. Example of similar systems are the *Series Elastic Actuators* (SEA), designed at MIT by Pratt and Williamson, which proposed the first prototype in 1995 (Pratt & Williamson (1995)). The SEA are characterized by an elastic interconnection element between the motor and the load.
This solution reduce the admittance of the system thus resulting to be safe. Nevertheless, problems in the transmission of energy and motion between the motors and the links become difficult. SEA have been employed for the design of robotic mechanisms

such as for the humanoid robot COG, developed by Brooks *et al.* (1999). More recently, the same systems have been developed to design ither humanoid robots, such as DOMO (Edsinger-Gonzales & Weber (2004)) and TWENDY-ONE Iwata & Sugano (2009).

A more recent solution to the problem related with SEA, is to give to the system the possibility to vary its own internal joint stiffness. This is the case of Variable Impedance Actuators (VIA) (see Bicchi & Tonietti (2004)), where the stiffness of the joint can be mechanically varied in order to have improved motion performances while moving (high stiffness), and safe interaction when contact occurs (low stiffness). Nevertheless, recent studies have shown that certain solutions present an undesired effect that may lead to unsafe behaviors. Elastic elements, especially if combined with actuators can store great amounts of potential energy which, once released, can be extremely unsafe, as recently shown in Haddadin *et al.* (2010a).

1.2 Force Perception for Active Compliance

Conversely, when *active compliance* is employed, the robot behavior is grounded on the sensory system. Through force information, the autonomous system can build its own *epistemology* of the interaction, and subsequently performs its action. Current measurements, joint torque sensors or 6-axis force/torque sensors (the latter typically placed at the end-effector), are required to measure the interaction of the system. Exploiting the sensors information, the robot can carry out force regulation, react safely to contacts and also take decisions about the tasks (see Fumagalli *et al.* (2010); Haddadin *et al.* (2010b); Luca (2006); Mistry *et al.* (2010); Siciliano & Villani (1996)).

1.2.1 Current Control

Current measurement of electric motors can be exploited to have an estimation of the torque that the motor transmits to the load through the electro-mechanical coupling:

$$\begin{cases} I\ddot{\theta} + f_c + f_d = \tau_m - \tau_l \\ L\frac{d}{dt}i + Ri - K\dot{\theta} = V\tau_m = Ki \end{cases} \tag{1.2.1}$$

this equation is valid for ideal backdrivable systems. More specifically, assuming an ideal control of the current, the method is meant to cancel the dynamic of the electrical part, such that $i = i_{desired}$. By doing this, it is possible to directly command and regulate the torque transmitted to the motor.
Nevertheless, if the mechanical part presents sources of dissipation in the motor or in the transmission (i.e. coulomb friction f_c and dynamic friction f_d), the actual torque that is transmitted to the load is reduced by these factors. Moreover, limitation in the controller design previously neglected, and limitation due to the current capability of the motor reduce the effectiveness of this approach.
It is remarkable that the lower the reduction ratio, the more effective is the approach. Direct Drive systems (DD) are the most significative example, where the reduction ratio between the motor side and the load side is $1 : 1$. On the other side, even if DD systems do not suffer from the problematic of the reflected inertia, they require bigger motors with high torque capabilities, which are not ideal for safe application. Low reduction gearboxes with high efficiency instead allow the controlled system to use smaller motors, maintaining high torque transmission, but limited reflected inertia. This solution have been used in fact for the whole arm manipulator (WAM) proposed by Salisbury *et al.* (1988), already presented in Section 1.1.2.
Generally speaking, when high torque capabilities are required, high reduction ratios are employed and the measurement of the current does not allow to properly represent the torque that is acting at the load side.

1.2.2 Force Control

The information of 6-axis force/torque sensors give the most complete knowledge of the interaction. Force sensors in fact directly measure the generalized force that is applied at the sensor frame. These sensor have been widely employed for the control of the interaction in manipulation tasks. They have been classically employed for research in the control of the interaction of industrial robots (see Sciavicco & Siciliano (2005a) and Siciliano & Villani (2000)).
Classical applications place this powerful source of information at the end-effector of the manipulator structure. The reason for this, is that we are interested in retrieving a direct measurement of such information at the point that we want the interaction to

occur. Given the measurement at the tool level, we can think of projecting it at the joint level through the transpose of the Jacobian of the manipulator, and thus perform joint level torque control.

$$\tau = J^{\mathbb{T}} F \tag{1.2.2}$$

Nevertheless, this limitation might be valid in an industrial scenario. When the robot is working in an unstructured environment, the possible point of interaction is not known a priori. Other sensors can work together to prevent interaction at other levels (such as cameras or proximity sensors), but they cannot give any sort of representation of the actual interaction and transmission of generalized force between the robot and the object or human. In this situation a different framework is required. A method to obtain a more distributed measurement of the interaction is necessary.

1.2.3 Joint Torque Control

Joint torque sensors are a possible solution to the problems that remain unsolved given the approaches previously mentioned.
Torque sensors are typically placed on the load side of the joint. They allow to have a reliable information of the interaction that occur, without being influenced by problems related to friction to which current measurements are subjected to. On the other side, they are distributed along the kinematic chain, thus solving the problem of measuring the interaction at other points of the robotic structure.
Nevertheless, they do not allow to obtain a precise, rather than complete, representation of the interaction. Joint level torque sensors measure the torque that is working along the joint axis. They can be projected in the Cartesian space through the inverse of the relationship presented in Eq. 1.2.2. It is remarkable here that this operation suffer from singularities of the Jacobian, and the reliability of the retrieved information becomes configuration dependent.

1.3 Significance of the Project

Recent robotic trends and future perspective of robotics are defining new guidelines of robotic research. Autonomous robots should be capable of coexisting with humans in an environment which is dynamically changeable. Their skills and behaviors should be dominated by processes that are capable of adapting to different situation and unpredictable events. Their actions should be the consequences of events, but events should also be the cause of the adaptation process that allow the autonomous system to create its knowledge and representation of the action and reaction itself.
Enaction is one of the possible ways of organizing the knowledge of the autonomous system. It is one of the form of knowledge that start from the interaction with the world. Bruner (1968) gave the first definition of Enaction. Later, Varela and Maturana (see Maturana (1970, 1975, 1980); Varela (1979)) gave a second definition in which they expressed that enactive knowledge is knowledge that comes through action and it is constructed on motor skills, such as manipulating objects, riding a bicycle or playing a sport. In other words, the enactive knowledges of entities are the ones acquired by doing.
Sandini *et al.* (2007) reported:

> *The enactive stance asserts that cognition is the process whereby an autonomous system becomes viable and effective in its environment. In this, there are two complementary processes operating: one being the codetermination of the system and environment (through action and perception and contingent self-organization) and the second being the co-development of the system as it adapts, anticipates, and assimilates new modes of interacting*.

They were affirming that a developmental cognitive architecture must be capable of adaptation and self-modification. It must be able to adjust its parameters that defines its phylogenetic skills through learning and modification of the structure and organization of the system itself. Through adaptation, it is capable of altering its system dynamics based on experience, to expand its repertoire of actions, and thereby adapt to new circumstances. The development should be based on explorative behavior. Without force

exploration, and thus through haptic and force information, the cognitive system cannot be able to have haptic experience, necessary for learning and adaptation. In other words, it cannot be able to make its own representation of the surrounding.
Cognitive processes have been classically studied as abstract theories, mathematical models, and disembodied artificial intelligence. Nevertheless, the cognitive processes are strongly entwined with the physical structure of the body and its interaction with the environment.
Intelligence and mental processes are deeply influenced by the structure of the body, by motor abilities and especially skillful manipulation, by the elastic properties of the muscles, and the morphology of the retina and the sensory system. The physical body and its actions together play as much of a role in cognition as do neural processes, and human intelligence develops through interaction with objects in the environment and it is shaped profoundly by its interactions with other human beings.
In other words, the cognitive and physical interaction, are not independent: physical interaction help in setting rules for cognitive evaluations of the environment during interaction tasks, while cognitive aspects improve the physical interaction by setting suitable control interaction parameters. As a simple example, haptics is used to understand the characteristics of an environment (soft or rigid), while cognitive-based inference rules can be considered for compliance control of manipulators physically interacting with humans (if the person is a child, then the compliance should be high). Therefore, an improved analysis of the problems related to the physical interaction with robots becomes necessary. This topic must be addressed considering together the design of mechanism, sensors, actuators and control architecture in the special perspective for the interaction with humans.
Force perception plays a fundamental role in robotics. Since it is impossible to model every action in an unstructured anthropic environment, the intelligent connection of perception with action of robots implies the presence of autonomous behavior to solve real problems. This work has been aimed by the need and requirements of force perception to define the basis of a framework in which the iCub humanoid robot (see Tsagarakis *et al.* (2007b)) is able to perceive generalized external and internal forces, in order to allow the robot to have an haptic measurement during its explorative behavior. Starting from the assumptions and the studies previously shown, that robot interaction has been hypothesized to be at the basis of cognitive processes (see Sandini

et al. (2007)), through interaction, cognitive systems create their own knowledge (their epistemology) of the surrounding environment.
While exploring, the robot should be equipped with low level basic behaviors that allow to have a complete perception of the interaction, in order to control it, in a continuously evolving environment.
Through the exploitation of force information, proper control strategies can be adopted. As an example, active compliance allows to reduce the admittance of the system, thus resulting in a more backdrivable behavior of the robot, when interaction occur.
However, measuring the interaction using localized sensors, as proposed in previous sections, might not allow a full perceptual representation of the interaction scenario, in terms of forces and torques which rise over the whole structure.
Even if, among the different approaches shown in Section 1.2, torque sensors are the more distributed over the entire structure of the robot and can reliably measure the internal dynamic as well as the interaction occurring on their link, their measurement lack of completeness and depending on the configuration of the robot, they suffer from null projection on the joint angles.
This work focuses on methods to enrich the perceptual capabilities of force of the iCub robot, through the exploitation of a distributed set of FTSs over the entire structure of the robot. The method have been implemented, with the goal of designing a software architecture that allow to give to the robot the possibility to perceive the interaction, wherever it occurs and control it. This work is necessary to create a basic low level sensor system, that will allow to perform the exploration and to create its own representation of the surrounding.
The manuscript will be organized as follows:

- Chapter 2 gives a brief overview of the iCub humanoid robot. Here, a description of the main parts constituting the iCub, the sensors it is equipped with and briefly the electronic and the basic software architecture will be presented. Apart from the description of the mechanical structure, emphasis will be given to the description of the sensors that have been fundamental to carry out with this work: an inertial sensor placed in the head of the robot, and a total of four force/torque sensors distributed along the kinematic tree, one for each limb, placed in a proximal position.

- Chapter 3 will present the basic concepts of multi-body system dynamics, with particular emphasis on the recursive Newton-Euler algorithm. A graph formulation which embeds information of internal sensors will be presented. This formulation, which have been called *Enhanced Graph Formulation*, allows to easily represent the flow of kinematical and dynamical information along the mechanical structure. The method has the main advantage to exploit force/torque sensor measurements and artificial skin, that allow to represent dynamically the interaction forces that arise on the links, during an interaction scenario, in a non pre-determined position.

- Chapter 4 shows the specialization of the method that will be shown in Chapter 3 to the iCub humanoid robot. A summary of the steps necessary to build generic EOGs will be presented. Examples that shows the generality of the method will also be reported. Finally an EOG representing the graph structure of the iCub will be defined. Here the method will also be validated through experiments which show the effectiveness of the method for internal dynamic estimation, virtual external wrench and also virtual joint torque measurement.

- Chapter 5 shows the study of force control of generally coupled transmission system. An inverse dynamic control approach will be proposed, which allow to decouple the dynamic of motors from the dynamic of the joints. The model based approach allows to assign to the control variable an input command to the motors, that gives the possibility to directly control the joint acceleration, as if motor and joints live on the same axis of rotation. Possible control strategies will be introduced, which allow to perform compliant control and joint impedance control of the iCub joints.

- Chapter 6 will finally define the software architecture which allows to perform the calculation of the iCub dynamic, and the framework that have been implemented to perform low level compliance control.

1. THE ROLE OF FORCE PERCEPTION AND BACKDRIVABILITY IN ROBOT INTERACTION

2

Platform

In this chapter, the main platform which has been used to carry out with this work is introduced, the iCub robot. This platform is a humanoid robot that have been developed at the Italian Institute of technology and university of Genoa, for research in embodied cognition. More precisely, the iCub robot is the result of a research project for the study of developmental capabilities of cognitive systems, which has been funded by the European Commission through Unit E5 *Cognitive Systems, Interaction & Robotics* (see Metta *et al.* (2008); Sandini *et al.* (2007); Tsagarakis *et al.* (2007a,b) and (robotCub.org, 2010)). In the framework of embodied cognition, perception is a fundamental tool for robot learning and development, which are the result of the continuous interaction with the environment, and not the outcome of abstract reasoning. To achieve learning capabilities and self development and reasoning, *anthropomorphism*, *compliance* and *sensorization* are fundamental. These aspects are necessary to study human and humanoid development. To allow the robot to become a self-reasoning system, the basic perceptual aspects of human become requisites of the design process of a humanoid platform.

2.1 The iCub Platform

In this Section, an overview of the iCub humanoid robot is briefly described. The description reported here, does not cover all the aspects of the robot design, which are

necessary to the final goal of the research on humanoid technologies. It is shown here, in fact, a brief overview of the sensori-motor system and the some design choices, which are preparatory to fully understand the goal of this thesis. The following sections, in fact, show the main characteristics of the design of this robotic platform, from the point of view of its motion capabilities, motors and sensors placement, but also give a brief overview of the electronic and software components (which instead will be described in detail in Chapter 6).

2.1.1 Overview of the iCub Robot

The iCub humanoid robot has been designed with the goal of creating an open hardware/software robotic platform for research in embodied cognition, as expressed in Tsagarakis *et al.* (2007b). Its design has been mainly developed within RobotCub[1] a European funded project with the goal studying natural and artificial cognitive systems (see for example Metta *et al.* (2008)). As a major design specification, the iCub should be capable of interacting with humans and environments using its sensors to react in response to external events. At the current state, the iCub robot has a total of 53 degrees of freedom (DOF), 6 in each leg, 7 in each arm, 6 in the head, 3 in the waist and 9 for each hand. The iCub employs brushless and brushed DC motors equipped with high reduction gearboxes. The reduction gearbox employed to increase the torque capability of the electric motors mounted on the robot are *harmonic drivers*, which are mainly employed with the brush-less DC motors (1:100 or bigger reduction), and planetary gearbox for the reduction of most of the brushed DC motors (from 1:256 to 1:1024 reduction ratio). This solution allows compact designs but makes the robot passively non back-drivable. Employed motors can be divided into two broad categories: brushless and brushed DC motors. Typically brushless motors have been employed in the bigger articulated joints (shoulders, elbows, hips, torso, knees) while small brushed motors actuate the distal degrees of freedom (hand joints, neck, eyes). Remarkably, non standard mechanical choices have been used to rise the torque/volume ratio as will be briefly shown in next subsections. Particular attention will be given to the shoulder mechanism in sections Section 2.1.1.1 and also in Chapter 5, where force feedback control will be discussed.

[1] RobotCub project IST-FP6-004370.

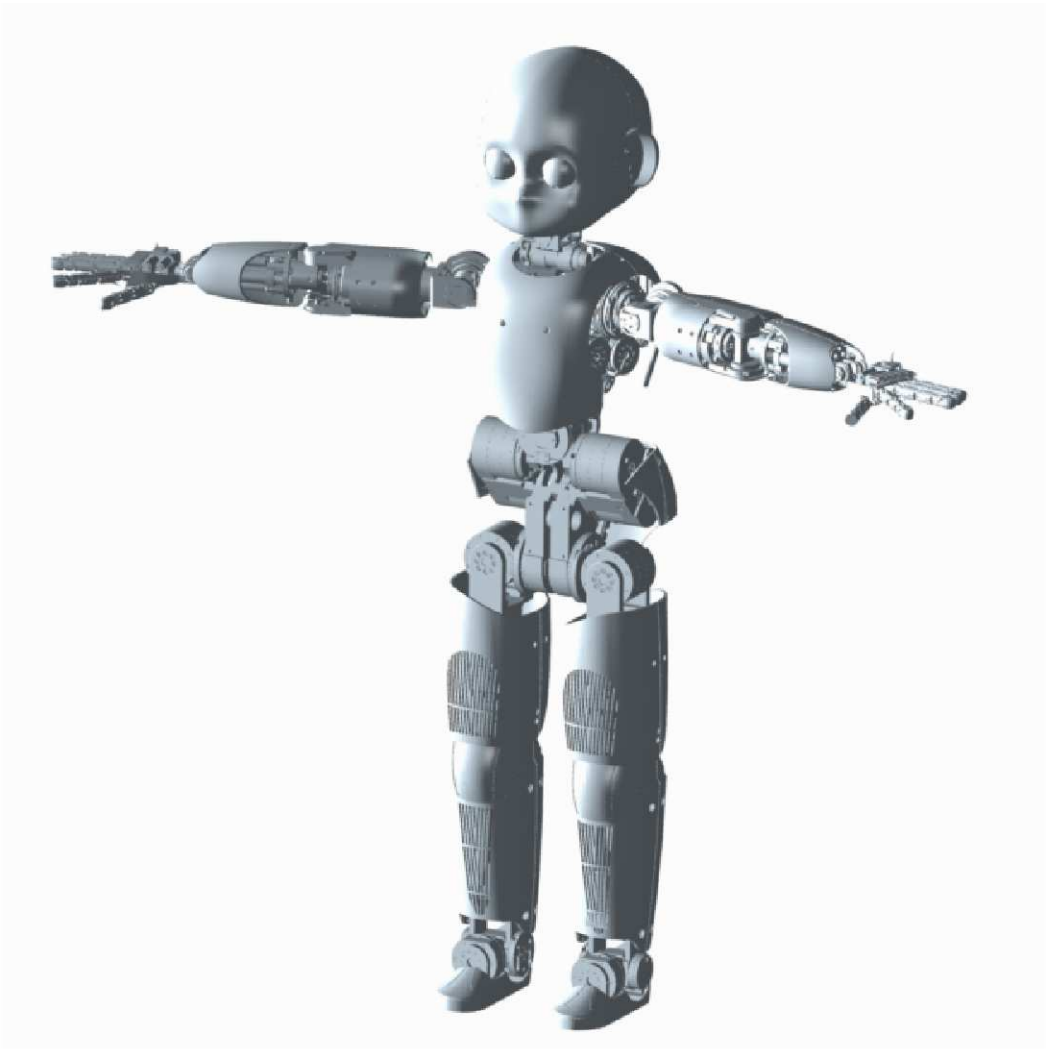


Figure 2.1: A CAD view of the iCub humanoid robot

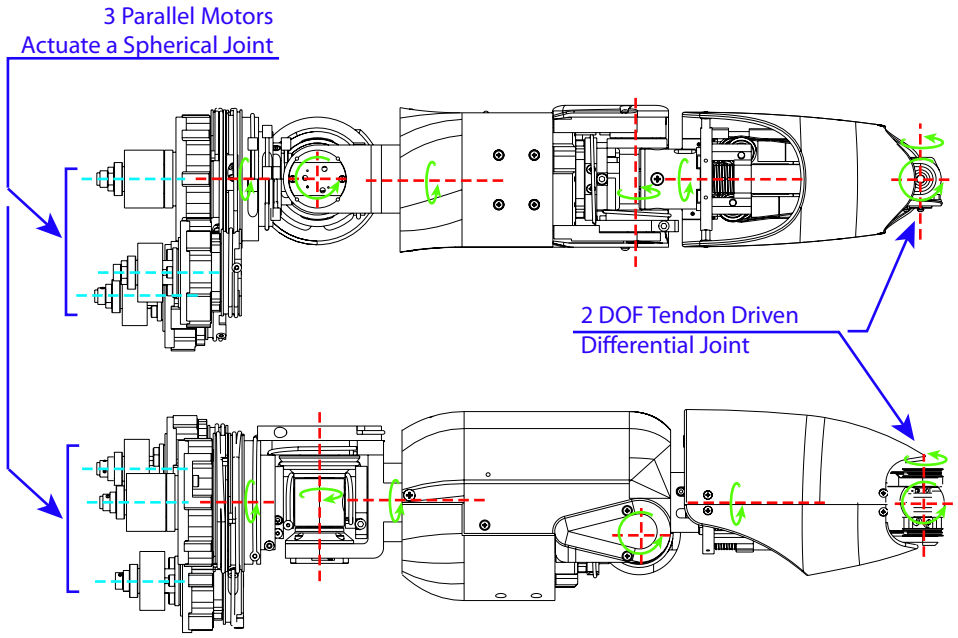


Figure 2.2: Sketch of the 7 degrees of freedom arm of iCub.

2.1.1.1 Arm

The iCub arms are 7 DOF open kinematic chains. Their upper part is commanded by four brushless motors, three for the shoulder movements and one for the elbow. The shoulder joint is a cable differential mechanism with a coupled transmission system (see Figure 2.3). Three coaxial motors (brushless frameless motors, RBE Kollmorgen series, with harmonic drive reductions, CSD series with 100:1 ratio) housed in the upper-torso move pulleys to generate the spherical motion of the shoulder (see Parmiggiani *et al.* (2009); Tsagarakis *et al.* (2007b) for a more detailed description of the shoulder universal joint). This design is evidently non-standard. Standard humanoids limb (e.g. the HRP-2 arm Kaneko *et al.* (2004)) have the shoulder motors in a serial configuration, with a single motor directly actuating a single degree of freedom. Tipically, in the HRP-2 arms, a pure pitch/yaw/roll rotation can be obtained by simply moving one motor and keeping the others fixed. In iCub, instead, a first bigger actuator (Motor 1 in Fig. 5.2) is capable of delivering $40Nm$ and two medium power motors (Motor 2 and Motor 3) provide $20Nm$ each. Motor 1 of Fig. 5.2 is directly connected

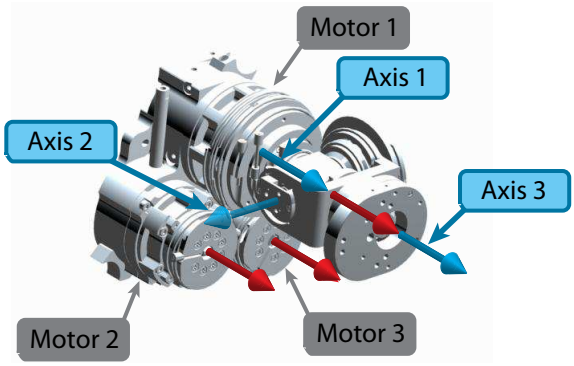


Figure 2.3: The iCub shoulder. A CAD view of the shoulder joint mechanism showing the three motors actuating the 3 degrees of freedom universal joint.

to the first joint (shoulder pitch), whereas Motor 2 and Motor 3 actuate two pulleys that are coaxial with Motor 1. The transmission of the motion from motors to joints is achieved through idle pulleys. Joint positions are measured by Hall effect based digital encoders with custom made electronics, (see `http://eris.liralab.it` (2010)). Motor 1 and Motor 3 mount the encoders directly on the motor shaft, while Joint 2 (actually the joint performing the yaw movement) mounts the encoder on the joint axis. Motors and joint movements are thus coupled with tendons and pulleys through a quasi-static relationship T_{mj} which can be represented as follows:

$$\dot{\theta}_m = T_{mj}\dot{\theta}_j \qquad\qquad T_{mj} = \begin{bmatrix} 1 & 0 & 0 \\ -r & r & 0 \\ -2r & r & r \end{bmatrix}, \tag{2.1.1}$$

where r is a constant value which depends on the radius of the pulleys, $\dot{\theta}_m = [\dot{\theta}_{m1}, \dot{\theta}_{m2}, \dot{\theta}_{m3}]^\top$ is the vector of motor angular velocities and $\dot{\theta}_j = [\dot{\theta}_{pitch}, \dot{\theta}_{roll}, \dot{\theta}_{yaw}]^\top$is the vector of joint angular velocities. The generalized dynamic of coupled system will be analyzed in Chapter 5. The elbow flexion/extension, is actuated by an independent frameless brushless motor, located in the upper arm. The joint is commanded with tendons in push-pull configuration, moving an idle pulley. The wrist is provided with 3 DOFs, namely pitch, roll and yaw (θ_{wp}, θ_{wr}, θ_{wy}), which correspond respectively to wrist flexion/estension, adduction/abduction and rotation. The roll movement is achieved by a single brushed motor directly coupled to the forearm. The pitch and yaw movements

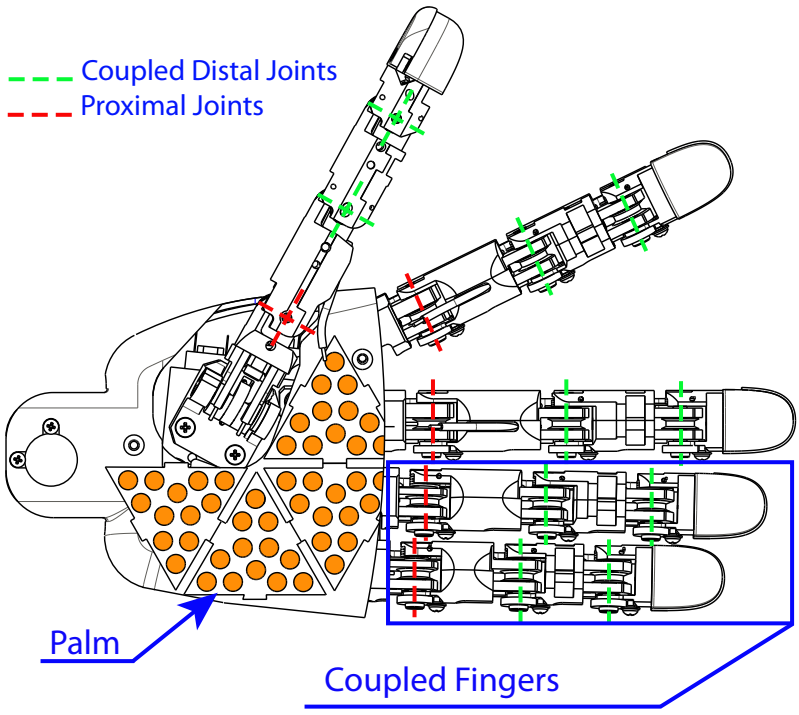


Figure 2.4: Particular CAD view of the iCub hand. 21 joints are actuated with 9 motors through tendon driven mechanisms. Tactile sensors are present on the palm.

instead are accomplished by two motors which move a semi-differential mechanism through tendons. Brushed motors (Faulhaber) have been employed for the wrist joints. Positions are measured through magnetic incremental encoders mounted on the motors.

2.1.1.2 Hand

The iCub hand has five fingers actuated employing tendon driven mechanisms. Each finger has four joints, but the hand has a total of 9DOF. Seven motors are placed remotely in the forearm and all tendons are routed through the wrist mechanism. This solution allows locating most of the hand actuators in the forearm rather than in the hand itself, where strict size and weight constraints are present. Two only motors are mounted directly on the hand: one of the motors controls the thumb abduction and the other actuates the fingers adduction/abduction. Moreover, the hand has a total of

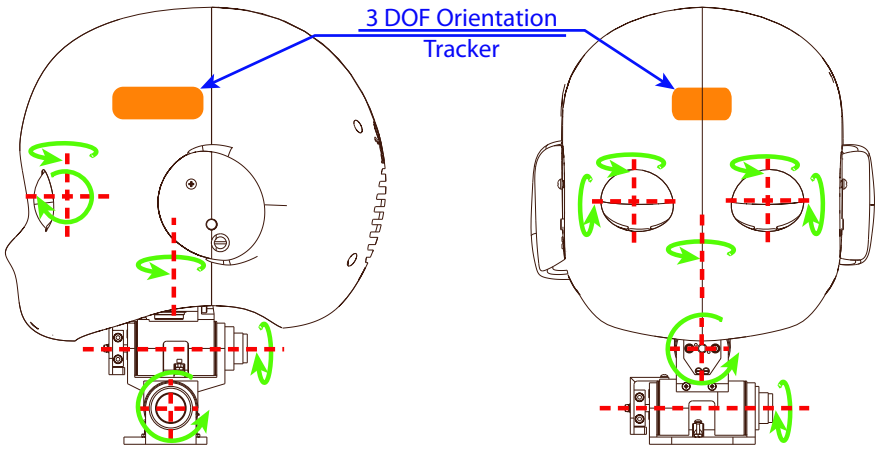


Figure 2.5: The 7 DOF head of the iCub robot. A 3 DOF Orientation Tracker is mounted at the top of the head.

17 joints. Both these joints mount tiny Hall effect position sensors. Finger positions are measured using 15 Hall effect sensors directly mounted on the phalanxes (three for each finger). The adoption of these solutions allow a design with very limited dimensions: the palm is $50mm$ long, $34mm$ wide at the wrist and $60mm$ wide at the fingers. The overall thickness of the hand is only $25mm$. On the palm is mounted a tactile array, which is a capacitance based touch sensor, which employs soft silicon rubber as dielectric material see Cannata *et al.* (2008a); Maggiali *et al.* (2008). On the tip of the fingers, a similar sensor is positioned, one for each finger (see Schmitz *et al.* (2010)).

2.1.1.3 Head and Waist

The head is equipped with two eyes, which can pan and tilt independently (4 DOFs), and is mounted on a 3-DOF neck, which allows the movement of the head as needed in the 3D rotational space (pitch, roll and yaw movements). The eyes exploit three brushed DC motors to control independently the pan and to simultaneously control the tilt. The eyes mounts two cameras which represent the iCub vision system. The neck is a 3 DOF serial kinematic chain which is moved with brushed DC motors and planetary gearboxes. On the head a PC104 is mounted, which is the core of the iCub motion control system. It allows the communication with other PC and the user, as will be shown in Section 2.2.2. The waist mechanism is a 3DOF kinematic chain (see

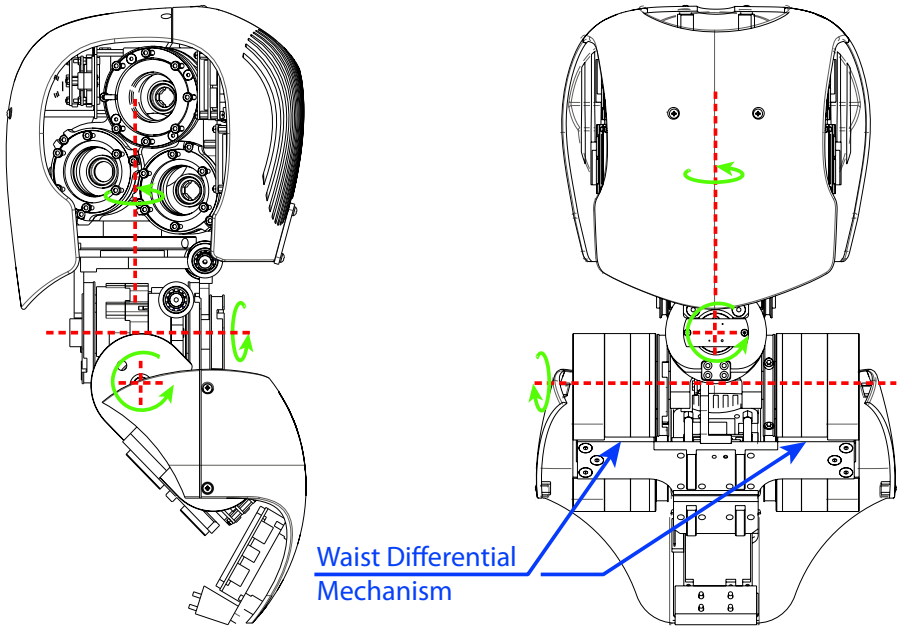


Figure 2.6: Sketch of the 3 DOF torso. A particular differential mechanism allow to rise the torque to volume ratio of the motors.

Tsagarakis *et al.* (2007a,b)). It makes use of a tendon-driven differential mechanism for pitch and yaw movements using two brushless motors. This configuration allows a better distribution of the joint torque on the motors, thus allowing a reduction of the mechanism dimension. Similarly to the shoulder actuation, the motor velocities $\dot{\theta}_m = [\dot{\theta}_{m1}, \dot{\theta}_{m2}]^\top$ and the joint velocities $\dot{\theta}_j = [\dot{\theta}_{pitch}, \dot{\theta}_{yaw}]^\top$ are kinematically coupled by the linear relation:

$$\dot{\theta}_m = T_w \dot{\theta}_j \qquad\qquad T_w = \begin{bmatrix} r & r \\ -r & r \end{bmatrix}, \tag{2.1.2}$$

where r is a constant value which depend on the dimension of the pulleys. The roll movement is actuated independently by a frameless brush-less DC motor with harmonic drive reduction (1:100).

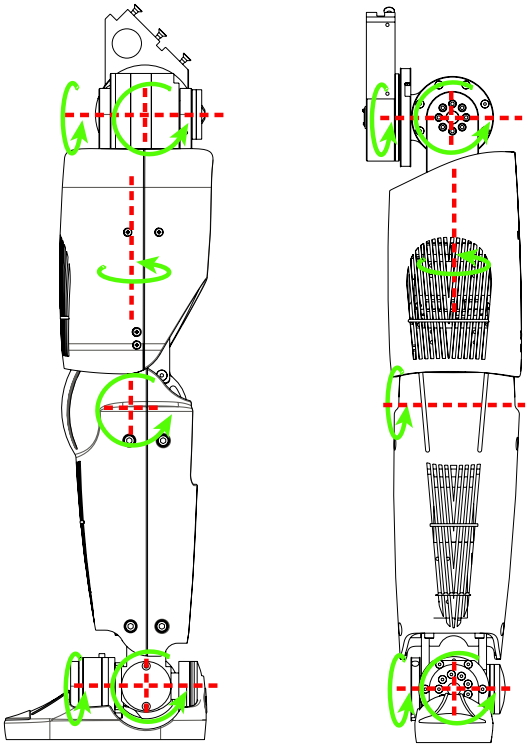


Figure 2.7: The 6 DOF leg of the iCub humanoid robots.

2.1.1.4 Legs

Legs are 6DOF serial kinematic. A detailed description of these mechanisms can be found in Tsagarakis *et al.* (2007a,b). All the six motors are frameless brushless (RBE Kollmorgen series) equipped with harmonic drive reduction of $100 : 1$ (CSD series). Absolute encoders directly measure the motor angles.

On the legs, proximal 6-axis force/torque sensors are mounted, one each leg.

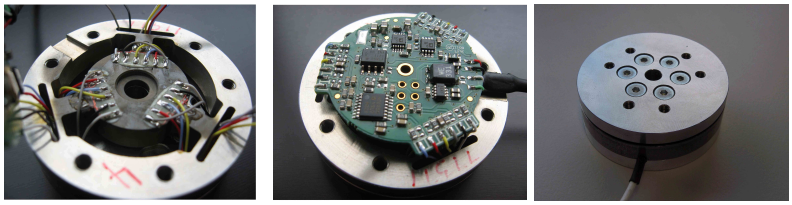
2.1.2 Electronics and Sensors

In this section, a brief overview of the iCub electronic and sensors will be given. A detailed description will be shown in Chapter 6, where the software and hardware architecture of the robot will be discussed. The iCub robot is equipped with a rich set of sensors, either commercial products or devices specifically realized for this platform,

which enables the robot to exploit visual, proprioceptive, kinesthetic, vestibular, tactile and force sensing.
A gigabit Ethernet interface allows the PC-104 to communicate with an external network, typically used for intensive data processing. CAN-bus lines are employed for the communication between the boards and the PC104 (see Sec. Section 2.1.1.3). The PC104 board has the principal role of collecting and synchronizing all the sensory and motor data. To achieve this, a CFW2 board is directly connected to the PC104 as the hardware interface between the overall 8 can-bus networks and the PC104. Motors are commanded with custom electronic boards mounting a Freescale 56F807 DSP. Two main control board have been employed, the BLL (BrushLessLogic unit) which control two brushless motors each board and the MC4 (MotorControl4) for the control of four DC motors for each board (see `http://eris.liralab.it` (2010)). The control rate of these boards is $1ms$. Force sensors embed an home made electronic board for strain data acquisition (Strain board). The associated circuitry samples and amplifies up to 6 analog channels which can be used to measure the voltage across 6 strain gauges in a Wheatstone bridge configuration. The analog to digital converter (AD7685, 16 bit, 250 Ksps, SPI interface) is multiplexed (ADG658) on the 6 channels and amplified with a standard instrumentation amplifier (INA155). In our specific case, the sampling rate is 1 kHz. The Strain Board has a CAN-bus interface which allows its connection directly on the CAN-network. A more detailed overview of the low level hardware connection will be detailed shown in Chapter 6.

2.1.2.1 the Force/Torque Sensor

Force/Torque sensors are the main source of information which will be discussed within this manuscript. Force/torque sensors allow to measure the interaction of the robot through a localized information about the internal generalized forces acting on the sensor position. This kind of sensor is very important while studying interaction tasks, and the position of the sensors along the kinematic chain of a robotic structure defines which information can be measured. All the four iCub limbs are equipped with custom made F/T sensors (see Tsagarakis *et al.* (2007a)). In the arms these sensors are placed in the upper part, in between the shoulder and the elbow; in the legs the sensor are located in between the hip and the knee (see Figure 2.8). These F/T sensors employ



(a) (b) (c)

Figure 2.8: The custom F/T sensor. **Left** (2.8(a)): Picture of the sensing elements where the strain gages are placed. **Center** (2.8(b)): the embedded board from which the measurements exits with sampled digital signal directly over a CAN-bus line, with a rate of $1ms$. **Right** (2.8(c)): The assembled sensor.

semiconductor strain gauges for measuring the deformation of the sensing elements. The signal conditioning and the analog to digital converters are embedded in the sensor. The data processing is performed on a 16 bit DSP from Microchip (dsPIC30F4013). The just described F/T sensor position differs from the classical distal configuration at the end-effector. Placing 6-axis F/T sensors at the tool level is a typical choice adopted in industrial robots, where the tool level is where it is required to measure the interaction (see Siciliano & Villani (2000)). Specifically, the iCub F/T sensors are mounted proximally in each limb. This solution has different advantages:

- The F/T measurements give information about the arm internal dynamic.
- External forces applied on the arm (e.g. not only forces applied at the end-effector) can be sensed.
- Information about the actual joint torques can be extracted from the the proximal F/T sensor.

Chapter 3 will show a formulation that allows to estimate internal wrenches, but also externally applied generalized forces given a set of distributed FTSs. Chapter 4 shows instead example of the application of the method, with particular attention to the iCub humanoid robot.

2.1.2.2 the Inertial Sensor

Figure 2.9 shows the position of the Xsens MTx-28A33G25 (see XsensMTx for details) which is mounted on the head of the iCub robot. This sensor allow to apply the

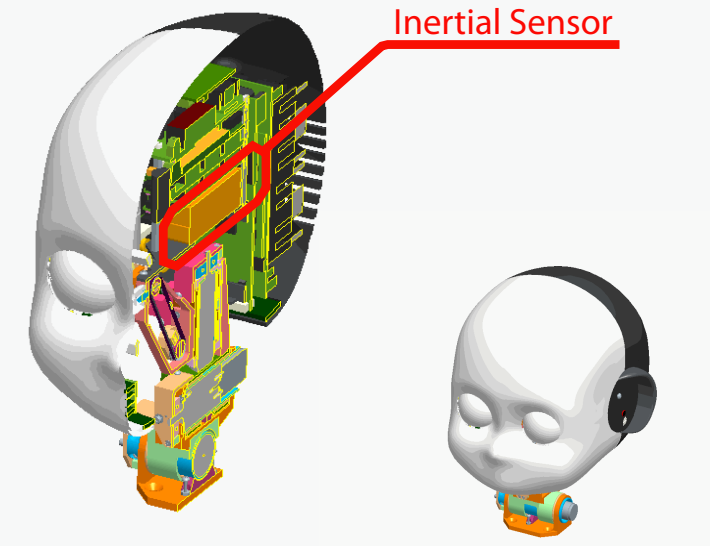


Figure 2.9: 3*d* dissection of the iCub head, which shows the *brain* of the robot, focusing on its vestibular system (the Inertial sensor (XsensMTx (2010))).

method presented in Chapter 3 to the iCub robot, in a general floating base configuration. A 3-DOF orientation tracker, in fact, allows to have a measurement of the linear and angular acceleration, and also of the angular velocity of the terminal link of the head. Given these measurements, it will be shown that it is possible to propagate the information along the links to perform the computation of kinematic quantities necessary to suddenly compute the inverse dynamic of the system.

2.2 The iCub Software

In this section, a brief overview of the software architecture which allow to command the robot is introduced. Again, the information that are presented here, have the goal to introduce the reader to the work which is presented within this thesis. A detailed description of the framework and the software and hardware architecture is reported in Chapter 6, where also the implementation issues regarding the computation of *virtual joint torque* measurements and the paths the information follow to suddenly perform control are addressed.

2.2.1 Overview of the Hardware Components

A cluster of Blades and standard PCs are interconnected through a 1GB Ethernet and constitute the core of the brain of iCub. These machines are dedicated to the high-level software, which is more computationally demanding (e.g. coordinated control, visual processing, learning), while the low-level motor control is implemented on the DSPs embedded in the robot body.

2.2.2 the Yarp Framework:

All the softwares which constitute the iCub high level *sensori-motor system* and cognitive architecture has been written using YARP (Metta *et al.*, 2006). YARP (Yet Another Robot Platform) is an open-source software framework that supports distributed computation under different operative systems (Windows, Linux and Mac OS) with the main goal of achieving efficient robot control. YARP facilitates code reuse and modularity by decoupling the programs from the specific hardware (using *Device Drivers*) and operative system (relying on the *OS wrapper* given by ACE (Gamma *et al.*, 1994; Schmidt, 2003)) and by providing an intuitive and powerful way to handle inter-process communication (using *Ports* objects, which follows the Observer pattern (Schmidt & Huston, 2002)). Furthermore, YARP provides mathematical (vectors and matrices operations) and image processing (basic *Image* class supporting IPL and OpenCV) libraries.
This software architecture allows to construct the *virtual force/torque sensors* framework and the subsequent joint torque control (that will be presented in Chapter 3 and Chapter 4, and Chapter 5 respectively) of the iCub as a collection of interconnected independent modules, running on different machines and exchanging data and control signals. All these notion will be discussed more in detail in Chapter 6. The choice of an architecture of this form is simple and practical: one single CPU, although powerful, can never be enough to cope with more and more demanding applications. From here, the necessity of dividing the processes between more calculators connected together on the same local network. Moreover, exploiting the modularity of the framework, it is also possible to generate behaviors that come out from the communication of the modules which allow a transparent diffusion of the information.

2.2.3 The iCubInterface:

The PC which is directly interfaced with the CAN-BUS runs a program (namely, *iCubInterface*) to communicate with the motor control boards.
The *iCubInterface* manages the YARP modules (see Metta *et al.* (2006)) that allow the communication between the robot interfaces and the user.
In the specific case, the *iCubInterface* allows to retrieve measurements from encoders, inertial sensor, FTs and in the future also of distributed tactile sensors over the entire body. Moreover, as will be shown in details in Chapter 6, it allows to send to the motor-control boards *virtual joint torque measurements* computed on the blades.

3

Propagation of Force Measurements Through MBSD

This chapter focuses on methods for building the dynamic model of single and multiple branches robotic mechanisms. The goal of this chapter is to give an overview of the formulation which has been used for the computation of the inverse dynamics of the iCub platform, presented in Chapter 2. The proposed method can be exploited for the computation of wrenches along the structure of generic serial mechanisms. In particular, the method allows to embed in the formulation, measurements of sensors attached to the robotic structure, in order to have a more reliable representation of internal generalized forces during robotic tasks, but also allows to retrieve *virtual force/torque measurements* of externally applied generalized forces. The chapter will be structured as follows: Section 3.1 introduces the reader to the basic concepts of rigid body dynamics; in Section 3.2 classical methods that are typically employed to model the kinematics and dynamics of articulated mechanisms are briefly introduced. Among different methods that allow to model and represent multiple links kinematic chain, more effort will be given to the Newton-Euler approach, as its recursive formulation can be easily exploited to perform the computations within graph formulation, as proposed in Section 3.3. Here a formulation for the computation of robot dynamic using graph theory is presented. It will be shown that this formulation, beyond the intrinsic recursive structure, that permit a very simple representation of the dynamic of multi-branched manipulators like humanoid robots, also allows to define *virtual measurements* of the internal dynamics by exploiting inertial and force/torque sensors.

Experiments which prove the effectiveness of the method, applied to the iCub robotic structure, will be presented in Chapter 4.

3.1 Introduction to Rigid Body Dynamics

The kinematics and dynamics of robots is generally described by set of elements, constrained together, which are characterized by parameters and intrinsic properties of length, mass and inertia. These elements are typically called *rigid bodies*. A *rigid body* is the basic element for the description of the dynamic of a rigid system. The relative and absolute motion of a set of rigid bodies (or links) is defined through *joint constraints*. A *joint constraint* defines a set of constraint equation which allow the kinematic and dynamic description of a link, in terms of relative displacement and constraint forces. The type of constraint equation which allows a mathematical formulation of joints depends on the mechanism which connects the links. It is shown in Cheli & E.Pennestri (2006) different but detailed formulation of methods adopted for multi-body system dynamics. Some of these methods are cited in Section 3.2.2 and in Section 3.2.3, where general methodologies for the description of the kinematics and dynamics of mechanisms are briefly introduced.

3.1.1 Kinematics of the Rigid Body

A rigid body can be defined as a set of points whose relative position does not change in time. It is in fact sufficient to describe the position and orientation of one of these points, with respect to a reference frame (generally indicated with $\langle \cdot \rangle$), that we can have the entire knowledge of the position and orientation of all the other points defining the rigid body.
In other words, the position and orientation of a rigid body, i.e. of all the points constituting the rigid body, is described by a point $p_i \in \mathbb{R}^3$ (actually the x, y and z coordinates with respect to a frame $\langle j \rangle$) and angles ϕ_i with respect to the $j-th$ frame of reference (see Sciavicco & Siciliano (2005b) for more details). The position and orientation of a rigid body, defined by frame i, with respect to frame j, can be described by

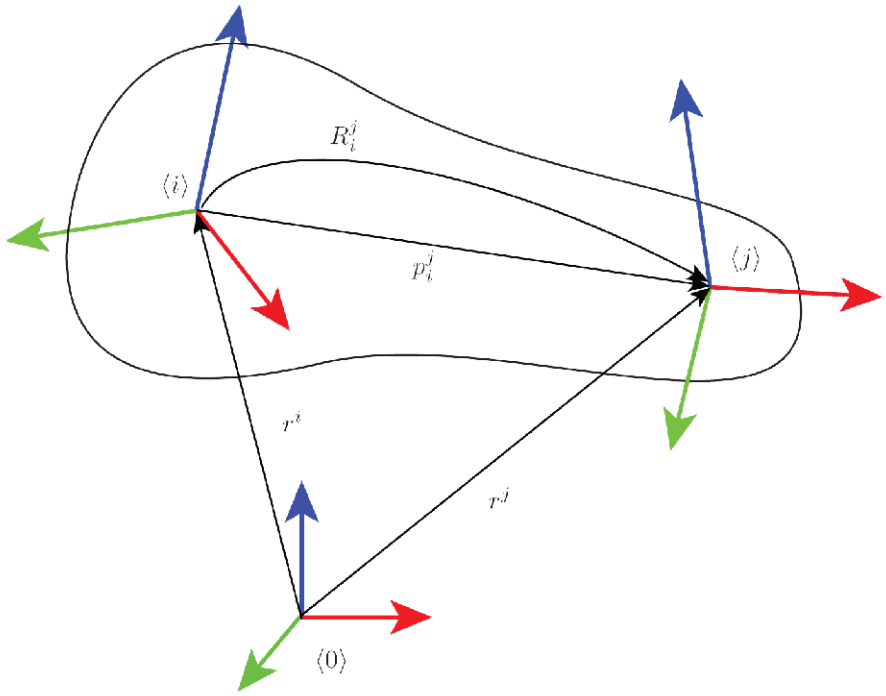


Figure 3.1: Sketch of a rigid-body link.

an homogeneous transformation. An homogeneous transformation is a matrix representation, in $\mathbb{R}^{4\times4}$, which defines the roto-translation matrix of one frame, with respect to another.

More specifically, an homogeneous transformation between two frames of references through a wide number of notation and conventions, with the form of matrix $T \in \mathbb{R}^{4\times4}$:

$$T_i^j = \begin{bmatrix} a_{xx} & b_{xy} & c_{xz} & x \\ a_{yx} & b_{yy} & c_{yz} & y \\ a_{zx} & b_{zy} & c_{zz} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_i^j & p_i^j \\ 0 & 1 \end{bmatrix} \tag{3.1.1}$$

where $R_i^j \in \mathbb{R}^{3\times3}$ represent the rotation matrix of $\langle j \rangle$ with respect to $\langle i \rangle$, and $p_i^j \in \mathbb{R}^3$ is the distance vector from the origin of $\langle i \rangle$ to $\langle j \rangle$

3.1.2 Dynamics of the Rigid Body

Let us consider a rigid body whose points P_i are characterized by a mass m_i ($i = 1, 2, ..., N$), where the relative distances between the points does not change in time, and with a center of mass in a point $C = [C_x, C_y, C_z]^T$. The overall force acting on the rigid body is given by:

$$\sum_{i=1}^{N} m_i \frac{d^2 r_i}{dt^2} = M \frac{d^2 r_C}{dt^2} = \sum_{i=1}^{H} f_{i,int} + \sum_{j=1}^{H} f_{j,ext} = \sum_{i=1}^{H} f_{i,ext} = F_{ext} \tag{3.1.2}$$

The equation describes the linear motion of a rigid body. In particular it shows that the motion of a set of points of mass m_i, rigidly constrained together, is equal to the motion of one body of overall mass $M = \sum_i m_i$, subject to one external force $F_{ext} = \sum_k f_{ext,k}$, being $\{f_{ext}\}$ a set of external forces acting on different points of the rigid body.

Similarly, the angular momentum balancing follows:

$$M r_C^{\langle o \rangle} \frac{d^2 R_{\langle o \rangle}}{dt^2} + \frac{d(I\omega)}{dt^2} = \sum_{i=1}^{H} \tau_{\langle o \rangle, i} = \tau_{\langle o \rangle, ext} \tag{3.1.3}$$

being $\langle o \rangle$ a generic frame of reference with origin in $\langle o \rangle$, and I the inertia tensor of the rigid body.

3.2 Dynamics of Serial Mechanisms

The dynamic of manipulators is described by means of rigid bodies, connected together through joints. The term *link* is used to refer to a rigid body, characterized by a frame of reference defined on a point *b* moving with respect to another frame *a* and whose relative position can change according to the definition of the joint constraint.

A manipulator, or multiple-link chain, is a set of links connected through joints. Here we limit the treatment to revolute joints. A revolute joint is a set of five scalar equations which defines a constraint between two links a and b, on the point p_a, where the joint is located, and p_b. With reference to figure Fig. 3.2, a revolute joint introduces a constraint that forces the position p_b of the origin of frame $\langle b \rangle$ and p_b of the origin

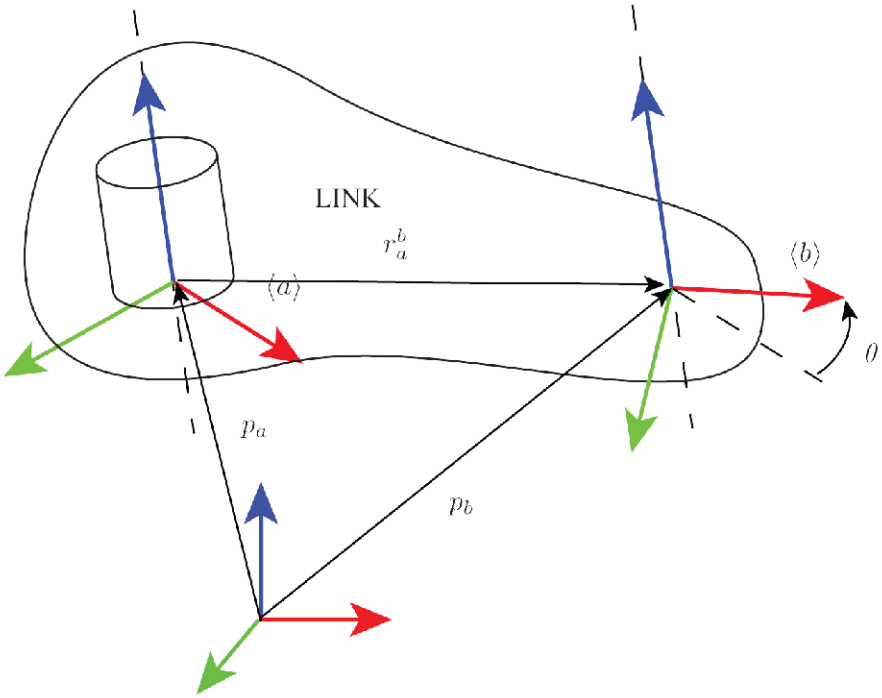


Figure 3.2: Sketch of a link with revolute joint.

of frame $\langle b \rangle$ to have a constant distance vector r_a^b (which correspond to three scalar equation $\Psi^p(p_a, p_b) = r_a^b$) and the orientation of z_a of frame $\langle a \rangle$ and z_b of frame $\langle b \rangle$ to have the same direction (which correspond to two scalar equation $\Psi^z(z_a, z_b) = 0$). The overall number of relative degrees of mobility of the kinematic couple of the two links is thus one, and refers to the relative angular position of x_a of frame $\langle a \rangle$ and x_b of frame $\langle b \rangle$, that we call θ.
Remarkably, different joints introduce different constraint equation. In this book, revolute joints are considered. For the definition of other kinematic couples, the interested reader should refer to Cheli & E.Pennestri (2006)

3.2.1 Notation

Let us introduce the notation of symbols that will be used to denote kinematic and dynamic quantities. With reference to figure Fig. 3.3 we call:

- $\langle \cdot \rangle$ generic Cartesian reference frame, where $\langle i \rangle$ refers to the reference frame attached to the i-th joint

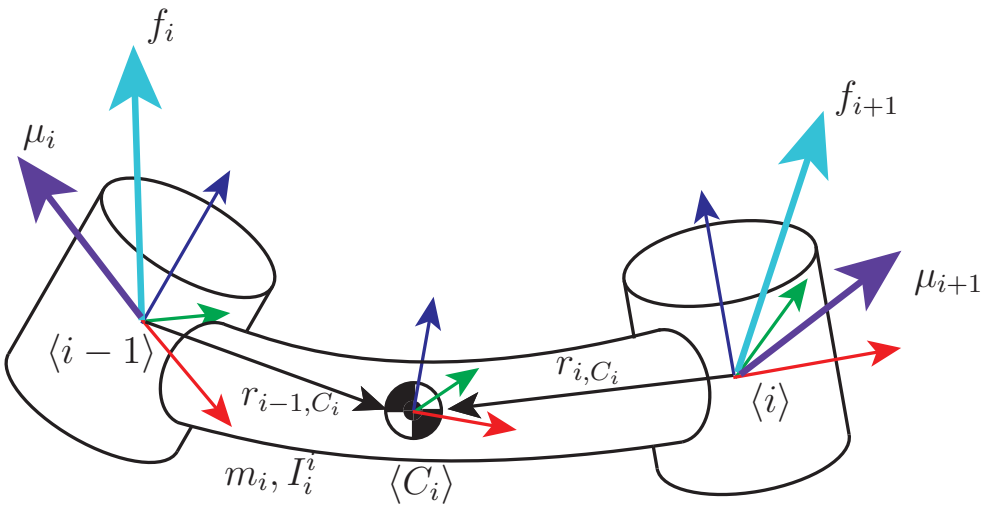


Figure 3.3: Notation for the i-th link of a kinematic chain.

θ_i the angle associated to the i-th joint. The vector of joint coordinates of the manipulator is denoted $\theta \in \mathbb{R}^n$

$\ddot{p}_i$ $\in \mathbb{R}^3$, denotes the linear acceleration of frame i-th

$\omega_i, \dot{\omega}_i$ $\in \mathbb{R}^3$, the angular velocity and acceleration of $\langle i \rangle$h

v_a : given $v \in \mathbb{R}^n$ a generic n-dimensional vector, v_a is v expressed in $\langle a \rangle$

R_a^b the $SO(3)$ rotation matrix from $\langle a \rangle$ to $\langle b \rangle$

$r_{a,b}$ distance vector r from $\langle a \rangle$ to $\langle b \rangle$.

C_i $\in \mathbb{R}^3$ the coordinate vector of the center of mass of link i-th, with respect to $\langle i \rangle$

z_i z-axis of $\langle i \rangle$ h, aligned with the axis of rotation of joint i

m_i mass associated with the i-th link

$\bar{I}_i^i$ $\in \mathbb{R}^{3\times3}$, defined with respect to the center of mass oriented as the frame $\langle i \rangle$, represent the inertia tensor of the i-th link

f_i $\in \mathbb{R}^3$, represent the forces applied on $\langle i \rangle$, that link $i+1$ exert on the i-th link

μ_i $\in \mathbb{R}^3$, represent the moment applied on $\langle i \rangle$, that link $i+1$ exert on the i-th link

$\tau_i \in \mathbb{R}$ the joint torque, i.e. the component of μ_i along z_i

$w_i \in \mathbb{R}^3$ the wrench (or generalized forces) $w = \begin{pmatrix} f \\ \mu \end{pmatrix}$ applied on $\langle i \rangle$, that link $i+1$ exert on link i

This notation allows to better understand the frames to which the quantities are referred. The formulation of the dynamics of multi-body systems requires the definition of these quantities, independently from the method used to compute the dynamics. In general, there are a lot of formalisms which allow to have a description of multi-body system dynamics, from the point of view of both the kinematics and dynamic characterization of the system. Section 3.2.2 shows some of the methods to describe the kinematics of manipulators, while Section 3.2.3 gives an idea on different formulations to describe the robot dynamics. In this chapter a general formulation which addresses the kinematics and dynamics computation for multiple branched robotic system, adopting graph theory, will be presented in Section 3.3. The formulation allows to define the robotic kinematic and dynamic structure of robotic mechanisms. It will be shown how sensors measurements can be exploited to enhance the representation of the graph structure of a multi-body system. In particular, the formulation allows to represent both known quantities (i.e. information coming from sensor measurements) and unknown quantities (i.e. the result of the computation) as will be shown in Section 3.3.1. Even though the formulation is generic from the method adopted for performing the computation of the unknowns, Section 3.4 specializes the recursion of kinematic and dynamic quantities exploiting the classical Newton-Euler formulation.
Moreover, in order to cope with external wrenches applied at continuously changing locations, graph representation of kinematic chain allows to define a theoretical, but also practical, framework which dynamically adapts to the contact locations. Classical pre-order and post-order traversal of this dynamically evolving graph allow to compute whole-body dynamics and external wrenches estimations. It will be shown that, under suitable conditions, a maximum of $N+1$ external wrenches can be estimated from N force/torque sensors, when each of the $N+1$ externally applied generalized forces refer is applied uniquely to a subgraph, bounded with known wrenches. The result of this formulation is a method which allows to enhance the perceptual capabilities of a robotic system in a physically interactive scenario, exploiting the information

of the available kinematic and dynamic model in combination with the measurements coming from the distributed sensors.

3.2.2 Kinematic Description: the Denavit-Hartenberg notation

The kinematic of manipulators can be described with a wide number of notation and conventions. Some methods that can be employed for the definition of a multi-link kinematic chain are:

- ***the method of constraint equation***: the link is described with 6 rigid body equation in free space (or 7, depending on the angular representation). Depending on the joint which connects the links, constraint equation are defined to reduce the number of degrees of freedom of the body. An example of the representation of kinematic constraints has been proposed in Section 3.2, for the definition of revolute joints.

- ***the method of natural coordinates***: allows a simple representation of rigid bodies in the absolute frame of reference.

- ***the Denavit-Hartenberg notation***: procedural and easy. It allows the description of links kinematic by employing four parameter, which define the relative position and orientation of sequential reference frames. It introduces constraints on the choice of reference frames.

These and other methods can be exploited for the representation of the kinematic of multi-branched robotic mechanisms.
In this chapter we define the structure of kinematic chains exploiting the Denavit-Hartenberg notation (see Sciavicco & Siciliano (2005b)). As previously mentioned, it allows the description of the position and orientation of each link relatively to the previous link, with a set of four parameters representing the relative distance and rotation angles between the reference frames. Particular links are the base link and the final link, whose frames of reference lack of a unique definition.
The method consists in defining one frame for each link, according to some rules (see

Sciavicco & Siciliano (2005b)) which allow the kinematic representation of the link in the form of an homogeneous transformation:

$$T_i^{i-1} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i)cos(\alpha_i) & sin(\theta_i)sin(\alpha_i) & a_i cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i)cos(\alpha_i) & -cos(\theta_i)sin(\alpha_i) & a_i sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2.1}$$

being, when revolute joints are considered, θ_i the relative rotation around the joint axis z_i of frame i with respect to frame $i-1$. α_i, a_i and d_i are fixed parameters describing the link kinematic

3.2.3 Dynamic Description: the RNEA

The dynamic description of a mechanical system can be achieved adopting different methods. Some relies on energetic formulation and conservation theories, such as the *Lagrange formulation*, while others relies on the balancing of the forces and moments acting on the rigid bodies, as the *Recursive Newton-Euler Algorithm* (RNEA). In the first case, the formulation gives origin to sets of equation in closed form that can be mainly employed for the analysis of control scheme and system properties. The latter instead are mostly used for implementation, due to the intrinsically recursive formulation, which allows fast numerical computation of the inverse dynamic, which is essential for real-time control systems. In this section, the analysis focuses on the RNE algorithm. We show the formulation of the classical method, and its adaptation to graph formulation.

The analysis of systems dynamic typically consists in two main steps: first, kinematic quantities, actually the linear position and orientation of frames, and corresponding velocities and accelerations of the links must be determined; secondly the computation of dynamic quantities is performed.

The classical Newton-Euler algorithm performs these computation through a recursion in two directions along the kinematic chain. A forward recursion allows to determine the kinematic quantities referred to the links' frame of reference, actually their position, velocities and acceleration. A backward recursion instead performs the compu-

tation of wrenches acting on the frame of references of the links. In the backward recursion, forces and moments acting on each link are determined through equations which consider the balancing of these quantities on the rigid body (as already introduced in Section 3.1.2).

The classical RNE equation follows here: adopting the Denavit-Hartenberg notation (once again refer to Sciavicco & Siciliano (2005a) for details), define a set of reference frames $\langle 0\rangle$, $\langle 1\rangle$, ..., $\langle n\rangle$ attached at each link. Considering a grounded manipulator, set the velocities and acceleration of the base frame as: $\ddot{p}_0 = -\mathbf{g}$, $\omega_0 = [0,0,0]$ and $\dot{\omega}_0 = [0,0,0]$. The Newton-Euler kinematic step consists in the propagation of velocity/acceleration information from the base to the end-effector (forward kinematics), considering the relative velocities and acceleration between subsequent links, induced by joint motion:

$$\begin{aligned}
\omega_{i+1} &= \omega_i + \dot{\theta}_{i+1} z_{i+1}, \\
\dot{\omega}_{i+1} &= \dot{\omega}_i + \ddot{\theta}_{i+1} z_{i+1} + \dot{\theta}_{i+1} \omega_i \times z_{i+1}, \\
\ddot{p}_{i+1} &= \ddot{p}_i + \dot{\omega}_i \times r_{i,i+1} + \omega_{i+1} \times (\omega_{i+1} \times r_{i,i+1}),
\end{aligned} \tag{3.2.2}$$

where z_{i+1} represent the z-axis of frame $i+1$. Measuring θ_i, $\dot{\theta}_i$, $\ddot{\theta}_i$ the above equations can be iterated to retrieve the i-th link angular velocity and acceleration (ω_i, $\dot{\omega}_i$) and linear acceleration ($\ddot{p}_i$).

Considering that the system is moving freely (i.e. without interacting with the environment), the robot dynamics is computed starting from the end-effector (where f_{i+1} and μ_{i+1} are set equal to zero) to the base. For each link, the force and torque components on joints which allow the maintenance of the system equilibrium are the computed as:

$$\begin{aligned}
f_i &= f_{i+1} + m_i \ddot{p}_{C_i}, \\
\mu_i &= \mu_{i+1} - f_i \times r_{i-1,C_i} + \\
&\quad + f_{i+1} \times r_{i,C_i} + \bar{I}_i^i \dot{\omega}_i + \omega_i \times (\bar{I}_i^i \omega_i),
\end{aligned} \tag{3.2.3}$$

where:

$$\ddot{p}_{C_i}^i = \ddot{p}_i^i + \dot{\omega}_i^i \times r_{i,C_i}^i + \omega_i^i \times (\omega_i^i \times r_{i,C_i}^i). \tag{3.2.4}$$

Assuming the system dynamical parameters are known (m_i, $\bar{I}_i^i$, r_{i-1,C_i}, r_{i,C_i}), wrenches are thus propagated to the base frame of the manipulator so as to retrieve f_0 and μ_0.

3.3 Dynamics of Multiple Branched Mechanisms: a Graph Formulation

Graph theory has been extensively used to represent mechanical systems (see Featherstone & Orin (2008); Wittenburg (1994)) and kinematic chains, producing compact and clear models, in matrix forms with beneficial properties (e.g. branch-induced sparsity, shown in Featherstone (2010)) when the connectivity among its elements is expressed. There is not a unique choice for a graph representing a chain: for example, in Featherstone (2007) graphs are undirected, nodes and arcs represent bodies and joints respectively; the resulting graph is undirected (i.e. non-oriented), but nodes are "labeled" according to a "regular numbering scheme".
This section presents the theoretical framework of the *Enhanced Oriented Graphs* (EOG), applied to the computation of both internal and external wrenches applied to single and multiple branches, generally non-grounded, kinematic chains. The proposed method is independent on the equation exploited for performing the calculation. Here, an extension of the classical RNEA (see Featherstone & Orin (2008); Sciavicco & Siciliano (2005a) for details on the RNE algorithm) will be adopted. Within this manuscript the *recursive Newton-Euler algorithm* will be employed, but it is remarkable to underline that this choice is not the unique that can be addressed for computing the inverse dynamic, adopting the graph formulation that will be presented in next sections. Here the classical *RNEA* will be rearranged to allow its application in a versatile framework for performing the computation of kinematic quantities and wrenches exploiting sensor measurements. More specifically, kinematic chains are represented as graphs. Differently to classical approach for inverse dynamic computation exploiting graph theories, the graph is enhanced with specific nodes representing both known and unknown (kinematic or dynamic) variables. As a consequence to the fact that within this formulation sensors will be considered in order to perform a more reliable estimation of the internal kinematic and dynamic quantities, nodes representing sensors placed along the kinematic tree, and nodes representing virtual sensors (e.g. unknown

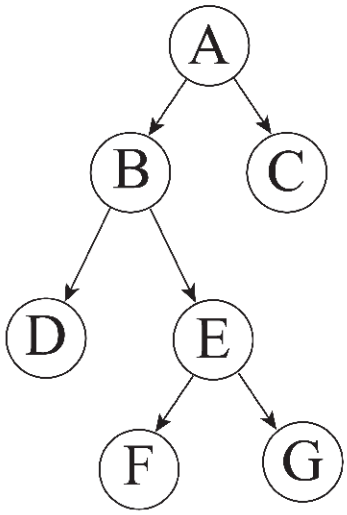


Figure 3.4: Example of multiple branched graph structure. When the graph is visited in pre-order, from the root A the visited nodes are B, D, E, F, G, C. When post-order, the visiting order is F, G, E, D, B, C and A

external wrenches acting on a certain link), will be added to the formulation. Remarkably, not all the unknowns will be specified a-priori (e.g. contacts at arbitrary locations might appear and other contacts might be removed) and therefore the graph structure will be adapted accordingly.

The computations of the system dynamics are performed through a pre-order and a post-order traversal visit of the graph itself, depending on the quantity to be computed, as will be shown in next sections. As a matter of completion, pre-order traversal visit of a graph means that from the root starts the propagation of the information to all the leaves of the tree. On the other side, post-order traversal refers to the way of visiting a graph, in which the propagation of the information starts from the leaves and sub-branches, to finally visit the root. Figure 3.4 describe these two possibilities for visiting a graph. It is remarkable here that, within this context, FTS, Inertial sensors, encoders and tactile arrays play a crucial role. More specifically, Inertial sensors provide information of the root leaves of the graph representation of kinematic quantities (as will be shown later on). Force/torque sensors allow to have an estimation of multiple external wrenches, but also a representation of the same internal quantities. Encoders are necessary for the propagation of the information along the structure, which passes from one edge to the others through the links, while distributed tactile sensors are primarily

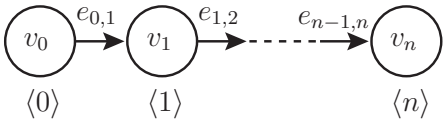


Figure 3.5: An open chain represented as a graph.

used to compute the presence and the location of externally applied wrenches. The resulting graph will thus allow to dynamically represent the unknown quantities (both kinematic and dynamic) of a link.

The resulting evolving graphical description of the chain modifies the way the graph is visited during the recursion, thus changing in particular the direction along which the information is propagated in the graph. In order to cope with this evolving representation another difference with respect to previous graphical representations is introduced. The kinematic chain will be represented as an *oriented* graph: the direction along which edges are traversed will determine the recursion formula to be employed, as will be shown in Section 3.4.1 and Section 3.4.2.

3.3.1 The enhanced graph representation

Let us consider an open (single or multiple branches) kinematic chain with n DOF composed of $n+1$ links. The i-th link of the chain is represented by a vertex v_i (sometimes called node). A hinge joint between the link i and the link j (i.e. a rotational joint) is represented by an oriented edge $e_{i,j}$ connecting v_i with v_j (see Figure 3.6). The orientation of the edge can be either chosen arbitrarily (it will be clear later on that the orientation simply induces a convention) or it can follow from the exploration of the kinematic tree according to the "regular numbering scheme" (Featherstone & Orin (2008)), which induces a parent/child relationship such that each node has a unique input edge and multiple output edges. As a convention, it is here assumed that each joint has an associated reference frame with the z-axis aligned with the rotation axis; this frame will be denoted $\langle e_{i,j} \rangle$. In kinematics, an edge $e_{i,j}$ from v_i to v_j represents the fact that $\langle e_{i,j} \rangle$ is fixed in the i-th link. In dynamics, $e_{i,j}$ represents the fact that the dynamic equations will compute (and make use of) $w_{i,j}$, i.e. the wrench that the i-th link exerts on the j-th link, and not the equal and opposite reaction $-w_{i,j}$, i.e. the wrench that the j-th link exerts on the i-th link (further details in Section 3.4). In

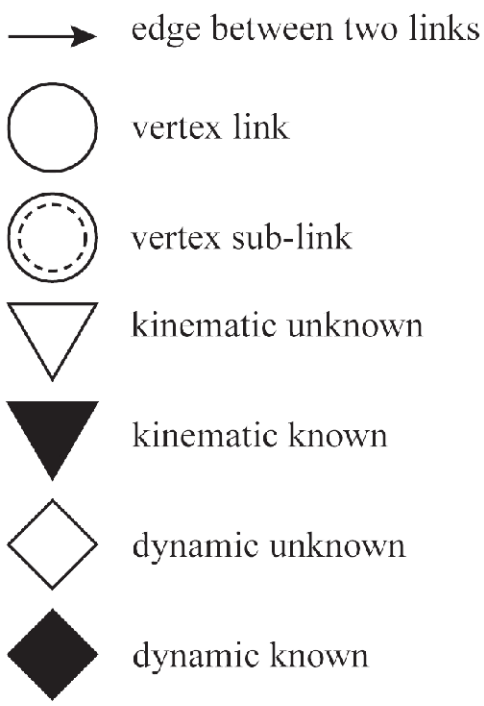


Figure 3.6: The notation introduced to represent nodes, sub-nodes, edges, known and unknown kinematic and dynamic variables in graphs.

order to simplify the computations of the inverse dynamics on the graph (as will be shown in Section 3.4), kinematic and dynamic measurements have been explicitly represented through additional types of nodes (see Figure 3.6). In particular, the graph representation has been enhanced with a new set of graphical symbols: a triangle to represent kinematic quantities (i.e. velocities and acceleration of links), and a rhombus for wrenches (i.e. force sensors measurements within a link), as shown in Figure 3.6. Moreover these symbols have been further divided into *known* quantities to represent sensors measurements, and *unknown* to indicate the quantities to be computed.

3.3.2 Kinematics

Kinematic variables can in general be measured by means of gyroscopes, accelerometers, or simply inertial sensors. When attached on link i-th, these sensors provide angular and linear velocities and accelerations (ω, $\dot{\omega}$, $\dot{p}$ and $\ddot{p}$) at the specific location where the sensor is located. We can represent these measurement in the graph with a *black triangle* (▼) and an additional edge from the proper link where the sensor is attached to the triangle. It is to be underlined here that, according to the kinematic convention previously mentioned, an edge $e_{i,j}$ is fixed on the i-th link. Therefore a sensor fixed in the i-th link, will be represented by $e_{i,s}$, i.e. an edge from the link to the sensor (see Figure 3.7(a) and Figure 3.7(b)). Also in this case, the reference frame associated to the edge corresponds to the reference frame of the sensor. Similarly, an unknown kinematic variable is represented with a *white triangle* ($\triangledown$) with an associated edge going from the link (where the unknown kinematic variable is attached) to the triangle. The reference frame associated to the edge will determine the characteristics of the retrieved unknown kinematic variables as it will be clear in Section 3.4.

3.3.3 Dynamics

Similarly, we introduce two new types of nodes with a rhomboidal shape (see Fig. 3.6): *black rhombi* (⧫) to represent known (i.e. measured) wrenches, *white rhombi* ($\Diamond$) to represent unknown wrenches which need to be computed. The reference frame associated to the edge will be the location of the applied or unknown wrench. Remarkably, there is not a fixed rule to determine the orientation of the edge connecting the

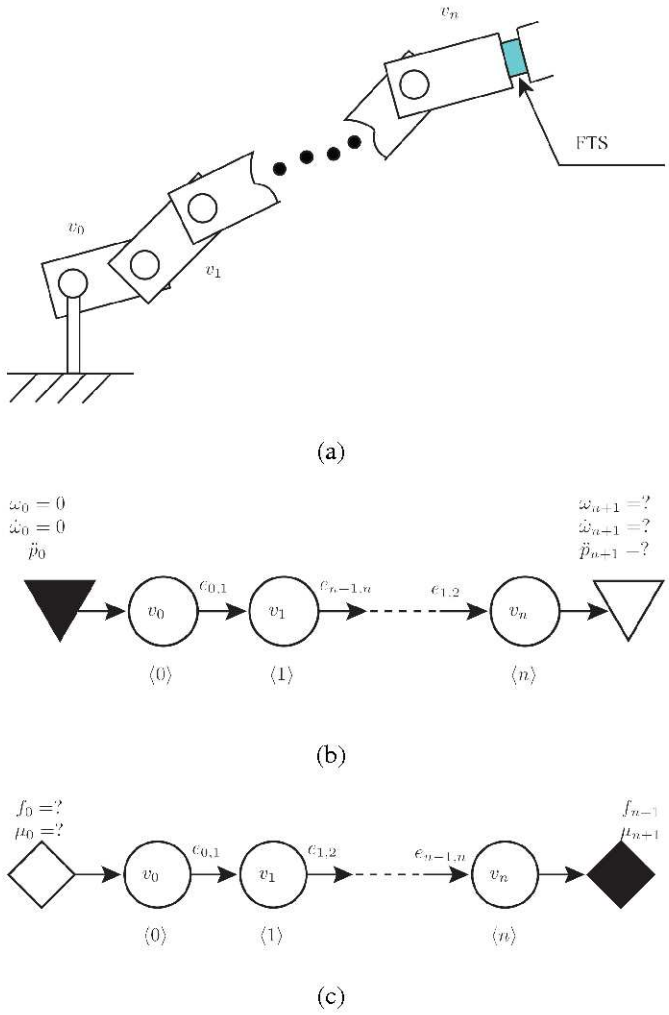


Figure 3.7: **Top**(3.7(a)): sketch of a serial robotic structure with a FTS placed at the end-effector; kinematic quantities are known at the base of the robot, where $\omega_0 = \dot{\omega}_0 = 0$ and $\ddot{p}_0 = \mathrm{g}$. **Center**(3.7(b)): graph representing the kinematic EOG of Figure 3.7(a). **Bottom**(3.7(c)): graph representing the dynamic EOG of Figure 3.7(a).

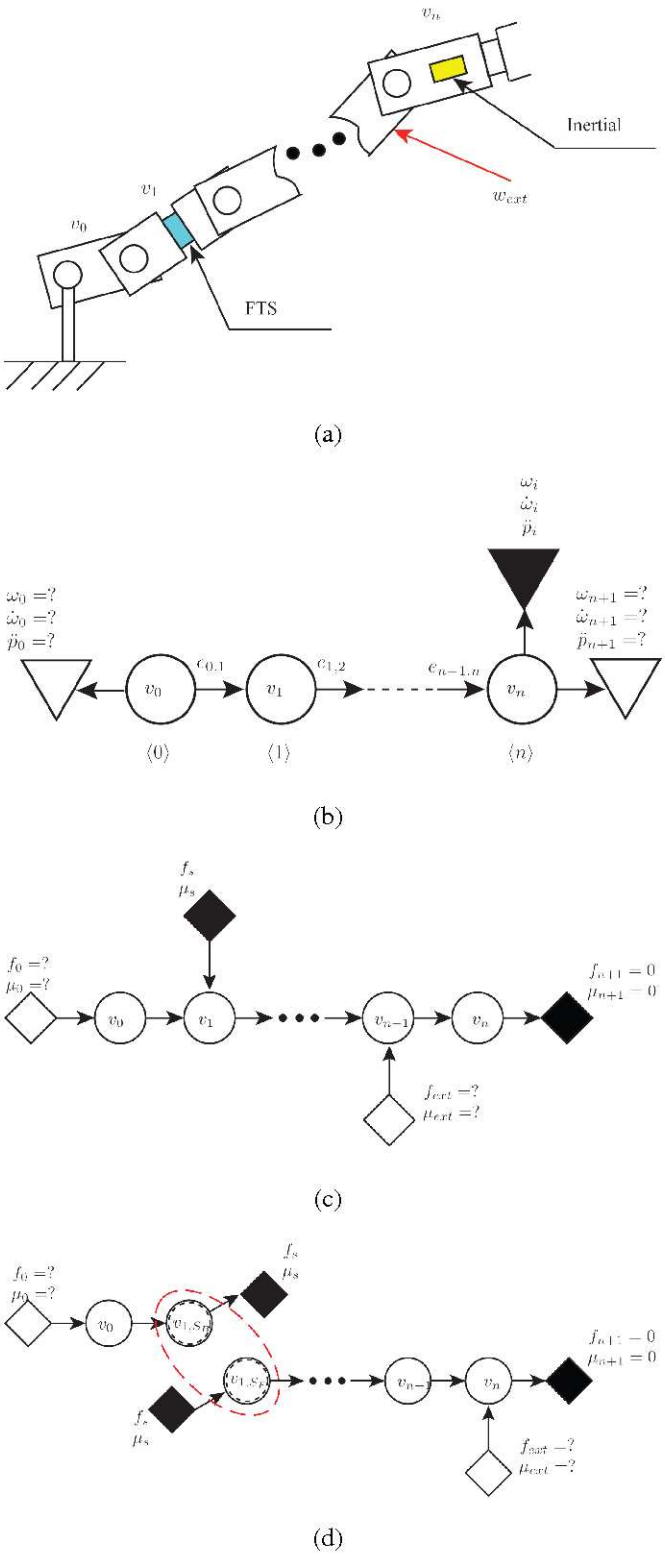
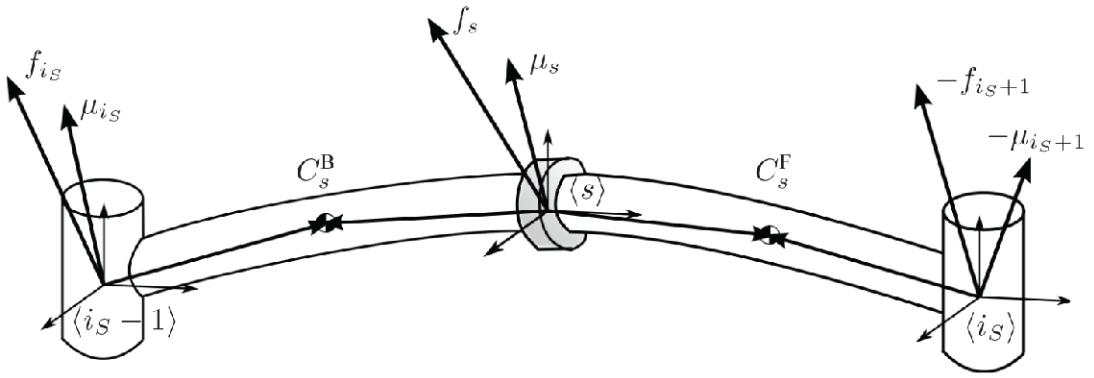


Figure 3.8: **Top**(3.8(a)):sketch of a serial robotic structure with a FTS placed on a proximal link; an inertial sensor is placed on the n-th link. **Bottom**(3.8(b)): graph representing the kinematic EOG of Figure 3.8(a). **Bottom**(3.8(c)): graph representing the dynamic EOG of Figure 3.8(a), before the division into two subchains. **Bottom**(3.8(d)): graph representing the two dynamic EOGs of Figure 3.8(a), consequence of the division of the link mounting the FTS into two sublinks; In this case, two unknown wrenches can be determined, one for each subchain.



(a)

Figure 3.9: A representation of an FTS within the i_S-th link. Note that the sensor divides the link into two sub-links, each with its own dynamical properties. In particular, it is evident that the center of mass (COM) of the original link, C_{i_S}, differs from $C_s^{\mathrm{F}}, C_s^{\mathrm{B}}$, i.e. the COM of the two "sub-links".

rhombi to the graph: according to our convention for representing the wrenches, the edge can be either directed from the rhombus to the link or *vice versa* depending on the variable we are interested in representing (i.e. the wrench from the link to the external environment or the equal and opposite wrench from the environment to the link). It is important to point out that, whereas the position of ♦ is static within the graph (because sensors are fixed in the manipulator), the location of ◊ instead can be dynamic (contact point locations are dynamically detected by the distributed tactile sensor). If a contact moves along a chain, the graph is accordingly modified. This rule shows a big benefit of the EOG, which dynamically adapts in response to the location of the unknown external wrenches. Within this representation, embedded FTS can be inserted by "cutting" the manipulator chain where the FTS is located and creating two virtual "sub-links" from the link hosting physically the sensor. The EOG is then split into two sub-graphs, where black rhombi (♦, i.e. known wrenches representing the FTS measures, one per graph) are introduced and attached to the sub-links. In practice, suppose that an FTS is placed in the i_S-th link (see Fig. 3.8(a)). Let $\langle s \rangle$ be the frame associated to the sensor. The sensor virtually divides link i_S into two "sub-links" (hereafter denoted forward v_{i,S_F} and the backward v_{i,S_B} sub-links, as shown in Figure 3.8(d)). The sensor therefore measures the wrench exchanged between the "forward" and the "backward" sub-links (this will be represented by two rhomboidal nodes). Under these

considerations, the FTS within a link is represented by splitting the node associated to the link into two sub-nodes (with suitable dynamical properties, see Fig. 3.8(d)). Two known wrenches in the form of black rhombi are then attached to the sub-nodes, with suitable edges whose associated reference frame is $\langle s \rangle$ for both edges.

3.4 Exploiting the RNEA for EOG

The graphical representation proposed in Section 3.3 can be adopted to represent the flow of information within kinematic chains, which are necessary to perform the subsequent computation of the internal dynamics of a (floating) kinematic chain provided with sufficient tactile, proprioceptive, haptic and inertial sensors. In particular, in this section we describe how to compute both kinematic and dynamic variables, associated to the edges of the graphical representation, for the general case of (floating) multiple branches kinematic chains. The method shown hereafter adopts the Newton-Euler formulas, but any other recursive formulation might be employed to perform the computation.

A first recursion on the graph (pre-order traversal) will compute the linear acceleration ($\ddot{p}$) and the angular velocity and acceleration (ω, $\dot{\omega}$) for each of the reference frames associated to the edges of the graph. This procedure practically propagates the information coming from a single inertial sensor to the entire kinematic chain. At each step, the values of ($\ddot{p}$, ω, $\dot{\omega}$) for a given link are propagated to neighbor links by exploiting the encoder measurements and a kinematic model of the chain. A second recursion (post-order traversal) will compute all the (internal and external) wrenches acting on the chain at the reference frames associated with all the edges in the graph. In this case, Newton-Euler equations are exploited to propagate force information along the chain. At each step, all but one wrench acting on a link are assumed to be known and the remaining unknown wrench is computed exploiting a dynamic model of the link and the output from the kinematic recursion.

When computing wrenches, it cannot in fact be performed the computation the computation of more than one unknown for each subchain. Practically speaking, each chain represents one 6-dimentional equation, where at least one 6-D unknown can be determined. If more than one unknown is present on a subchain, an infinite of solu-

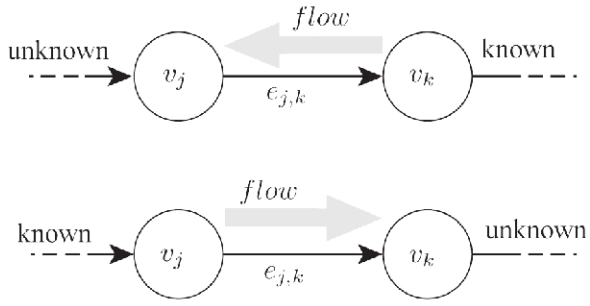


Figure 3.10: The basic operation for propagating information across an EOG. Given v_j we assume to know ω_j, $\dot{\omega}_j$, $\ddot{p}_j$. This information can then be propagated to all the connected nodes. If v_k is connected to v_j by $e_{j,k}$ (i.e. the edge is directed from v_j to v_k) then we can compute ω_k, $\dot{\omega}_k$, $\ddot{p}_k$ using (3.4.1) (just replace $i+1$ with k). If v_k is connected to v_j by $e_{k,j}$ (i.e. the edge is directed from v_k to v_j) then we can compute ω_k, $\dot{\omega}_k$, $\ddot{p}_k$ using (3.4.1) (just replace $i-1$ with k). Similar considerations can be done for dynamic variables.

tion can be found, and thus the method cannot be applied without introducing some assumptions on the contacts.

3.4.1 Kinematics

We here describe the basic equations for propagating the kinematic information within the graph. The proposed notation might seem a little bit too general, especially if compared with the classical computations where the major simplification is the assumption that kinematics are propagated in the kinematic tree along a constant path. In our case instead, we are interested in a formulation capable of exploiting multiple (dynamically inserted) inertia sensors to propagate the kinematic information from the sensors to the surrounding links. Therefore the flow of kinematics cannot be predefined but needs to be dynamically adapted to the current structure of the EOG. The basic step here described consists in propagating the kinematic information associated to an edge connected to a node v to all the other edges connected to v. As usual, for each edge i we consider the associated reference frame $\langle i \rangle$. Referring to Fig. 3.11(a)-3.11(c) we assume that knowing the linear acceleration ($\ddot{p}_j$) and the angular velocity and acceleration (ω_j, $\dot{\omega}_j$) of the reference frame $\langle j \rangle$ we want to compute the same quantities for the frame $\langle i \rangle$ sharing with $\langle j \rangle$ a common node v. Fig. 3.11(a) represents the case where the edge i exits v but the edge j enters v; recalling the kinematic meaning of the edge

directions, the sketch in Fig. 3.11(a) represents a situation where $\langle i \rangle$ is attached to v while $\langle j \rangle$ is rotated by the joint angle θ_j around z_j. The situation is exactly the one we have in the classical Denavit-Hartenberg forward kinematic description and therefore we have[1] (see Sciavicco & Siciliano (2005a)):

$$\begin{aligned} \omega_i &= \omega_j + \dot{\theta}_j z_j, \\ \dot{\omega}_i &= \dot{\omega}_j + \ddot{\theta}_j z_j + \dot{\theta}_j \omega_j \times z_j, \\ \ddot{p}_i &= \ddot{p}_j + \dot{\omega}_i \times r_{j,i} + \omega_i \times (\omega_i \times r_{j,i}), \end{aligned} \tag{3.4.1}$$

where z_j and θ_j indicate the rotational axis and the angular position of the joint associated to the edge j. Similarly, Fig. 3.11(b) represents the case where the edge i enters v but the edge j exits the node; therefore Fig. 3.11(b) represents a situation where $\langle j \rangle$ is attached to v while $\langle i \rangle$ is rotated by the joint angle θ_i. The situation is exactly the opposite encountered in classical Denavit-Hartenberg so that we have:

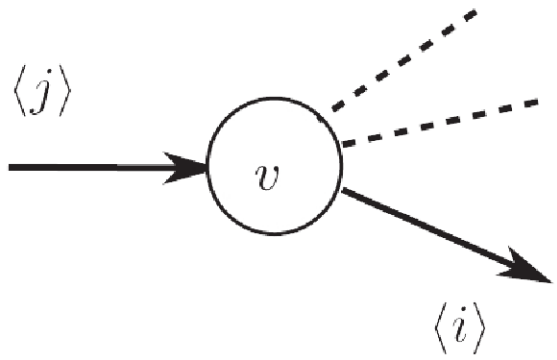
$$\begin{aligned} \omega_i &= \omega_j - \dot{\theta}_i z_i, \\ \dot{\omega}_i &= \dot{\omega}_j - \ddot{\theta}_i z_i - \dot{\theta}_i \omega_j \times z_i, \\ \ddot{p}_i &= \ddot{p}_j - \dot{\omega}_j \times r_{i,j} - \omega_j \times (\omega_j \times r_{i,j}). \end{aligned} \tag{3.4.2}$$

Finally, Fig. 3.11(c) represents the case where both $\langle i \rangle$ and $\langle j \rangle$ are attached to the link represented by v. In this case, continuity formulas are obtained putting $\dot{\theta}_i = 0$ and $\ddot{\theta}_i = 0$ in Equation 3.4.1 (or equivalently Equation 3.4.2):

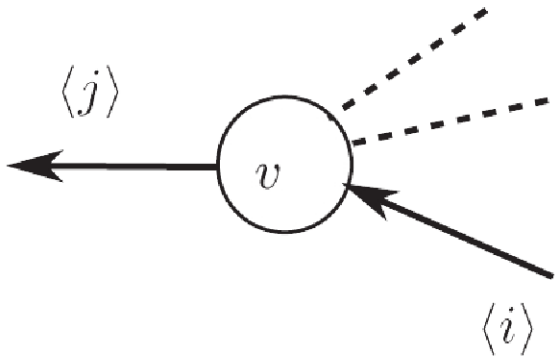
$$\begin{aligned} \omega_i &= \omega_j, \\ \dot{\omega}_i &= \dot{\omega}_j, \\ \ddot{p}_i &= \ddot{p}_j + \dot{\omega}_i \times r_{j,i} + \omega_i \times (\omega_i \times r_{j,i}). \end{aligned} \tag{3.4.3}$$

These rules can be used to propagate kinematic information across different edges

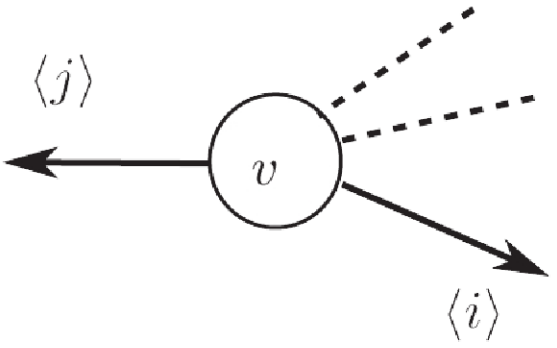
[1] In the classical recursive kinematic computation (as in Sciavicco & Siciliano (2005a)) there is a one-to-one correspondence between links and joints (see Figure 3.3) thus resulting in a kinematic equations slightly different from Equation 3.4.1. Classically, the i-th link has two joints and associated reference frames $\langle i \rangle$ and $\langle i-1 \rangle$, respectively. Only $\langle i \rangle$ is attached to the i-th link while $\langle i-1 \rangle$ is attached to the link $i-1$. The rotation between these two links is around the z-axis of $\langle i-1 \rangle$ by an angle which is denoted θ_i and therefore the analogous of Equation 3.4.1 in Sciavicco & Siciliano (2005a) refer to $\dot{\theta}_i$ in place of $\dot{\theta}_j$ and z_{i-1} in place of z_i. In our notation, we get rid of this common labeling for joints and links by explicitly distinguishing the link represented with the node v and the attached joints represented with the edges i, j, ... and associated frames $\langle i \rangle$, $\langle j \rangle$, ... whose axes are therefore z_i, z_j, ... with associated angles θ_i, θ_j.



(a)



(b)



(c)

Figure 3.11: The three cases accounting for the exchange of kinematic information.

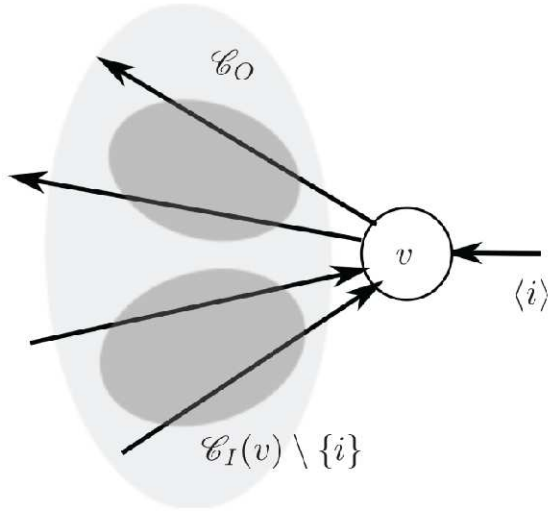
connected to the same node. The only situation which cannot be solved is the one where all edges enter the node v, i.e. none of the associated reference frames is fixed to the link v. We can handle these cases *a posteriori* by defining a new arbitrary reference frame $\langle v \rangle$ attached to the link. In our formalism, this is achieved by adding a kinematic unknown ($\triangledown$) and an edge from v to $\triangledown$ with associated frame $\langle v \rangle$. It is remarkable here that, if the edge directions are chosen according to a "regular numbering scheme" as proposed in Section 3.3.1, each edge will have a unique ingoing edge and multiple outgoing edges.

The only nodes with no outgoing edges will be the ones corresponding to the leaves of the kinematic tree (typically the end-effectors). For these nodes, we will add a kinematic unknown ($\triangledown$) and an edge from v to $\triangledown$ with associated frame $\langle v \rangle$ (typically the end-effector reference frame of the classical Denavit-Hartenberg notation).

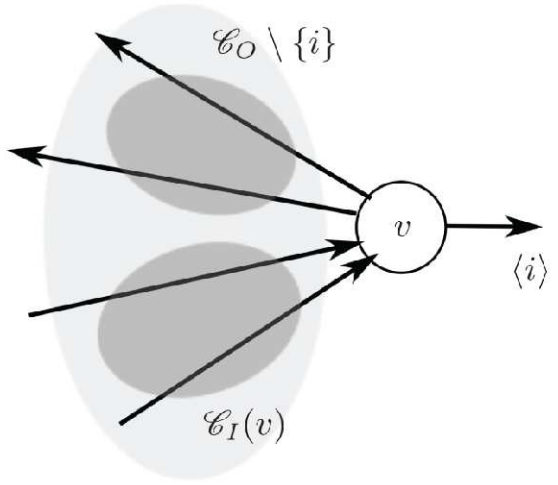
3.4.2 Dynamics

We here describe the basic equations for propagating the dynamic information within the graph. Also in this case, the flow of dynamical information cannot be predefined because the graph structure continuously changes according to the position of the applied external wrenches (as detected by the distributed tactile sensor). The basic step proposed in this section assumes that all but one wrench acting on a link are known and the remaining unknown wrench is computed by using the Newton-Euler equations. Using the graph representation, a node v with all its edges represents a link with all its joints. As proposed in Section 3.3.1, at each edge $e_{u,v}$, we can associate the wrench $w_{e_{u,v}}$ that u exerts on v. At each edge $e_{v,u}$ we can associate the wrench $w_{e_{v,u}}$ that v exerts on u. The Newton-Euler equations for the link v can therefore be written as follows:

$$\begin{aligned} \sum_{e_I \in \mathscr{C}_I(v)} f_{e_I} - \sum_{e_O \in \mathscr{C}_O(v)} f_{e_O} &= m_v \ddot{p}_{C_v}, \\ \sum_{e_I \in \mathscr{C}_I(v)} (\mu_{e_I} + f_{e_I} \times r_{e_I,C_v}) & \\ - \sum_{e_O \in \mathscr{C}_O(v)} (\mu_{e_O} + f_{e_O} \times r_{e_O,C_v}) &= \bar{I}_i \dot{\omega}_i + \omega_i \times (\bar{I}_i \omega_i), \end{aligned} \tag{3.4.4}$$



(a)



(b)

Figure 3.12: The two cases accounting for the exchange of dynamic information.

where[1]:

$$\ddot{p}_{C_v} = \ddot{p}_i + \dot{\omega}_i \times r_{i,C_v} + \omega_i \times (\omega_i \times r_{i,C_v}), \tag{3.4.5}$$

and where $\mathscr{C}_I(v)$ is the set of ingoing edges, $\mathscr{C}_O(v)$ is the set of outgoing edges and where the index i refers to any edge in $\mathscr{C}_O(v)$ (necessarily non-empty in consideration of what we discussed in Section 3.4.1). In other terms, recalling the kinematic meaning of outgoing edges, i is an edge associated with any of the arbitrary reference frames $\langle i \rangle$ fixed with respect to the link v. As anticipated, Equation 3.4.4 can be used to propagate the dynamic information across the graph. Assuming that all but one wrench acting on a link are known, the remaining unknown wrench can be computed with Equation 3.4.4. Let us denote with i the edge associated with the unknown wrench. If $i \in \mathscr{C}_I(v)$, then the situation is the one represented in Fig. 3.12(a) and we have:

$$\begin{aligned} f_i &= -\sum_{\substack{e_I \in \mathscr{C}_I(v) \\ e_I \neq i}} f_{e_I} + \sum_{e_O \in \mathscr{C}_O(v)} f_{e_O} + m_v \ddot{p}_{C_v}, \\ \mu_i &= -f_i \times r_{i,C_v} - \sum_{\substack{e_I \in \mathscr{C}_I(v) \\ e_I \neq i}} (\mu_{e_I} + f_{e_I} \times r_{e_I,C_v}) \\ &\quad + \sum_{e_O \in \mathscr{C}_O(v)} (\mu_{e_O} + f_{e_O} \times r_{e_O,C_v}) + \bar{I}_i \dot{\omega}_i + \omega_i \times (\bar{I}_i \omega_i). \end{aligned} \tag{3.4.6}$$

If $i \in \mathscr{C}_O(v)$, then the situation is the one represented in Fig. 3.12(b) and we have:

$$\begin{aligned} f_i &= \sum_{e_I \in \mathscr{C}_I(v)} f_{e_I} - \sum_{\substack{e_O \in \mathscr{C}_O(v) \\ e_O \neq i}} f_{e_O} - m_v \ddot{p}_{C_v}, \\ \mu_i &= -f_i \times r_{i,C_v} + \sum_{e_I \in \mathscr{C}_I(v)} (\mu_{e_I} + f_{e_I} \times r_{e_I,C_v}) \\ &\quad - \sum_{\substack{e_O \in \mathscr{C}_O(v) \\ e_O \neq i}} (\mu_{e_O} + f_{e_O} \times r_{e_O,C_v}) - \bar{I}_i \dot{\omega}_i - \omega_i \times (\bar{I}_i \omega_i). \end{aligned} \tag{3.4.7}$$

Note that, with reference to Equation 3.4.6-3.4.7, if only one edge is connected to the generic node v, then $\mathscr{C}_I(v) \cup \mathscr{C}_O(v) = \{i\}$. Hence, the sums $\sum f_k$, $\sum(\mu_k + f_k \times r_{k,C_v})$ (being k the generic index for the edge) are null and the equations are basically simpler.

[1] With slight abuse of notation we indicated with $r_{\star,C_v}$ the vector connecting the generic frame $\langle \star \rangle$ to the one placed on the center of mass C_v of the v-th link.

This case is peculiar, and its significance will be clear later on when the solution of the EOG is discussed in detail.

4

Building EOG for Computing Dynamics and External Wrenches of the iCub Robot

Chapter 3 has shown a dynamic formulation for the computation of externally applied wrenches along kinematic trees. The method makes use of a graphical formulation which employs graph theories for performing the computation of both kinematic quantities (i.e. angular velocities of the center of mass of links, but also linear and angular acceleration), and dynamic (internal forces and moment on the connection elements between the links, but also externally applied wrenches). A classical graph formulation has been adapted to the case in which distributed inertial sensors and force/torque sensors are embedded within the links. It has been shown that, given N FTSs, $N + 1$ external wrenches can be computed, one for each subchain resulting from the employment of FTS measurements in a link. With reference to the classical approaches(see Featherstone & Orin (2008)), two main modification have been added: new nodes have been added to embed sensor measurements in the computation, and edge orientation is exploited to define the kinematic representation of reference frames on a link.
The chapter shows the application of the method presented in Chapter 3, applied to the dynamic formulation of the iCub humanoid robot (see Chapter 2). It will be briefly summarized in Section 4.1 the steps required for building the kinematic and dynamic model of robotic mechanisms. The application of the method will be shown for different simple robotic structures, and will then be applied to the modeling of the iCub

robot. Some results will be reported in Section 4.3.2, where the validation of the inverse dynamic algorithm and the model is addressed through the comparison of computed and measured internal wrenches at the sensor reference frame. Two other experiments show the comparison of externally applied forces measured with an external FTS, with virtual FTS exploiting the proposed method. Also a comparison of external torques on the joints will be performed.

4.1 Summary of the EOG Definition on Robotic Structure: Case Studies

Section 3.4.1 and Section 3.4.2 presented the basic steps for propagating kinematic and dynamic information across a graph representing a kinematic tree. This flow of information can be used to determine unknown (actually non directly measured) quantities along the kinematic tree. In particular, it is possible to have a *virtual measurement* of both kinematic quantities (ω_i, $\dot{\omega}_i$ and $\ddot{p}_i$ for each link i) and dynamic (f_i and μ_i). In particular, the discussion addressed in this chapter will mainly focus on the *virtual force/torque sensor* method, exploiting the dynamical model of the robotic structure, to have an estimation of both internal wrenches that the links reciprocally exchange on joint connections, and externally applied generalized forces at any location of the kinematic structure. For the kinematic quantities it will be addressed only the case in which one only Inertial sensor is present on the structure, as it is sufficient to have an estimation of the kinematic quantities along all the structure. Case studies of robotic mechanisms which employ one or more distributed FTSs, instead, will be analyzed (e.g. the iCub robot). It is remarkable to underline that, given N FTS distributed along the kinematic chain, $N + 1$ *virtual measurement* of externally applied forces (one for each sub-chain) can be performed. Moreover, in case the point of application of the external wrench is known (practically speaking, if an artificial skin covers the robot structure), the unknown leaves can be dynamically moved along the graphs, thus resulting in a non fixed path followed by the information flow during the computation, as already mentioned in Section 3.3.

This section summarizes the basic steps to compute the whole-body dynamics, with specific attention at getting estimates for the externally applied wrenches (denoted with

$\Diamond$).
Hereafter follows the steps for the definition of the graph structure:

1. Create the graph representing the kinematic tree; define a node for each link and an edge for each joint connecting two links. The edge orientation is arbitrary and in particular it can be defined according to a "regular numbering scheme".

2. For each inertial sensor (measuring the linear acceleration and the angular velocity and acceleration) insert a *black triangle* (▼) and an edge from the node v to the triangle, where v represents the link to which the sensor is attached. Associate to the edge the reference frame $\langle s \rangle$ corresponding to the sensor frame.

3. For any node v with only ingoing edges, add a *white triangle* ($\triangledown$) and an edge from v to the triangle. Associate to the edge an arbitrary reference frame $\langle v \rangle$.

At point 2, it should be noticed that kinematic chains are often grounded and therefore there exists a base link with null angular kinematics, $\omega = [0, 0, 0]^\top$, $\dot{\omega} = [0, 0, 0]^\top$ and gravitational linear acceleration $\ddot{p} = g$, being g the vector representing the gravity force (as an example, $g = [0, 0, -9.81]$ if the base frame has the z-axis oriented as the gravity component. Any other condition is obviously allowed, and depend on the orientation of the base frame).
These steps define the kinematic EOG which can be used to compute the kinematics of the entire chain. Specifically, if this graph contains a single inertial sensor (represented by a ▼ node), the associated measurements can be used to compute the linear acceleration and angular acceleration and velocity for all the edges of the graph. Computations can be performed following the procedure in Alg. 1, that is a *pre-order* traversal of the tree with elementary operations defined by Eq. 3.4.1, Eq. 3.4.2 or Eq. 3.4.3. If multiple ▼ nodes (i.e. inertial sensors) are present in the graph, each path between two of these nodes corresponds to a set of three equations containing the measurements:one for the linear accelerations, one for the angular velocity and one for the angular accelerations. These equations can be used to refine the sensor measurements or to give better estimates of the joint velocities and accelerations (typically derived numerically from the encoders and therefore often noisy), as reported in Appendix 6.3.3.
Figure 4.1 shows an example of a floating base multiple branches mechanism which

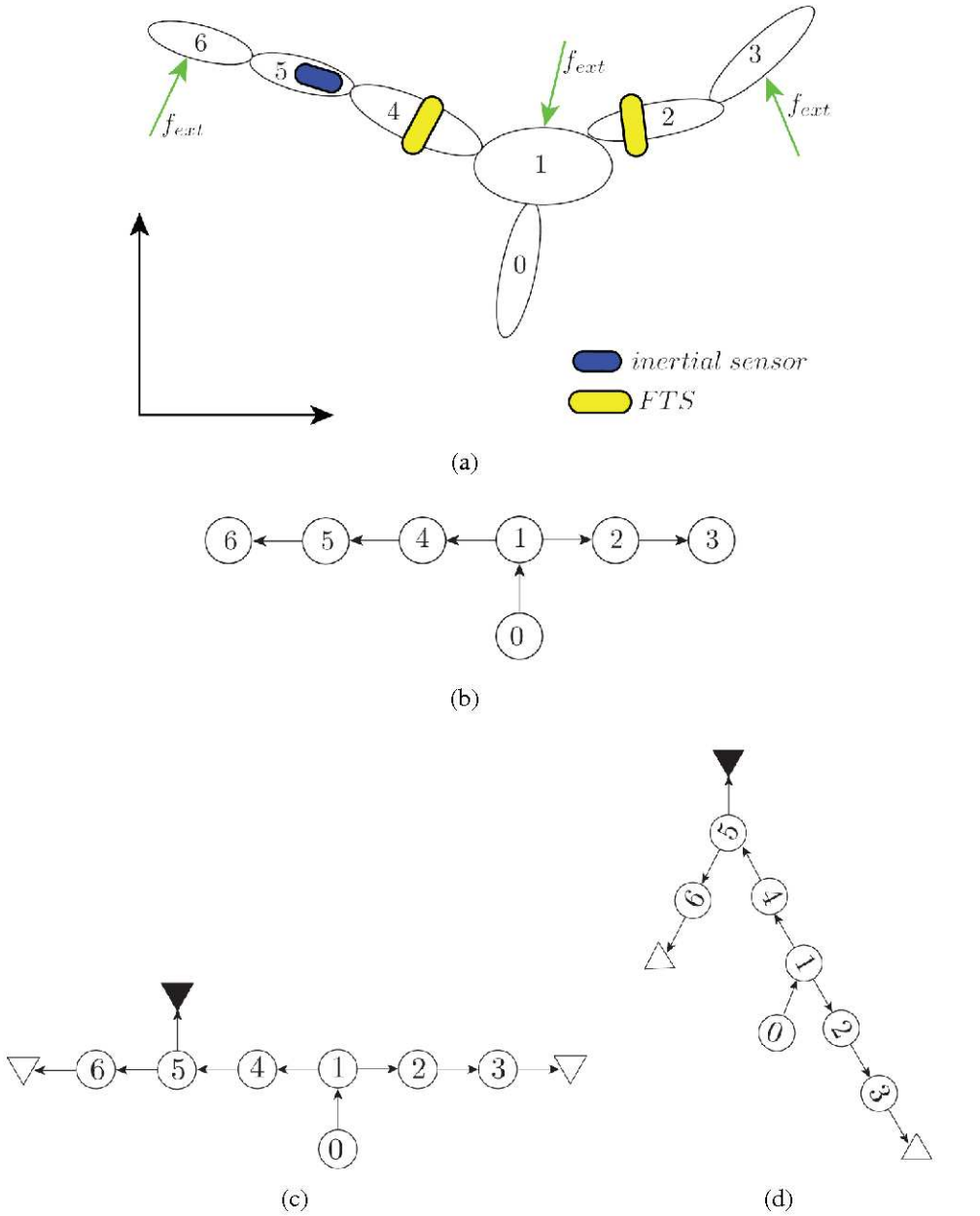


Figure 4.1: Top (4.1(a)): an example of generic floating, multiple branches kinematic chain subject to external forces; The chain mounts two FTSs and one inertial sensors. **Center** (4.1(b)): Classical representation of the graph structure; The direction of the edges represent the kinematic conventions for the definition of the kinematic of the chain. **Bottom left** (4.1(c)): enhanced graph representation of the kinematic of the chain. **Bottom right** (4.1(d)): rearrangement of the kinematic EOG to underline the way to visit the graph, actually a pre-order traversal; starting from the root node, the computation is performed in the following order: 5, 6, unknown leaf, 4, 1, 0, 2, 3, unknown leaf.

mount one inertial sensor and multiple FTSs, represented in Figure 4.1(a) . The kinematic of this chain has been defined starting from link 0, as shown in Figure 4.1(b). The enhanced graph associated to this mechanism takes the form of the graph shown in Figure 4.1(c), which can be rearranged as in Figure 4.1(d). It is clear that the way to visit the graph is a pre-order traversal. It is remarkable here that the visit order is not related to the edge direction, since the latter only affects the recursive equations that must be used to propagate the variables, as was already shown with Fig. 3.10. Once velocities and accelerations have been computed for all edges, a new series of steps needs to be performed on the EOG to obtain the dynamic enhanced subgraphs.

1. For each FTS embedded in the link v cut the graph into two subgraphs according to the procedure shown in Figure 3.8(d). Divide v into two nodes v_{i,S_B} and v_{i,S_F} representing the sub-links (with suitable dynamic properties); define two *black rhombi* (⧫) and add two edges from the rhombi to the nodes. Associate to both the edges the same reference frame $\langle s \rangle$ corresponding to the sensor frame.

2. If there are other known wrenches acting on a link (e.g. sensors attached at the end-effector), insert a *black rhombus* (⧫) and an edge from the rhombus to v, where v represents the link to which the wrench is applied. Associate to the edge the reference frame $\langle s \rangle$ corresponding point where the external wrench is applied.

3. If the distributed tactile sensor is detecting externally applied wrenches, insert a *white rhombus* (◇) for each externally applied unknown wrench. Add an edge connecting the rhombus with v, where v represents the link to which the wrench is applied. The edge orientation is arbitrary depending on the wrench to be computed (i.e. the wrench from the link to the external environment or the equal and opposite wrench from the environment to the link). Associate to the edge the reference frame $\langle c \rangle$ corresponding to the location where the external wrench is applied.

After these steps have been performed, the dynamic enhanced subgraphs are obtained, each of which can be considered independently. Wrenches can be propagated to the unknown nodes (◇) if and only if there exists a unique unknown for each sub-graph.

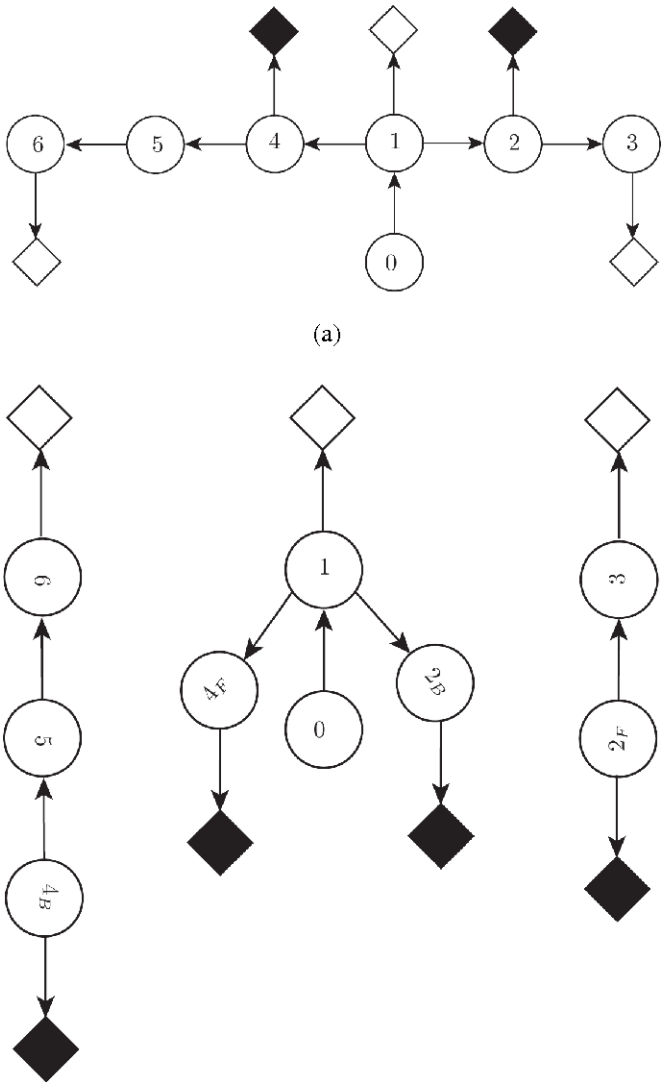


Figure 4.2: The figure refers to the floating, multiple branches kinematic chain subject to external forces shown in Figure 4.1(a).**Top** (4.2(a)): enhanced graph representation of the kinematic of the chain. Two known quantities are present (♦), which represent the FTSs, together with the three unknowns representing the external forces (◊); nodes 4 and 2 will be divided into two sub-nodes, giving origin to three sub-chains. **Bottom** (4.2(b)): rearrangement of the dynamic EOGs to underline the way to visit the graph, actually a post-order traversal; starting from the leaf nodes, the computation is performed in the following order: (**case 1**(**left**)) 4_B, 5, 6, unknown leaf, (**case 2**(**center**)) 4_F, 0, 2_B, 1, unknown leaf, 2_F, 3, unknown leaf.

If this is the case, then for each unknown we can define a tree with the node $\Diamond$ as root. Wrenches can be propagated from the leaves to the root following the procedure in Alg. 2, which is basically a *post-order* traversal of a tree(see Cormen *et al.* (2002)) with elementary operations defined by Eq. 3.4.6 or Eq. 3.4.7. Figure 4.2 show the graph associated to the mechanism of Figure 4.1(a). In this case, since there are 2 FTSs in the chain, which correspond to 2 known quantities in the enhanced graph (see Figure 4.2(a)), an overall of 3 external forces can be determined, which correspond to 3 unknown nodes in the enhanced graph. To perform the computation, the graph should thus be splitted into 3 subgraphs, as in Figure 4.2(b), each of which allows to detect one external wrench $\Diamond$. If there is no $\Diamond$ node in a subgraph (i.e. no external forces are acting on the subgraph), then the *post-order* traversal of this graph produces two equations (one for forces and the other for wrenches) with no unknowns[1]. These equations can be used to estimate on-line the dynamical parameters of the corresponding kinematic sub-tree exploiting the linearity of these parameters in the equations (see Sciavicco & Siciliano (2005a)).
Remarkably, in the considered cases (one $\Diamond$ per subgraph at maximum) each edge in the subgraph is visited during the *post-order* traversal. As a result, all internal wrenches are computed and therefore a complete characterization of the whole-body dynamics is retrieved.
It is now clear that, as a consequence of what has been shown, given N FTS distributed on a chain, $N+1$ sub-graphs are produced and therefore a maximum of $N+1$ external wrenches can be estimated (one for each sub-graph).

4.2 Performing the Computation

With reference to figure Figure 4.2, in order to clarify how to exploit computation of wrenches on an EOG, different situation are hereafter reported. Once again, the reader should note that the employment of the Denavit-Hartenberg notation for the definition of the kinematic structure of the links, and the RNEA for the definition of the dynamics of the system, are not mandatory. Custom choices can be adopted.

[1]Practically, these equations can be obtained by defining an arbitrary $\Diamond$ connected to an arbitrary node. A *post-order* traversal of the graph with $\Diamond$ as root determines the equations by simply assuming that the wrench associated to the edge connected to $\Diamond$ is null.

Algorithm 1 Solution of kinematic EOG exploiting a tree

Require: EOG, $\omega_0, \dot{\omega}_0, \ddot{p}_0$
Ensure: $\omega_i, \dot{\omega}_i, \ddot{p}_i, \forall v_i$
1: Attach a node ▼ for every kinematic source (e.g. inertial sensor)
2: Set $\omega_0, \dot{\omega}_0, \ddot{p}_0$ in ▼
3: Re-arrange the graph with a ▼ as the root of a tree
4: *KinVisit*(EOG,v_{root})

KinVisit(EOG,v_i)
1: Compute $\omega_i, \dot{\omega}_i, \ddot{p}_i$ with Eq. 3.4.1 or 3.4.2 or 3.4.3 according to direction of the edges i, j connected to v
2: **for all** child v_k of v_i **do**
3: *KinVisit*(EOG,v_k)
4: **end for**

Algorithm 2 Solution of dynamic EOG exploiting a tree

Require: EOG, $w_s \forall$ FTS
Ensure: $w_i, \forall v_i$
1: For every FTS, attach a node ♦ to the corresponding link
2: Set w_s in each ♦
3: For each ♦, split the graph and create two sub-graphs (see text for details)
4: Attach a node ◊ to each link where a contact is detected: if there is no contact in a subgraph, choose an arbitrary position and attach a fictitious ◊[1]
5: Re-arrange each sub-graph with a ◊ as the root of a tree
6: **for all** subgraph **do**
7: *DynVisit*(EOG,v_{root})
8: **end for**

DynVisit(EOG,v)
1: **if** v has children **then**
2: **for all** child $e_{v,h} \in \mathscr{C}(v), e_{v,h} \neq i$ **do**
3: $w_{e_{v,h}} =$ *DynVisit*(EOG,h)
4: **end for**
5: **end if**
6: Compute w_i with Eq. 3.4.6 or Eq. 3.4.7 according to the direction of the edges

When passing from one vertex to the other, Eq. 3.4.6 or Eq. 3.4.7 is used. This equation provides the computation of both internal and external forces, depending on the definition of known and unknown variables on the graph. With reference to figure Fig. 4.2(b), when the flow of the information is along the same direction of the edge, f_i of Eq. 3.4.6 or Eq. 3.4.7 is to be computed. One of the forces among the $\sum f_k$ otherwise, depending on which of the k links the unknown is located.

4.2.1 Single-Branched Open Chain

In the case presented in Fig. 4.2(b), left and right side, a single open chain exists, for the links in between the sensors and the final links. In this situation, when the flow of the information is along the same direction of the edge, f_i of Eq. 3.4.6; of Eq. 3.4.7 otherwise. Next sections show the case which demonstrate the generality of the EOG method also for open, multi-branched kinematic chains.

4.2.2 Multiple-Branched Nodes and External Forces

With respect to Fig. 4.2(b), we point out that the unknown $\lozenge$ attached to the base is used if a contact is detected on that link (e.g. if the artificial tactile skin reveals a contact at the base). In absence of contact (as in the case of node 0), the node $\lozenge$ is not needed. More in general, if f_{ext} is not present, it is possible to write the recursive equations as a compact set, where all the dynamic variables are known: this formulation can be exploited to obtain, for example, a better estimate of the rigid-body model parameters, e.g. links mass.

On the other side, external forces may be acting in other locations different from the end-effector (e.g. on an internal link in between the base and the end-effector), as a consequence of contacts with the environment. In such cases, the application point (or the centroid of the contact region) must be known. Also in this case, Eq. 3.4.6 or Eq. 3.4.7 holds. Note that one external force can be determined if, and only if, all the other wrenches flowing through the edges connected to the link can be determined.

Consider the general example of one link connected to N other links, $N \geq 2$ (i.e. node 1 of Fig. 4.2). The graph associated to a similar situation instead is the central one of Fig. 4.2(b). The first step consists in setting the unknown wrenches given the quantities that have flown from the known leaves. These quantities can in general be measured

by FTS within a link. Secondly each of the links 4_F, 0 and 2_B connected to 1 perform the calculation (using Eq. 3.4.6 or Eq. 3.4.7) necessary to define the information passing through the edge which connect them to link 1, according to the direction of the edge. Then vertex 1 preforms again the evaluation of the force transmitted through the edges connecting 4_F, 0 and 2_B to 1, to perform the computation of the quantities flowing through the unknown edge, again from Eq. 3.4.6 or Eq. 3.4.7, according with the direction of the edge $e_{j,i}$. Note that in this example, the assumption that f_{ext} is the only unknown must hold.

4.2.3 Virtual Joint Torque Sensors

In case the Denavit-Hartenberg notation has been employed for the definition of the kinematic of the structure, an estimation of the joint torque can be performed, once the i-th wrench is known:

$$\tau_i = \mu_i^\top z_{i-1} \tag{4.2.1}$$

where z_{i-1} is the z-axis of the reference frame $\langle i-1 \rangle$ as in Fig. 3.3 (see Sciavicco & Siciliano (2005a)). The method shows that it is possible to have an estimation of joint torques, which can be used, successively, for joint torque control. Moreover, this is not the only information that it is possible to extract from the method. Joint torques are here found as one component of the wrenches flowing through the edges. These wrenches allow having a better representation of the possible contact situation, which can be used as a virtual measurement, to perform every kind of tasks involving force detection and control. It is necessary to note that the more 6-axis FTS are employed, the more accurate will be the estimation.

4.3 A Case Study: iCub dynamics

The method described in Sections Section 3.3, has been implemented on the 53 DOF humanoid robot iCub. More specifically, the hands have been considered as a unique rigid body, assuming that the motion of the fingers does not contribute to the variation of FTSs measurements. The same assumption has been considered for the eyes. The resulting overall number of degrees of freedom of the dynamical model of the robot is 32: 7 for each of the two arms, 6 for the legs, 3 for the torso and 3 for the neck. In

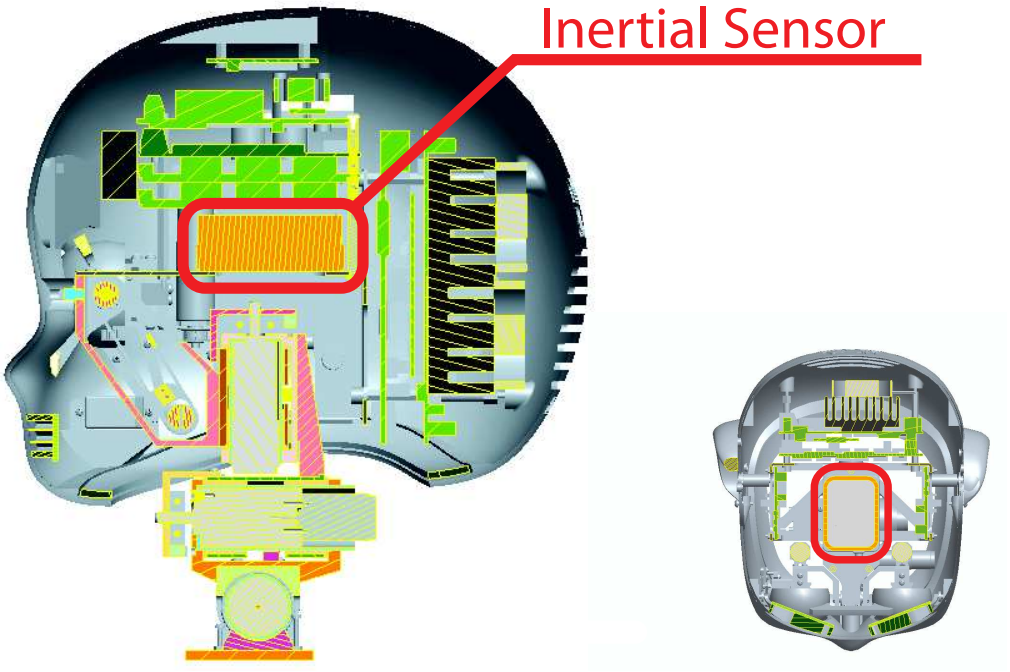


Figure 4.3: Section of the iCub head, to show once more the position of the 3 DOF Orientation Tracker.

this section it is analyzed the graph which refers to the kinematic and dynamic of the iCub. The sources of information are reported in Section 4.3.1, while Section 4.3.2 show the results obtained which validate the dynamical model and that allow to carry on the overall goal of this thesis, which is to improve the perceptual capabilities of the humanoid platform and consequently design force control for the iCub robot.

4.3.1 sensors

As shown in Chapter 2, iCub is equipped with one inertial sensor (Xsens MTx-28A33G25 XsensMTx (2010)) at the top of the head (see Figure 4.3), and four custom-made 6-axes F/T sensors (see Fumagalli *et al.* (2010)), one per leg and arm, each placed proximally. Excluding the hands DOF from the model, 32 DOF have been taken into account. The structure of the enhanced graphs required for the computation of the robot kinematic and dynamic quantities are reported and described in Section 4.3.1.1 and 4.3.1.2.

In this section it will be given a detailed description of the information that can be retrieved on the iCub robot, given the set of sensors that will be shown hereafter.

4.3.1.1 the inertial sensor

The inertial sensor provides angular velocities and linear and angular acceleration at the reference frame of the sensor. When connected to a link, these quantities can be propagated to the entire link, with Eq. 3.4.3, or to the connected links through Eq. 3.4.2 or Eq. 3.4.1 depending, in the graph framework, on the direction of the edges that connect the nodes. The graph associated to the propagation of these information, actually of velocities and acceleration of all the links, is shown in Figure 4.4. The structure of the enhanced graphs which is required to perform the computation of the robot kinematic quantities thus results to have one single known input (▼), which is placed at the final node of the head, and four unknowns (▽) each placed on the end-effectors of the limbs, and whose edges represent the frame of reference of the links they are connected to. These symbols have been positioned at the terminal points, because it is required the entire knowledge of the quantities flowing through all the edges of the graph representing the robot, which means that it is necessary to know the velocities and acceleration of the reference frames of all the links constituting the robot. Otherwise, the computation of wrenches cannot be performed. It is noticeable that all the proprioceptive information are required. In case of lacking of encoder measurement, a single inertial sensor would not be sufficient to propagate the information. On the other side, more inertial sensors might be employed to perform an improved estimation of the joint velocities and acceleration, as reported in Appendix 6.3.3.
and joint torques is shown in Figure 4.9. Notice that the dynamic enhanced graph is divided in five sub-graphs as a consequence of the application of step 2, in the procedure described in Section 4.1. Unknown wrenches (white rhombi) have been statically attached to hands and feet. However, this choice is totally arbitrary and depends on the application.
Moreover, it is to be underlined the importance of the inertial sensor, which allows to perform the Newton-Euler computations without a fixed base frame (as it is usually assumed in its classical applications). A clarifying example can be the case in which the iCub crawls as shown in Figure 4.5

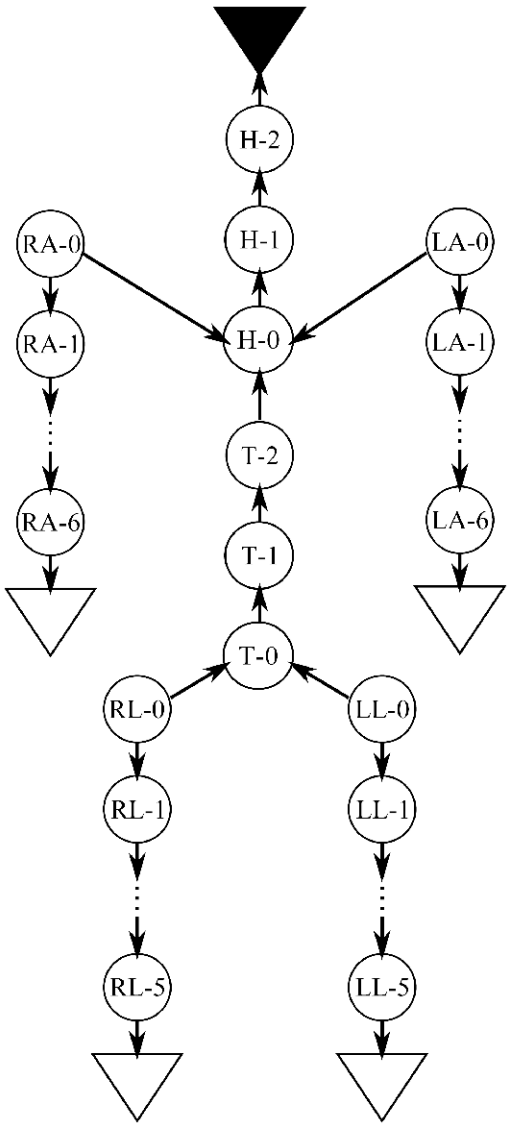


Figure 4.4: Representation of iCub's kinematic graph, using the notation of Fig. 3.6. A complete description of the iCub kinematics can be found in the online documentation available on the iCub website wiki, at the page: `http://eris.liralab.it/wiki/ICubForwardKinematics`.

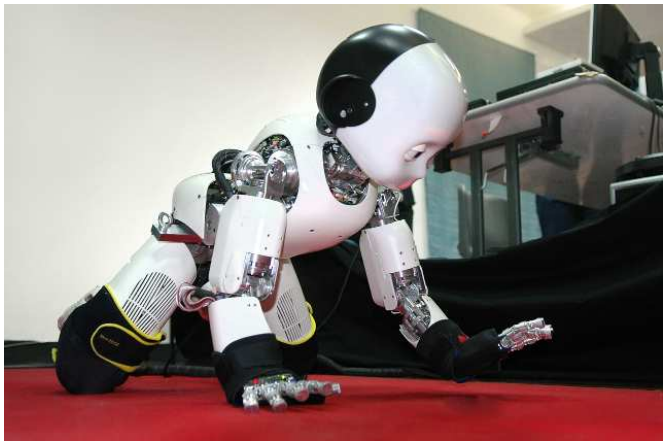


Figure 4.5: The humanoid robot iCub performing the crawling task. The task stimulates all the sensors, from the externally applied reaction forces of the floor.

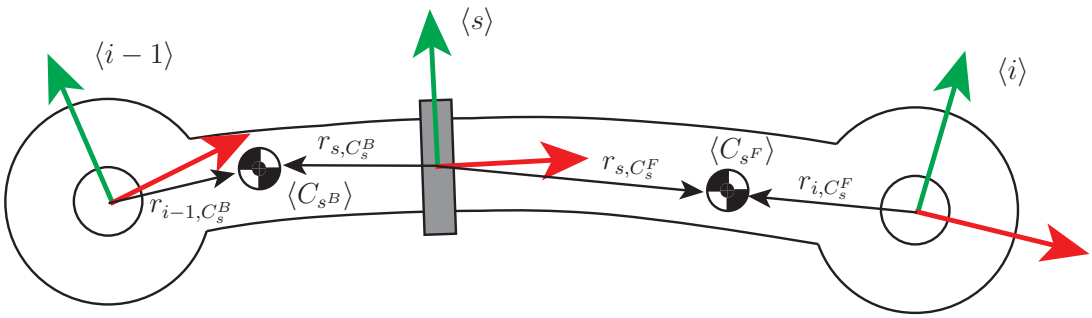


Figure 4.6: A representation of an F/T Sensor within the i_S-th link. Note that the sensor divide the link into two sub-links, each with its own dynamical properties.

4.3.1.2 the force/torque sensor

As previously expressed in Chapter 3, and summarized in Section 4.1, a sensor embedded in a link (see Figure 4.6) allows to divide the link into two sub-links, where the equilibrium is guaranteed by assigning to the edge, connecting the sub-links, the measure of the FTS. Practically it correspond to dividing the graph into two sub-graphs and introducing two black rhombi (i.e. two known wrenches), one on each sub-graph. More specifically, the sensor measures the wrench exerted by the "forward" sub-link to the "backward" sub-link (this will be represented by a first rhomboidal node). However, a wrench equal and opposite to the sensor measurement is also exerted by the "backward" sub-link to the "forward" sub-link (this will be represented by a second rhomboidal node). Under these considerations, the F/T sensor within a link will be

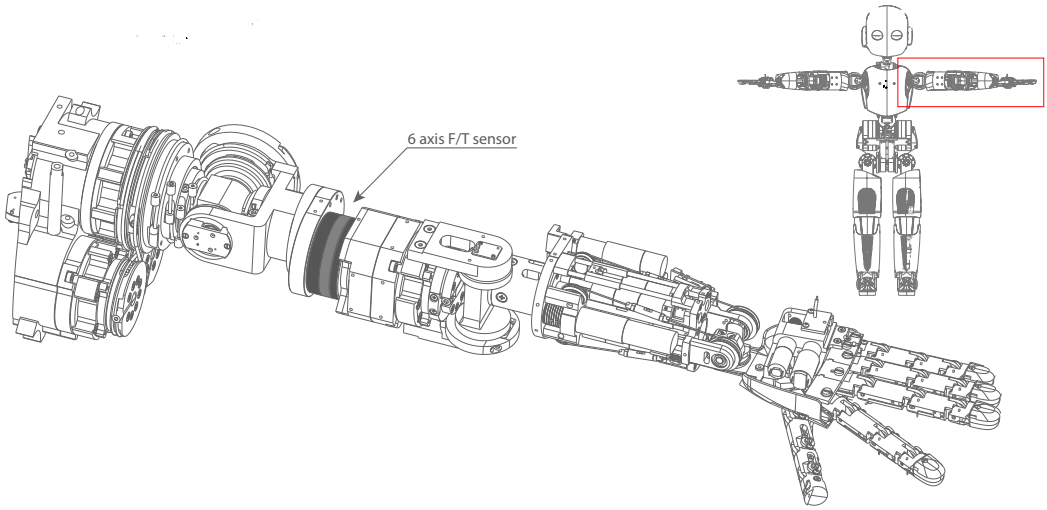


Figure 4.7: The iCub arm. A CAD view of the iCub arm to put in evidence the presence and the position of the F/T sensor.

represented by splitting the node associated to the link into two sub-nodes (with suitable dynamical properties, see Fig. 3.9(a)). On each sub-node, we have a known applied wrench which will be represented with black rhomboidal nodes. Figure 4.9 shows the graph corresponding to the iCub dynamic model. As represented in Figure 4.7 and Figure 4.8, the iCub robot mounts a set of four distributed FTSs. Each FTS is placed in a proximal position, within the limb. The FTSs of the arms are placed right after the 3-DOF shoulder universal joint, while the FTSs of the legs are placed after the first 2 joints of the hip. The corresponding graph can thus be divided into 5 sub-chains. Four chains are serial, single branched kinematic chain, while one is a multiple branched kinematic tree. The resulting graphs show that it is possible to detect a total of $5\Diamond$, one for each sub-chain. It should be noticed once more the importance of an artificial skin, which allows a dynamical allocation of unknown contact points, through sets of distributed tactile sensors, currently under development Cannata *et al.* (2008b). As soon as the tactile feedback will be available, the graph structure will be defined on the fly, based on the point of contact positions.

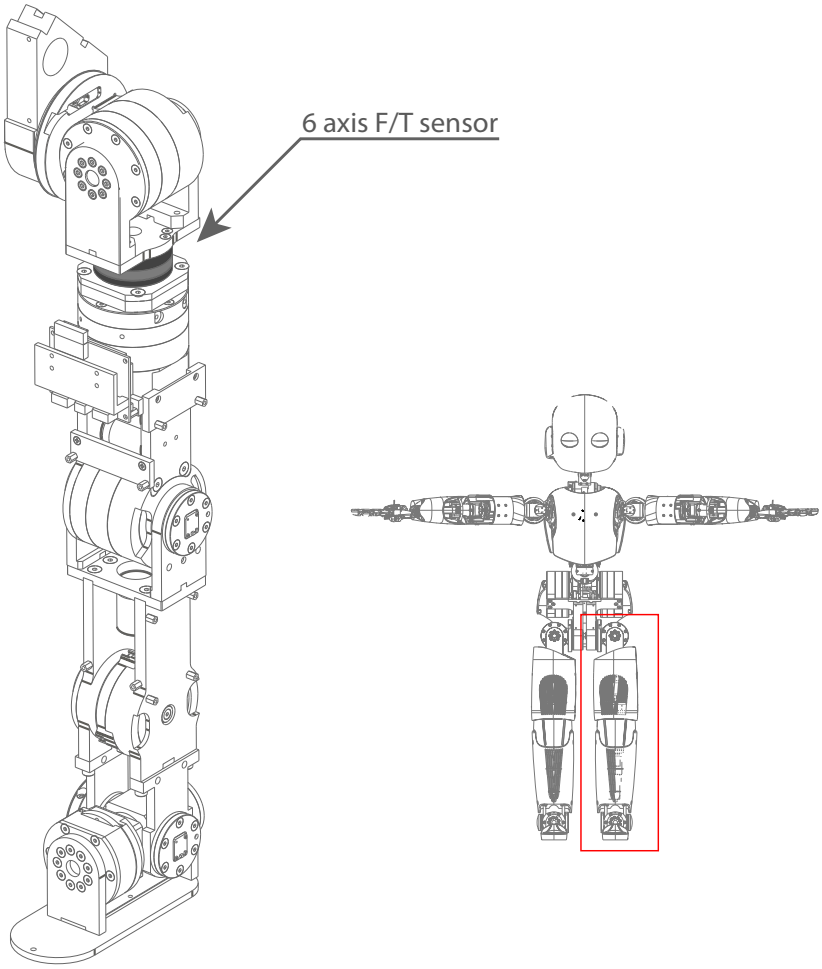


Figure 4.8: The iCub leg. A CAD view of the iCub leg to put in evidence the presence and the position of the F/T sensor.

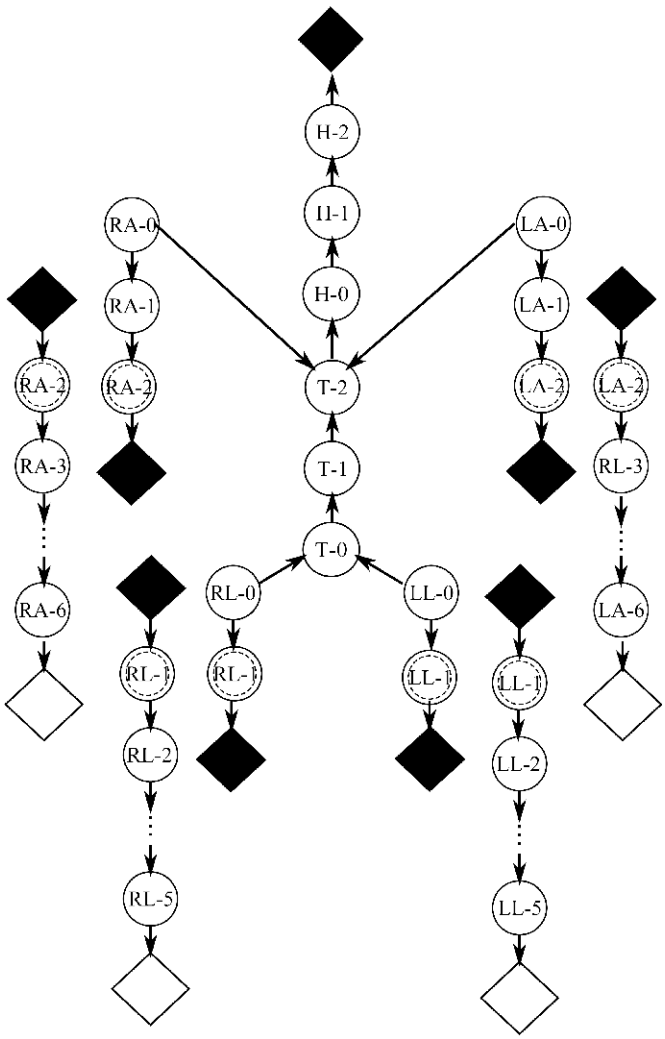


Figure 4.9: Representation of iCub's dynamic enhanced graph. Notice that all the limbs are divided into two sub-graphs in correspondence of the the F/T sensors located as sketched in Figure 4.7 and Figure 4.8

4.3.2 Experiments

The aim of this section is to validate the theoretical method presented in Chapter 3. The experiments have been conducted on the iCub, where a dynamical model with the form of the graphs represented in Figure 4.4 and Figure 4.9 has been built. A 3D Orientation Tracker Xsens MTx placed on its head allows to measure ω, $\dot{\omega}$, $\ddot{p}$ for the head link. Encoders are used to measure all the joints positions and joint velocities and accelerations are derived from position measurements through a least-squares algorithm based on an adaptive window Janabi-Sharifi *et al.* (2000). Force/torque sensors mounted proximally in the limbs as in Figure 4.7 and Figure 4.8 allow to measure also forces acting in between the sensor and distal joints.
The method has been tested through three experiments. As first, the validation of the dynamical model is performed by comparing measurements from the F/T sensors with the prediction of these measurements based on the dynamical model only (actually the limbs where moving freely, without interaction of externally allpied forces). Secondly, we exploited a commercial F/T sensor to produce a known external wrench on the robot and to compare the external sensor measurement with the external wrench computation obtained as described in Section 4.1. Finally, we tested our procedure for computing joint torques (4.2.1) by comparing our joint torque estimation with a joint torque measurement obtained by projecting a known wrench (once again measured with an external sensor) on the joints.

4.3.2.1 Validation of the Dynamical Model

In a first experiment we tested the validity of our dynamical model of the iCub limbs. To this purpose, the measurements w_s from the 4 six-axes F/T sensors embedded in the limbs have been compared with the analogous quantity $\hat{w}_s$ predicted by the dynamical model only. Sensor measurements w_s can be predicted assuming null wrench at the limbs. extremities (hands or feet) and then propagating forces up to the sensors. Data presented in this section were recorded under this assumption. Table 4.1 summarizes the statistics of the errors $w_s - \hat{w}_s$ for each limb during a sequence of movements. In particular, the table shows the mean and the standard deviation of the errors between measured and predicted sensor wrench during the movements. More in particular, Fig. 4.10 plots w_s and $\hat{w}_s$ for the left arm during the same sequence of movements.

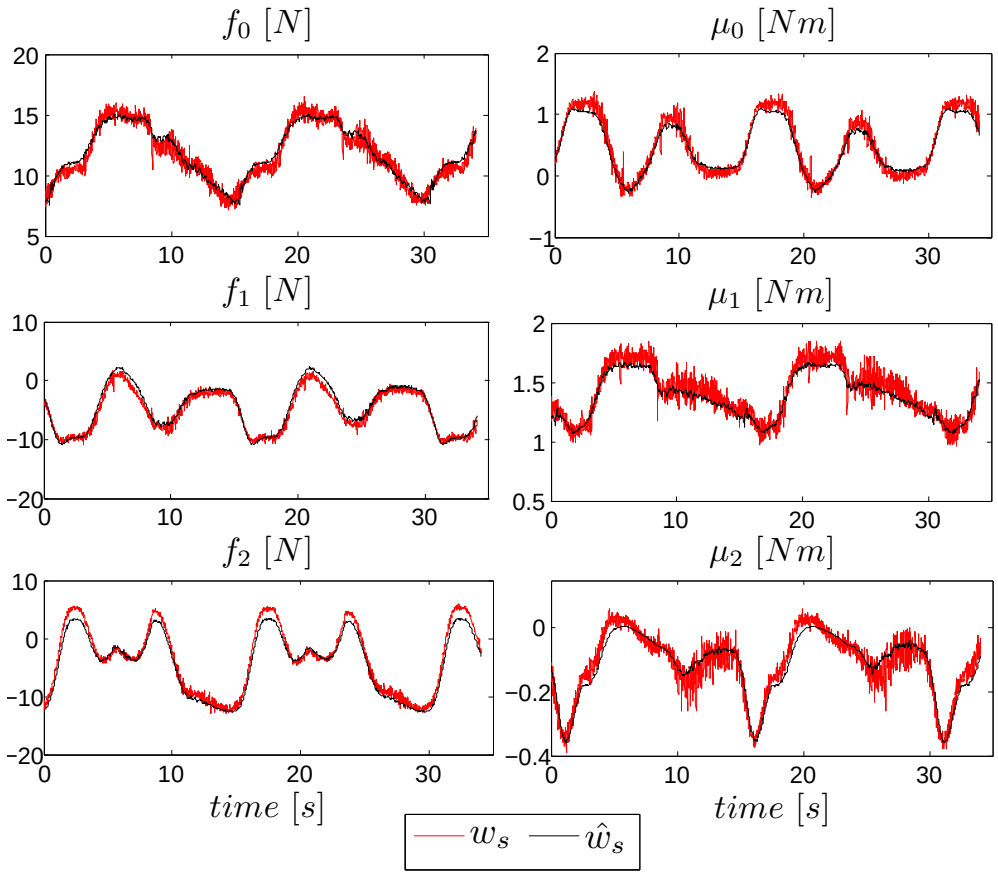


Figure 4.10: Left arm: comparison between the wrench measured by the FT sensor and the one predicted with the model, during the "Yoga" demo.

right arm: $\epsilon \triangleq \hat{w}_{s,RA} - w_{s,RA}$

	ϵ_{f_0}	ϵ_{f_1}	ϵ_{f_2}	ϵ_{μ_0}	ϵ_{μ_1}	ϵ_{μ_2}
$\bar{\epsilon}$	-0.3157	-0.5209	0.7723	-0.0252	0.0582	0.0197
σ_ϵ	0.5845	0.7156	0.7550	0.0882	0.0688	0.0364

left arm: $\epsilon \triangleq \hat{w}_{s,LA} - w_{s,LA}$

	ϵ_{f_0}	ϵ_{f_1}	ϵ_{f_2}	ϵ_{μ_0}	ϵ_{μ_1}	ϵ_{μ_2}
$\bar{\epsilon}$	-0.0908	-0.4811	0.8699	0.0436	0.0382	0.0030
σ_ϵ	0.5742	0.6677	0.7920	0.1048	0.0702	0.0332

right leg: $\epsilon \triangleq \hat{w}_{s,RL} - w_{s,RL}$

	ϵ_{f_0}	ϵ_{f_1}	ϵ_{f_2}	ϵ_{μ_0}	ϵ_{μ_1}	ϵ_{μ_2}
$\bar{\epsilon}$	-1.6678	3.4476	-1.5505	0.4050	-0.7340	0.0171
σ_ϵ	3.3146	2.7039	1.7996	0.3423	0.7141	0.0771

left leg: $\epsilon \triangleq \hat{w}_{s,LL} - w_{s,LL}$

	ϵ_{f_0}	ϵ_{f_1}	ϵ_{f_2}	ϵ_{μ_0}	ϵ_{μ_1}	ϵ_{μ_2}
$\bar{\epsilon}$	0.2941	-5.1476	-1.9459	-0.3084	-0.8399	0.0270
σ_ϵ	1.8031	1.8327	2.3490	0.3365	0.8348	0.0498

*: $\epsilon \triangleq \hat{w} - w = [\epsilon_{f_0}, \epsilon_{f_1}, \epsilon_{f_2}, \epsilon_{\mu_0}, \epsilon_{\mu_1}, \epsilon_{\mu_2}]$
*: SI Unit: $f : [N], \mu : [Nm]$.

Table 4.1: Errors in predicting F/T measurement (see text for details)

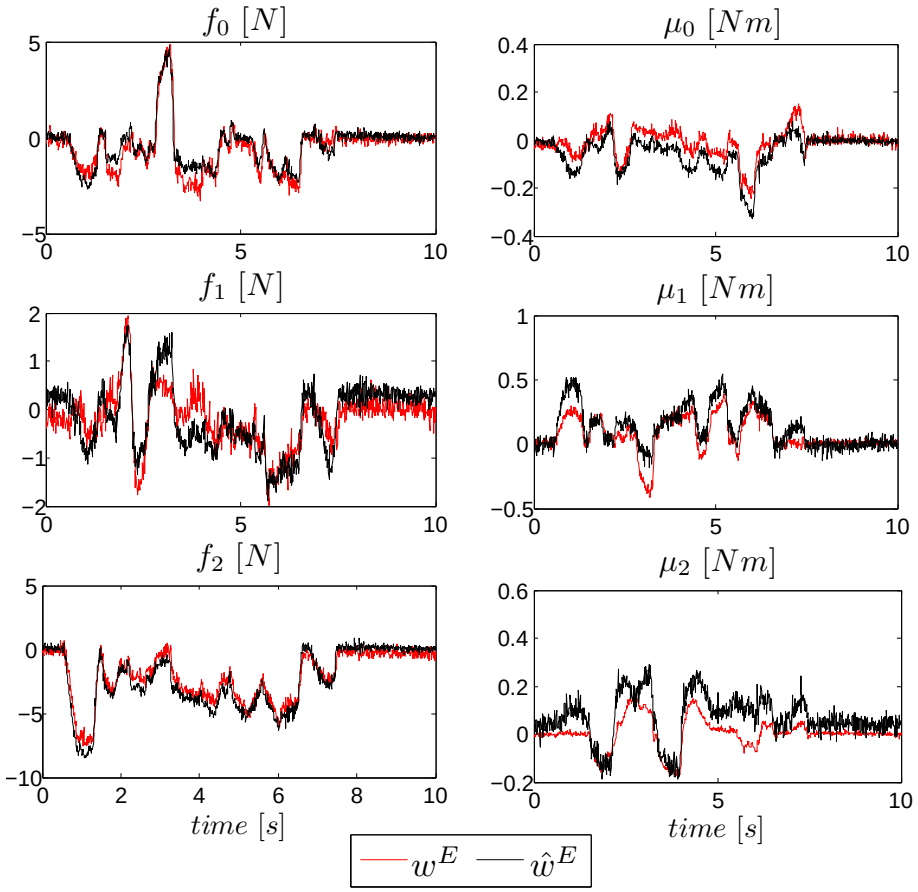


Figure 4.11: Left arm: comparison between the external wrench estimated after the FT sensor measurements and the one measured by an external FT sensor, placed on the palm of the left hand.

4.3.2.2 Estimation of external wrench

In a second experiment, we tested the effectiveness of our procedure for measuring unknown external wrenches as described in Section 4.1. In order to validate our measurement we generated the "unknown" wrenches with the help of an external six-axes F/T sensor so as to have a ground truth of the applied wrench. Experiments in this case were conducted only on the left arm. An external wrench w^E was applied at the left hand and measured with the external F/T sensor. Its value was then compared with $\hat{w}^E$, the estimation of the external wrench obtained by propagating the internal F/T measurements from the left arm sensor to the frame where w^E was applied. Propagation

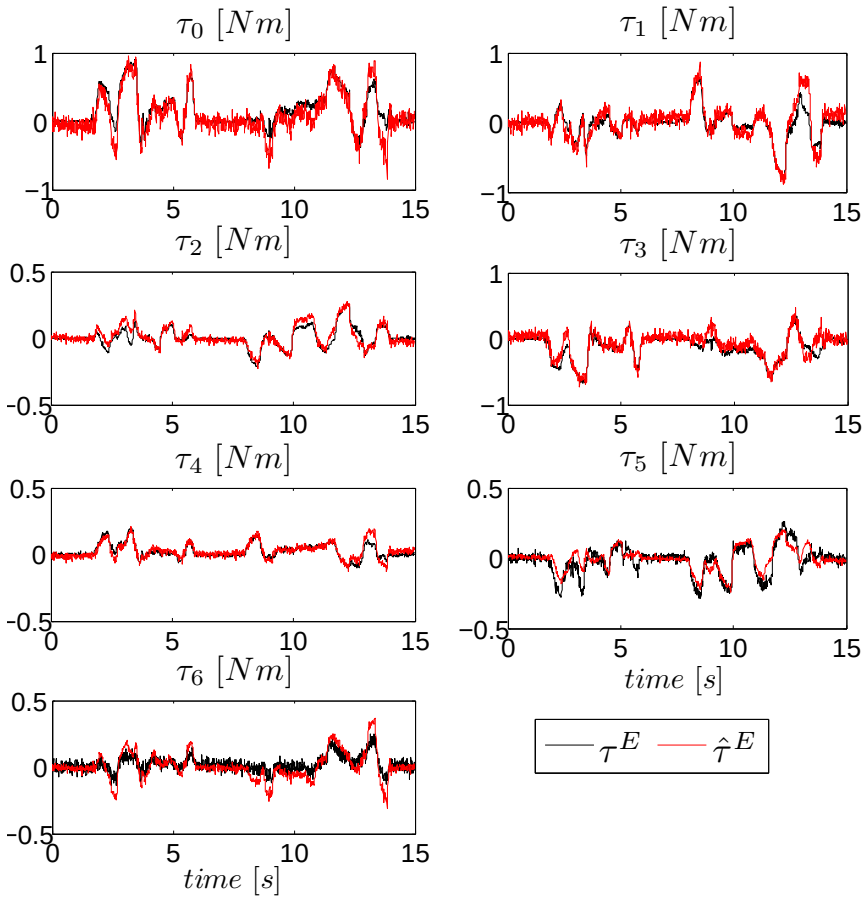


Figure 4.12: Left arm: comparison between the torques computed exploiting the FT sensor and the ones obtained by projecting the external FT sensor on the joints.

was performed according to the left arm enhanced graph in Fig. 4.4. A plot of w^E and $\hat{w}^E$ for various values of the stimulus is given in Fig. 4.11.

4.3.2.3 Estimation of external torques

In a third experiment we tested the validity of our procedure for estimating joint torques from the embedded F/T sensor[1]. Let this estimation be $\hat{\tau}$. In this case, given the difficulties in generating known torques at the joints, we proceeded as in the previous

[1]Torques estimation from embedded F/T sensors can be obtained using (4.2.1) with the μ_i computed propagating the F/T sensor information within the graph according to the procedure presented in previous sections.

experiment generating a wrench w^E and computing the corresponding torques τ^E at the joints with the following formula:

$$\tau^E = J_E^\top w^E, \tag{4.3.1}$$

where $J_E \in \mathbb{R}^{6\times n}$ is the Jacobian for the given wrench application point. The torques τ at the joints generally differ from τ^E since we have $\tau = \tau^I + \tau^E$, where τ^I represents the vector of joint torques due to the system internal dynamics. Keeping the robot fixed and assuming w^E null, we have τ=τ^I and we can use $\hat{\tau}$ to obtain an estimation of τ^I, denoted $\hat{\tau}^I$. Keeping the robot in the same configuration but letting $w^E \neq 0$, we can then estimate τ^E with the following formula:

$$\hat{\tau}^E = \hat{\tau} - \hat{\tau}^I.$$

Figure 4.12 shows a comparison between τ^E and $\hat{\tau}^E$ obtained with the procedure just described.

4. BUILDING EOG FOR COMPUTING DYNAMICS AND EXTERNAL WRENCHES OF THE ICUB ROBOT

5

Active Compliance Control

The iCub robot shoulder mechanism is constituted of three motors with parallel axis moving idle pulleys that cooperate together to move a universal 3DOF joints. This mechanism allows compactness and wide range of movements, similar to that of human shoulder. Moreover, the arm are lightweight because of the allocation of the motors in the torso link. Nevertheless, the control of such mechanism requires to be analyzed. In this section, an analysis of the dynamic of such mechanism is conducted. This chapter highlights the problematics related to coupled mechanisms during their motion control. In particular it will be shown how the decoupling of the iCub shoulder joint has been performed within this work. What will be shown here will allow to control joint movements while directly actuating the motors.

5.1 Dynamics of Coupled Mechanism

The shoulder mechanism can be considered as a reduction box which connects the motors and the joints through relationships that will be reported hereafter. Generally speaking, the shoulder mechanism allows the transmission of the motion and torque between the two side of the mechanism, the motor side and the load side. In this treatment, the system will be considered as ideal, i.e. without energy loss. It will be shown in Section 5.1.2 the conservation of power between the two input/output of the transmission, actually the load side and the motor side. In the particular case of the iCub robot, on one side the three motors are placed, with parallel axis. On the other

side, a 3 D.O.F. mechanism is connected, actually the shoulder of the robots. Here we define as *motor* the motor side and its dynamic, and *joint* the load side, actually the side of the transmission where links and joints are placed. The two systems, the motor and the joint are linked through kinematic relationships which couple the dynamic of the two systems. The motor and joint dynamic, indeed, cannot be studied separately. The motors should thus cooperate together to perform pure joint movements. Moving one joint means actuating more than one motor, accordingly to the kinematic of the transmission. In this Section, a complete analysis of the system is conducted. Section 5.1.1 shows the dynamic of the motors and of the joints separately. In Section 5.1.2, the equations of the transmission are defined, which dynamically couple the two sides. Section 5.1.3 instead describe the overall system dynamic. This relationship will then be used to derive the control strategies of the decoupled system, as presented in Section 5.2.

5.1.1 Dynamic of motors and links

In this paragraph, the formulation of the dynamic of the motors and the joints is derived. The two sides are here considered separately, even though it is put here in evidence the toques that the two subsystems exert reciprocally. The reason for this is that the two system are connected through a transmission system, which will be presented in 5.1.2. Step by step, through the following paragraphs, we will converge to the overall formulation of the coupled system. The dynamic of the motors can be in general described through its mechanical and electrical equations. The way the equations are derived is not the target of this Chapter. The motor side equation of motion is reported in equation (5.1.1):

$$I_m\ddot{\theta}_m + D_m\dot{\theta}_m = \tau_m - \tau_{mj} \tag{5.1.1}$$

where given a system actuated with N motors, I_m and $D_m \in \mathbb{R}^{N\times N}$ represent the diagonal inertia and damping matrices of the motors. The variable $\tau_m \in \mathbb{R}^N$ is the vector of motor torques, which is the input variable of the mechanical system. Particular attention is required for term $\tau_{mj} \in \mathbb{R}^N$. This vector represent the torque vector which the joint side of the transmission system exchange with the motor side, represented on the motor side. In other words, this term represent the overall torque that the joint side passes to the motor side, through the transmission mechanism. The mechanical and

electrical dynamic are linked together through electro mechanical interactions. Let us generally consider the electrical dynamic of the motors as:

$$L_m\dot{i} + R_m i = V_{in} - V_{bemf} \tag{5.1.2}$$

being L_m, R_m the motor inductance and resistance of the spires of the motor and $V_{bemf} = k_\omega \dot{\theta}_m$ *Back Electro-Motive Force* generated by the relative motion of the magnetic field and the spires of the motor.
The overall torque that the magnetic field can generate on the motor shaft can be derived, for this kind of model as:

$$\tau_m = k_i i \tag{5.1.3}$$

being i the current passing through the spires of the motors and k_i the torque constant diagonal matrix. The knowledge of the electric dynamic of the motor is necessary for determining the current which flows into the wires, given an input voltage $V_{in} \in \mathbb{R}^N$.
The dynamic of the joint side of the transmission, actually the dynamic of the robotic system, can be described as shown in Chapter 3. The only difference that is reported here is that the input torque to the link side of the manipulator is not directly the motor torque τ_m, but it is the torque that the transmission system generates on the joint side τ_{jm}. In general, the joint equation of motion can thus be written as:

$$I_j(\theta_j)\ddot{\theta}_j + C_j(\theta_j, \dot{\theta}_j)\dot{\theta}_j + G_j(\theta_j) = \tau_{jm} - \tau_j \tag{5.1.4}$$

where $I_j(\theta_j)$, $C_j(\theta_j, \dot{\theta}_j)$ and $G_j(\theta_j)$ represent the Inertia matrix of the manipulator and the centrifugal and Corioli's term and gravitational contribution to the transmission system, τ_j is the external joint torque vector. For ease of treatment, hereinafter the gravitational and centrifugal terms will be included in a single term τ_j, which also includes possible external joint torque components. The overall equation of motion of the joint side of the transmission thus becomes:

$$I_j(\theta_j)\ddot{\theta}_j = \tau_{jm} - \tau_j \tag{5.1.5}$$

5.1.2 Kineto-Static Equation of the Transmission

As mentioned in Section 5.1, the transmission system couples the dynamic of the manipulator (actually the joint side of the transmission system) to the dynamic of the motors. In this treatment, we consider only the case of rigid transmission. Elasticity of the transmission elements is not here introduced for it is not inside the scope of this manuscript. The main reasons of this choice are the difficulty in the identification of the stiffness of the elastic elements (mainly the tendons), the variability of the stiffness due to the tendons preload (which is not controlled during the assembling of the robot), the lack of sensors on the two sides of the transmission, the high nonlinearity of tendon stiffness and the complication which follows from the introduction of an elastic element in the low level software implementation of an appropriate control strategy. Moreover, we consider here a linear and non dissipative transmission system, which allow us to assume that the power at the joint side P_j and the power at the motor side P_m is conserved. In this section the relationships which couple the dynamic of motors to that of joint is presented. First a kinematic analysis of the mechanism is presented, then the coupling of torques is also analyzed. The treatment of these concept in this paragraph is conducted for quasi-static condition. Nevertheless, the assumptions does not affect the generality of the approach and, in next paragraph, the case is extended to dynamic conditions.
The kinematic of the load side of the mechanism can be in general described as a function of the motor variables:

$$\theta_j = f(\theta_m) \tag{5.1.6}$$

being $f(.)$ a general function which describes the relationship between the motor coordinates θ_m and the joint θ_j. Deriving equation (5.1.6) with respect to time follows that the kinematic coupling between motor angular velocities $\dot{\theta}_m$ and joint velocities $\dot{\theta}_j$ can be described with a linear operator of the form:

$$\dot{\theta}_j = T_{jm}(\theta_m)\dot{\theta}_m \tag{5.1.7}$$

where $T_{jm}(\theta_j) \in \mathbb{R}^{m\times n}$ represents in general a non-linear operator (a Jacobian) which relates the velocities in the motor space $\dot{\theta}_m \in \mathbb{R}^m$ to the velocities of the joint space

$\dot{\theta}_j \in \mathbb{R}^n$. This relationship shows the contribution of motor velocities to joint velocities. As an example, the velocity of joint k depend on the velocities of the motors as $v_j^k = f_{k,1}v_{m1} + f_{k,2}v_{m2} + \cdots + f_{k,n}v_{mn}$.
The inverse relationship T_{mj}, which relates motor velocities as a function of joints velocities as $v_m^i = f_{i,1}^{-1}v_{j1} + f_{i,2}^{-1}v_{j2} + \cdots + f_{i,l}^{-1}v_{jl}$ can be derived as follows: let us assume that an inverse operator exists, which relates motor velocities to joint velocities:

$$\dot{\theta}_m = T_{mj}(\theta_j)\dot{\theta}_j \tag{5.1.8}$$

substituting 5.1.8 in (5.1.7) we obtain:

$$\dot{\theta}_m = T_{mj}(\theta_j)T_{jm}(\theta_m)\dot{\theta}_m \tag{5.1.9}$$

If we now bring both the terms to the left sides of the equation:

$$\left(I - T_{mj}(\theta_j)T_{jm}(\theta_m)\right)\dot{\theta}_m = 0 \tag{5.1.10}$$

which, for every vector of motor velocities is verified if:

$$T_{mj}^{-1}(\theta_j) = T_{jm}(\theta_m) \tag{5.1.11}$$

The inverse relationship exists whenever $det(T_{mj}) \neq 0$[1].
If we now take into consideration the work that the motor and the load generates in quasi-static conditions:

$$\begin{aligned} \delta W_m &= \tau_m^\top \delta\theta_m = \tau_m^\top \dot{\theta}_m \delta t \\ \delta W_j &= \tau_j^\top \delta\theta_j = \tau_j^\top \dot{\theta}_j \delta t \end{aligned} \tag{5.1.12}$$

that, in the hypothesis that the transmission is not dissipative and rigid, the overall instantaneous power $P\delta t$ (or virtual work δW, being $\delta W = P\delta t$) is conserved, which means that $\delta W_m = \delta W_j$ it follows:

$$\tau_m^\top \delta\theta_m = \tau_j^\top \delta\theta_j \tag{5.1.13}$$

[1] As will be shown in section 5.3, the coupling matrix of the iCub shoulder mechanism is triangular, with constant values and non-null elements on its diagonal. Given these consideration, the inverse relationship exists.

In non static condition, assuming that the hypothesis are still valid, the variables which are the input/output to the transmission system are no longer the motor torque τ_m and joint torque τ_j, but the torques which are effectively transmitted from the motor to the load and viceversa. These quantities, that are τ_{jm} and τ_{mj}, are the overall torques which are input/output of the transmission mechanism and which take into account also the dynamic of the two sides. In non quasi-static condition equation (5.1.12) thus becomes:

$$\begin{aligned} P_m \delta t = \tau_{mj}^\top \delta\theta_m = \tau_{mj}^\top \dot{\theta}_m \delta t \\ P_j \delta t = \tau_{jm}^\top \delta\theta_j = \tau_{jm}^\top \dot{\theta}_j \delta t \end{aligned} \tag{5.1.14}$$

and dividing by δt both the members:

$$\begin{aligned} P_m = \tau_{mj}^\top \dot{\theta}_m j \\ P_j = \tau_{jm}^\top \dot{\theta}_j m \end{aligned} \tag{5.1.15}$$

and considering again that $P_m = P_j$, it follows that:

$$\tau_{mj}^\top \dot{\theta}_m = \tau_{jm}^\top \dot{\theta}_j \tag{5.1.16}$$

Let us now substitute (5.1.8) into (5.1.16)

$$\tau_{mj}^\top T_{mj}(\theta_j)\dot{\theta}_j = \tau_{jm}^\top \dot{\theta}_j \tag{5.1.17}$$

which is true for every non zero joint velocities if:

$$T_{mj}(\theta_j)^\top \tau_{mj} = \tau_{jm} \tag{5.1.18}$$

here again, the inverse relationship exists and is given by:

$$\tau_{mj} = T_{mj}(\theta_j)^{-\top} \tau_{jm} = T_{jm}(\theta_m)^\top \tau_{jm} \tag{5.1.19}$$

where $A^{-\top}$ indicates the transposed (pseudo-)inverse of a matrix A.

5.1.3 Dynamics:

Given the consideration of the previous paragraph, let us now couple the motor side and the joint side through equation (5.1.19). If we now consider the dynamics of the coupled joints, the equation of motion of the coupled mechanism become:

$$\begin{aligned} I_m\ddot{\theta}_m + D_m\dot{\theta} &= \tau_m - \tau_{mj} \\ I_j\ddot{\theta}_j &= n^\top T_{jm}^{-\top}\tau_{mj} - \tau_j \end{aligned} \tag{5.1.20}$$

where $n \in \mathbb{R}^{m\times m}$ is the diagonal matrix of reduction ratio of the gear boxes. T_{jm} represents the coupling matrix of (5.1.7).
Combining τ_{mj} of equation (5.1.20), considering (5.1.9) the motor side equation of motion become:

$$I_m\ddot{\theta_m} + D_m\dot{\theta_m} = \tau_m - T_{jm}^\top n^{-\top}(\tau_j + I_j n^{-1}T_{jm}\ddot{\theta}_m) \tag{5.1.21}$$

which can be rearranged as:

$$\left(I_m + T_{jm}^\top n^{-\top} I_j n^{-1} T_{jm}\right)\ddot{\theta_m} + D_m\dot{\theta_m} = \tau_m - T_{jm}^\top n^{-\top}\tau_j \tag{5.1.22}$$

Equation (5.1.22) represents the equation of motion of the rigid system. Equation (5.1.22) shows the dependency of the coupled system dynamic from the motor torque and from the external torque on the joint side. The resulting system dynamics thus become:

$$\tau_m = \left(I_m + \frac{\left(T_{jm}^\top I_j T_{jm}\right)}{n^2}\right)\ddot{\theta_m} + D_m\dot{\theta_m} + T_{jm}^\top n^{-\top}\tau_j \tag{5.1.23}$$

5.2 Control

In section 5.1.1 it has been introduced the dependency of motor torques from the motor and load dynamics, through dynamical relationship which are kinematically coupled together through a transmission transmission element which does not allow to have a direct correspondence between one motor and one joint. Moreover, Eq. 5.1.2 have shown the dependency between the electrical dynamic and the mechanical dynamic.

Considering the electro-mechanical generation of motor torque, as shown in (5.1.3), and considering the coupled dynamics of joints and motors, in terms of motor torque balancing represented by Eq. 5.1.23, the overall system dynamic becomes:

$$\begin{cases} k_i i = \left(I_m + \frac{\left(T_{jm}^{\top} I_j T_{jm}\right)}{n^2} \right) \ddot{\theta_m} + D_m \dot{\theta_m} + T_{jm}^{\top} n^{-\top} \tau_j \\ L_m \dot{i} + R_m i = V_{in} - k_\omega \dot{\theta}_m \end{cases} \tag{5.2.1}$$

The goal of this section is to define a proper control strategy which allows to control pure joint movements, by assigning a control input to the motors. If we consider the system dynamic, the control input that allows to generate motor torque is the input voltage to the motor spires V_{in}. Setting $V_{in} \neq 0$ cause the rise of the current i_m flowing through the motor spires which, on the mechanical side, generates motor torque.
If we consider, for ease of treatment, that the electrical dynamic is much more fast than the mechanical dynamic, it is possible to neglect the transition of the current, and thus the term $L_m \dot{i}$ of Eq. 5.2.1. If this assumption is verified, the dynamic of the electro-mechanical coupled system can be written as:

$$k_i \frac{V_{in} - k_\omega \dot{\theta}_m}{R_m} = \left(I_m + \frac{\left(T_{jm}^{\top} I_j T_{jm}\right)}{n^2} \right) \ddot{\theta_m} + D_m \dot{\theta_m} + T_{jm}^{\top} n^{-\top} \tau_j \tag{5.2.2}$$

where, as previously mentioned, V_{in} is the input voltage that we can directly control to generate motor torque. Let us consider the goal of controlling pure joint movements, in the sense that we are interested in assigning a proper control input to the motor, such that dynamic of the transmission becomes *invisible* from the point of view of high level commands. To achieve this issue, we can assign to the input voltage V_{in} a control input u of the form:

$$u = \frac{R_m}{k_i} \left[\left(I_m + \frac{\left(T_{jm}^{\top} I_j T_{jm}\right)}{n^2} \right) n T_{jm} y + (D_m + k_\omega) \dot{\theta}_m + T_{jm}^{\top} n^{-\top} \tau_j \right] \tag{5.2.3}$$

If the control input u is dimensionally consistent with the input voltage V_{in}, we can directly assign $u = V_{in}$ in order to obtain a dynamic of the controlled system of the form of:

$$\ddot{\theta}_m = n T_{mj}^{-1} y \tag{5.2.4}$$

which, considering Eq. 5.1.19 represent the decoupled dynamic of the system, where y is a new control input which allows to directly control the joint accelerations in the joint space:

$$\ddot{\theta}_j = y \tag{5.2.5}$$

It is remarkable here to notice that, when implemented on motor-control boards, we cannot typically consider to perform operation with floating point quantities. The floating point operation, in fact, require a dedicated unit (FPU, Floating Point Unit). If not present, the software is in charge of simulating this unit, but this requires a lot of resources that cannot be employed with DSPs when they are also required to perform real-time control. In this case, u and V_{in} will be scaled of an integer quantity α, which allows to convert the quantities necessary for control into machine units. the controlled system becomes:

$$u = \alpha \frac{R_m}{k_i} \left[\left(I_m + \frac{\left(T_{jm}^{\top} I_j T_{jm}\right)}{n^2} \right) n T_{jm} y + (D_m + k_\omega)\dot{\theta}_m + T_{jm}^{\top} n^{-\top} \tau_j \right] \tag{5.2.6}$$

In the following sections will be defined the new control input such that the system takes the desired behavior. In particular, starting from the definition of position control law, also torque control and impedance control will be defined.

5.2.1 Position Control

The control of the position can be achieved by assigning a control input y of the form:

$$y = \ddot{\theta}_j^d + K_d(\dot{\theta}_j^d - \dot{\theta}_j) + K_p(\theta_j^d - \theta_j) \tag{5.2.7}$$

The control of Eq. 5.2.7 requires the knowledge of the position error ($e = \theta_j^d - \theta_j$) and its derivative. Moreover, it requires the desired acceleration of the system $\ddot{t}heta_d$ as a feed-forward term. If substituted to Eq. 5.2.5, the overall controlled system dynamics takes the form of:

$$\ddot{e} + K_d\dot{e} + K_p e = 0 \tag{5.2.8}$$

Eq. 5.2.8 shows that if the dynamic of the system is perfectly known, it is possible to ideally obtain a second order dynamic of the tracking error, which is asymptotically

stable for $K_d > 0$ and $K_p > 0$.
If the model of the system is not perfectly known, another kind of controller is preferred, such that the disturbances introduced by model errors become negligible. When errors are present in the inverse dynamic compensation of the system, in fact, the feedback linearization brings to a dynamic of the form:

$$\ddot{\theta}_j = y + \zeta \tag{5.2.9}$$

where ζ takes into account all the disturbances due to the presence of errors in Eq. 5.2.6. Different strategies can handle this issue, such as the robust control shown in Sciavicco & Siciliano (2005a). Here is presented the effect of the integral component of the position error.
If we assign y of the form:

$$y = \ddot{\theta}_j^d + K_d\dot{e} + K_p e + K_i \int e \tag{5.2.10}$$

and if we change variables such that $x = \int e$, the overall system dynamics takes the form of:

$$\dddot{x} + K_d\ddot{x} + K_p\dot{x} + K_i x = \zeta \tag{5.2.11}$$

Which shows that at steady-state, the disturbance ζ influences the term $K_i x$, while $\dot{x}$ goes to zero. Considering finally that $\dot{x} = e$, we have obtained a null steady state error.

5.2.2 Torque Control

When instead we are interested in assigning to the controlled system a behavior that is compliant to externally applied forces, it is possible to define a control input y of the form:

$$y = I_d^{-1}(\tau_j^d - \tau_j) - I_d^{-1} D_d \dot{\theta}_j^d \tag{5.2.12}$$

This control input allows to obtain a controlled system which behave as a damped mass. Practically speaking, when an external generalized force is applied to the robot, the system accelerates as if we are applying the same force to an inertial system, with inertia along the joint direction I_d. At steady state, the system moves with a velocity that depends on the term D_d as $\dot{\theta}_{ss} = \frac{1}{D_d} e_\tau$.

A control input y as the one proposed in Eq. 5.2.12 allows, in fact, to obtain a controlled system behavior of the form:

$$I_d\ddot{\theta}_j + D_d\dot{\theta}_j = \tau_j - \tau_j^d \tag{5.2.13}$$

It is remarkable here that τ_d is a torque reference, which needs to be tracked. Looking in details at Eq. 5.2.12, the torque error $e_\tau = \tau_j^d - \tau_j$ is controlled through a pure proportional gain $Kp = I_d^{-1}$. We can in fact here write Eq. 5.2.12 as:

$$y = K_p(\tau_j^d - \tau_j) - K_d\dot{\theta}_j^d \tag{5.2.14}$$

where $K_d = I_d^{-1}D_d$. If we are interested to control the torque error, i.e. we are interested in obtaining a null steady state torque error, the control strategy can be modified as follows:

$$y = K_p(\tau_j^d - \tau_j) + K_i\int(\tau_j - \tau_j^d) - K_d\dot{\theta}_j^d \tag{5.2.15}$$

where the integral term guarantees null torque error at steady state.

5.2.3 Impedance Control: classical

A pretty classical approach to impedance control exploiting feedback linearization is presented here. Implementing impedance control means that we want to assign to the controlled system a behavior of a mass, spring and damping system which moves connected to a desired trajectory. When a force acts on this *virtual mass*, the system dynamically respond to this disturbance accelerating, moving to a new equilibrium position which depends on the applied force as $e = \frac{F_{ext}}{K_s}$, being F_{ext} the externally applied force, and K_s the stiffness of the *virtual spring*.
In practice, to achieve this behavior, it is possible to assign a controlled input of the form:

$$y = \ddot{\theta}_j^d + I_d^{-1}(D_d(\dot{\theta}_j^d - \dot{\theta}_j) + K_d(\theta_j^d - \theta_j) + (\tau_j - \tau_j^d)) \tag{5.2.16}$$

This controlled input does not differ from the one of Eq. 5.2.7, a part from the addition of a force feedback term which allows to obtain a compliant behavior to the application

of external forces. By applying y, the controlled system takes the form:

$$I_d\ddot{e}_j + D_d\dot{e}_j + K_d e_j = e_\tau \tag{5.2.17}$$

being $e_\tau = \tau_j - \tau_j^d$. Another possibility to achieve the same behavior which, from an implementation point of view, allows to reuse part of the code which performs torque control expressed by Eq. 5.2.14. It is possible in fact to assign the spring and damping behavior through the control input τ_d. We can in fact define:

$$\begin{cases} y = I_d^{-1}(\tau_j^d - \tau_j) \\ \tau_j^d = I_d\ddot{\theta}_j^d + D_d(\dot{\theta}_j^d - \dot{\theta}_j) + K_d(\theta_j^d - \theta_j) + \hat{\tau}_d \end{cases} \tag{5.2.18}$$

where $\hat{\tau}_d$ becomes a new torque reference which can be used, for example, to compensate for the gravitational term.
Also in this case, model errors can give origin to disturbance terms that does not allow to perfectly achive null steady state error. Another possible solution to this problematic can be achieved by the exploitation of Eq. 5.2.15 as will be presented in next section.

5.2.4 Impedance Control: the role of integral

Let us now consider a torque regulator of a form similar to the one presented in Eq. 5.2.15:

$$y = K_p(\tau_j^d - \tau_j) + K_i \int (\tau_j^d - \tau_j) - K_i K_p \dot{\theta}_j^d \tag{5.2.19}$$

where $Kp = I_d^{-1}$. Let us define the desired torque τ_d of the form of:

$$\tau_j^d = I_d\ddot{\theta}_j^d + D_d(\dot{\theta}_j^d - \dot{\theta}_j) + K_d(\theta_j^d - \theta_j) + \hat{\tau}_d \tag{5.2.20}$$

If we substitute this control input in Eq. 5.2.19 and y in Eq. 5.2.9 we obtain a dynamic of the error of the form of:

$$I_d\ddot{e}_j + D_d\dot{e}_j + K_d e_j + K_i D_d \int \dot{e}_j + K_i K_d \int e_j = e_{\hat{\tau}} + \int e_{\hat{\tau}} + K_i I_d \int \ddot{\theta}_j^d - K_i K_p \dot{\theta}_j^d + \zeta \tag{5.2.21}$$

and if we change the variable name, such as $x = \int e_j$, the resulting dynamics takes the form of:

$$I_d \dddot{x} + D_d \ddot{x} + (K_d + K_i D_d)\dot{x} + K_i K_d x = e_{\hat{\tau}} + \int e_{\hat{\tau}} + \zeta \tag{5.2.22}$$

The stability of this system depends on the eigenvalues of the third order system, but it is not here analyzed. It is remarkable to notice that the overall stiffness of the controlled system become dependent on both the desired value K_d, but also on the integral gain of the torque regulator and on the desired damping of the impedance relationship. Moreover, the coupled dynamics may introduce oscillatory behaviors that are due to the presence of the integral of the torque error and on the integral of the position error that, due to possible errors in the model of the system, might be in contrast. Depending on the dynamic (the gains) of the controlled system, stable behaviors can achieved.
Nevertheless, it is important to notice that the presence of un-modeled effects, such as coulomb friction, might induce to cycle limits that do not allow to properly control the system position at steady state.

5.2.5 Considerations

From equation 5.1.23 and the control strategies presented in the chapter, it is possible to define different situation which are important for an easy implementation on the control boards.

- In case of high reduction ratio n of the gear boxes, the term $J_j \frac{\left(T_{mj}^\top T_{mj}\right)^{-1}}{n^2}$ can be neglected.

- The back-emf compensation contributes to the rising of the performances for both the torque control and also impedance control.

- When impedance controlled, the damping gain D_d give an important contribution to the tracking of position trajectories. When the stiffness is set to a low value, small errors lead to noticable errors in the tracking of the trajectory. An high damping gain on the derivative of the position error allows to reduce the

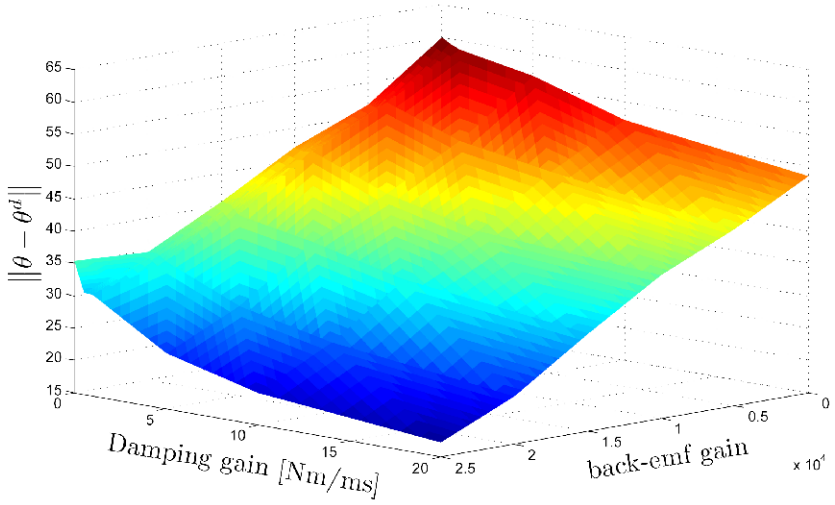


Figure 5.1: Plot of the dependence on the back-emf and desired damping D_d, of the norm of the trajectory error $\|\theta - \theta^d\|$. The plot refers to the motion tracking of one joint controlled through the impedance relationship of Section 5.2.3

tracking error during motion tasks. Moreover, low damping is not desired because it introduces overshoots. Figure 5.1 shows the effects of the back-emf and of the damping gain D_d on the norm of the tracking error, given a desired trajectory.

5.3 a Case Study: The iCub Arm

In this section is presented the case of the iCub arm, which have been taken into account as the most significant example of coupled mechanism of the iCub robot. Other coupled mechanisms that are present in the iCub are the wrists and the torso. The wrists present a tendon driven semi-differential mechanism that perform the pitch and roll movements of the hand. The third joint, actually the yaw movement of the hand, is achieved by a third motor housed in the forearm, proximal to the elbow link.
Similarly to the wrist, the torso joints movements are performed through a similar mechanism which employ two motors in a differential configuration to move the roll and the pitch joints. A third motor actuates the yaw movement. It is to be noticed that the choice of these mechanisms are the result of two different requirements: the wrist employs a semi-differential mechanism to minimize the dimension and weight of the

wrist joints (it will be better presented later); the torso instead adopt this solution to obtain an higher torque to dimension ratio. The two motors actuating the torso mechanism, in fact, work together to move the whole mass of the iCub robot. As an example, when the iCub is on its support structure, the torso joints have the task to move the entire upper part of the robot. A serial solution would have required bigger motors in order to handle with the requisites of torque necessary to carry the load of the upper part (upper torso, head and two arms).

The shoulder mechanism have been designed to achieve a similar task. The requisite of low weight and small dimension of the platform, have conducted the choice of the shoulder mechanism to the design of a differential mechanism, where three motors, housed in the upper torso, cooperate together to actuate a universal 3DOF mechanism. While performing the motion of the shoulders, the motors are not moved. The shoulder joint is a cable differential mechanism with a coupled transmission system (see Fig. 5.4). Three coaxial motors housed in the upper-torso move pulleys to generate the spherical motion of the shoulder (see Parmiggiani *et al.* (2009); Tsagarakis *et al.* (2007b) for a more clarifying explanation). Figure 5.3 shows a bi-dimensional schematic representation of the transmission of the motion from the motors to the joints. It is remarkable here the complication that follows for the control of the motion of such mechanism. More specifically, considering the control strategy presented in Section 5.2 it is remarkable to show the sensors that are present within this mechanism. Absolute hall effect based sensors measure the angular position of the joints, after the reduction gearboxes. The resolution of these sensors is 4092 tics per round. Motor position are measured instead with the hall effect sensors that are placed internally, in the spires of the brushless motor. These sensors are used to commutate the current flowing through the spires of the motor. 3 hall effect sensors measure 8 changes in the magnetic field of the rotor while moving, corresponding to 48 ticks (on/off signals) per round. This value should be multiplied for the reduction ratio of the gearbox, to make a comparison with the joint encoder resolution (the reduction ratio is 1:100, for a total of 4800 tics per round). Motor encoders are not only used for performing the commutation of the phases, but also to perform the compensation of the *back-emf* torque of the motor.

The motor groups are brushless frameless motors (RBE Kollmorgen series) with harmonic drive reductions (CSD series with 100:1 ratio) Tsagarakis *et al.* (2007b). These

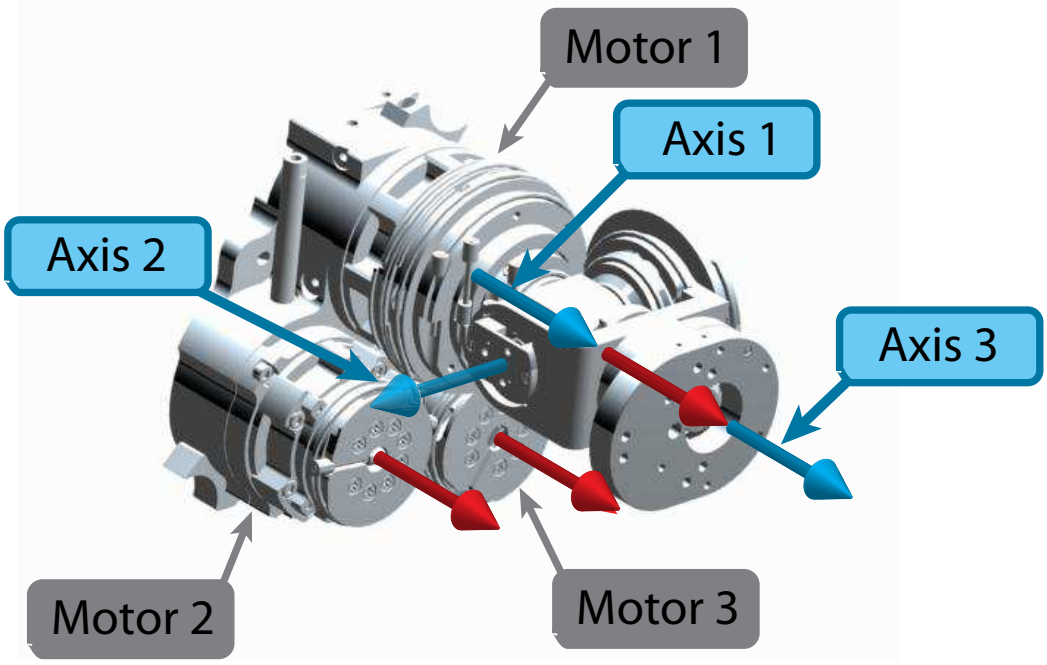


Figure 5.2: Particular of the iCub shoulder. A CAD view of the shoulder joint mechanism showing the three motors actuating the joint and the pulley system.

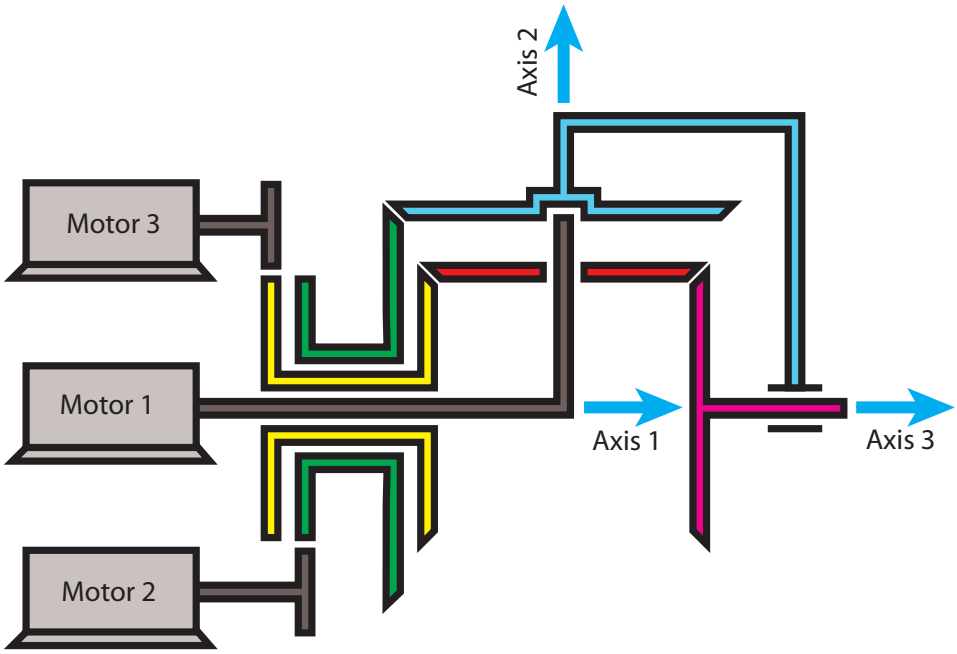


Figure 5.3: Sketch of the working principle of the mechanism of Figure 5.2.

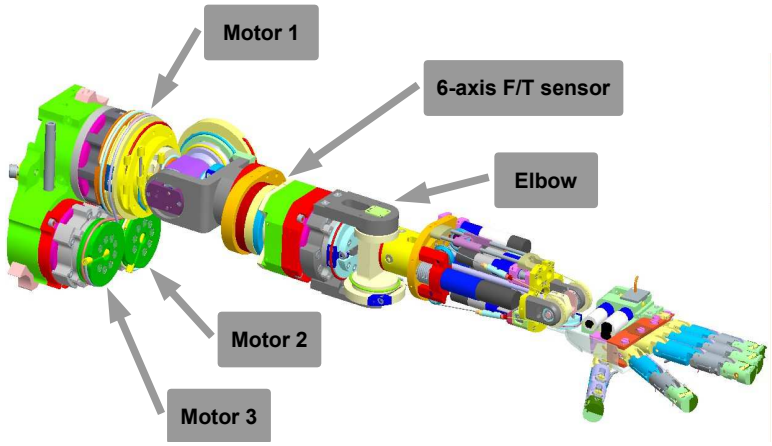


Figure 5.4: The iCub arm. A CAD view of the shoulder joint mechanism showing the three motors actuating the joint and the pulley system, the F/T sensor, the elbow and the hand.

motors are located in the torso frame. A first bigger actuator (Motor 1 in Fig. 5.4) is capable of delivering $40Nm$ and two medium power motors (Motor 2 and Motor 3) provide $20Nm$ each. The first motor actuates directly the first joint (shoulder pitch), whereas the second and third motors actuate two pulleys that are coaxial with the first motor.

The elbow joint has an independent frameless brushless motor. The joint is commanded with tendons in push-pull configuration, moving an idle pulley. Also in this case, the position of the joint is controlled through an absolute hall effect based encoder placed on the joint side, while hall effect sensors placed inside the motor spired are employed to perform the commutation and to compensate for the *back-emf* component.

The wrist is a 3DOF manipulator. The roll movement (i.e. the movement competing for the pronosupination of the wrist) is achieved by a single brushed motor directly coupled to the forearm. The pitch and yaw movements instead are accomplished by two motors, which move a semi-differential mechanism through tendons.

The iCub shoulder kinematic coupling matrix is constant and depends on the ratio among the radius of the pulleys moved by each motor. This operator takes the form of

	M_1	M_2	M_3
$R_m[\Omega]$	0.664	0.698	0.698
$k_i[\frac{Nm}{Amp}]$	0.0410	0.0236	0.0236
$k_\omega[\frac{Amp \cdot s}{rad}]$	0.0410	0.0236	0.0236
$I_m[Kgm^2]$	$9.2e-6$	$9.2e-6$	$9.2e-6$
$D_m[\frac{Nms}{rad}]$	$2.09e-6$	$9.18e-7$	$9.18e-7$
$N[]$	100	100	100
$h[\frac{Duty}{V}]$	33.25	33.25	33.25
$\gamma[\frac{tics}{rad}]$	651.9	651.9	651.9

Table 5.1: Datasheet parameters of the motors actuating the shoulder mechanism.

5.3.1

$$T_{mj} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & a & 0 \\ 0 & -a & a \end{bmatrix}, \tag{5.3.1}$$

where a is a constant value which depends on the dimension of the pulleys. For the iCub shoulder $a = 40/65 \approx 0.6154$. The inverse relationship is:

$$T_{mj} = \begin{bmatrix} 1 & 0 & 0 \\ -r & r & 0 \\ -r & r & r \end{bmatrix}, \tag{5.3.2}$$

being $r = 1/a = 1.625$.

Tab. 5.1 shows the parameters retrieved from the data sheet of the motors present in the shoulder.

To implement the inverse dynamic on the coupled mechanism, as shown in 5.2, it is convenient to group the single quantities and evaluate each terms. Let us define a control input of the form:

$$U = I_U y + D_U \Theta_M + T_\tau \tau_j \tag{5.3.3}$$

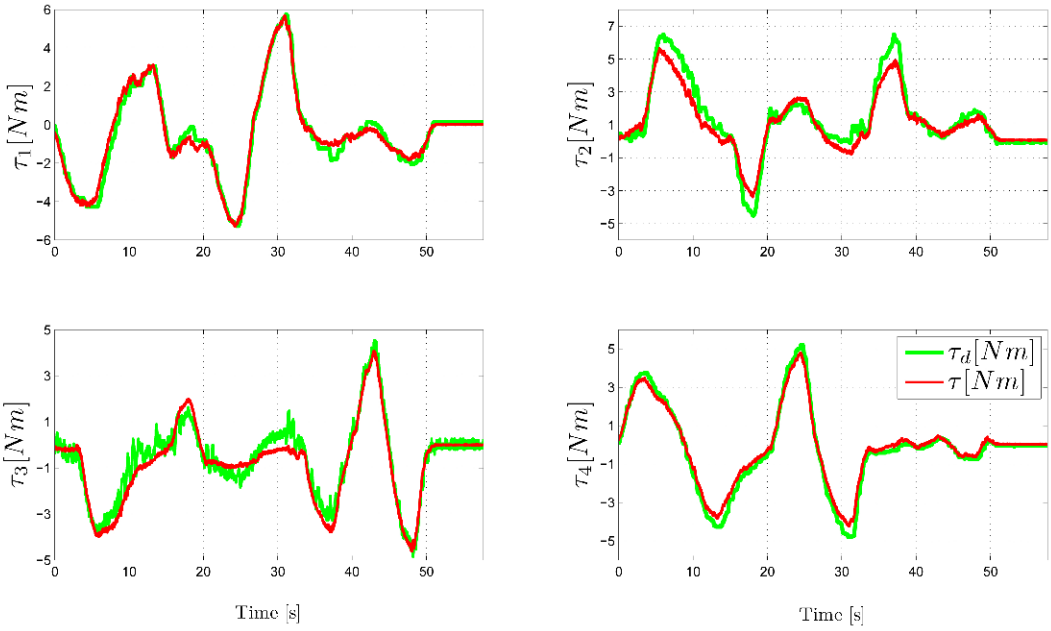


Figure 5.5: Torque control: desired (green line) vs. actual (red line) external joint torques. It is here shown the torque regulation resulting from the application of the controller proposed in Section 5.2.3. The method has been applied to four joints of the iCub right arm. The desired torque τ_d derives from the impedance regulator as described in Eq. 5.2.18.

being

$$I_U = \begin{bmatrix} \alpha\frac{NR_1I_{m1}}{k_{i1}} & 0 & 0 \\ -\alpha\frac{NR_2I_{m2}}{k_{i2}a} & \alpha\frac{NR_2I_{m2}}{k_{i2}a} & 0 \\ -\alpha\frac{NR_3I_{m3}}{k_{i3}a} & \alpha\frac{NR_3I_{m3}}{k_{i3}a} & \alpha\frac{NR_3I_{m3}}{k_{i3}a} \end{bmatrix} \tag{5.3.4}$$

$$D_U = \begin{bmatrix} \frac{1}{\gamma}\frac{R_1(D_{m1}+k_{\omega1})}{k_{i1}} & 0 & 0 \\ 0 & \frac{1}{\gamma}\frac{R_2(D_{m2}+k_{\omega2})}{k_{i2}} & 0 \\ 0 & 0 & \frac{1}{\gamma}\frac{R_3(D_{m3}+k_{\omega3})}{k_{i3}} \end{bmatrix} \tag{5.3.5}$$

After this compensation, the system is capable of behaving as preferred. It can be a stiff system, as proposed in Section 5.2.1. It can behave as a damped mass, as shown in Section 5.2.2. Or it can simulate a mass, spring and damper system, as in the case of Section 5.2.3.

The control strategy employed to present some of the results obtained with the iCub robot is the one of Section 5.2.3, shown in Eq. 5.2.18. This control strategy imple-

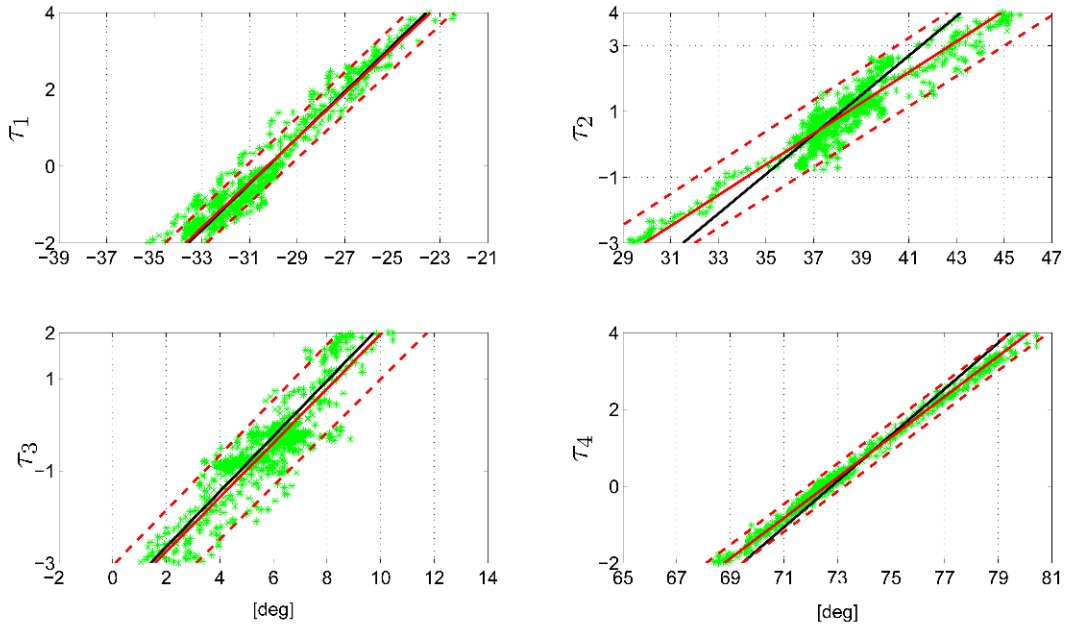


Figure 5.6: Impedance control: desired (black solid line) and measured (red solid line) stiffness resulting from the application of the impedance controller Eq. 5.2.18 to four different joints of the iCub right arm in $q_d = [-30, 37, 6, 73]^\top$. The measured line is the result of linear fitting the measured data points (represented by green dots). A 95% confidence interval for the measured stiffness is represented with red dashed lines.

ments both torque control, and assigns a reference torque such as the overall behavior of the system is the one of a mass, spring and damper mechanism. Figure 5.5 shows the torque tracking while the robot is interacting with the environment.

An external force is applied at the end-effector of the robot and is measured with the FTS embedded in the third link of the arm, just after the shoulder. The torques are measured through the algorithm of Chapter 3 and Chapter 4. These *virtual joint torque measurements* are used as feedback to perform the control.

The control is performed on the motor control boards at the rate of $1kHz$. Chapter 6 will show how these information are computed, and the path the information follow in order to obtain a measurement to perform feedback control on the motor control boards.

The commanded torques of Figure 5.5 depend on the desired impedance behavior of the system. In particular, they refer to the stiffness control reported in figure Figure 5.6.

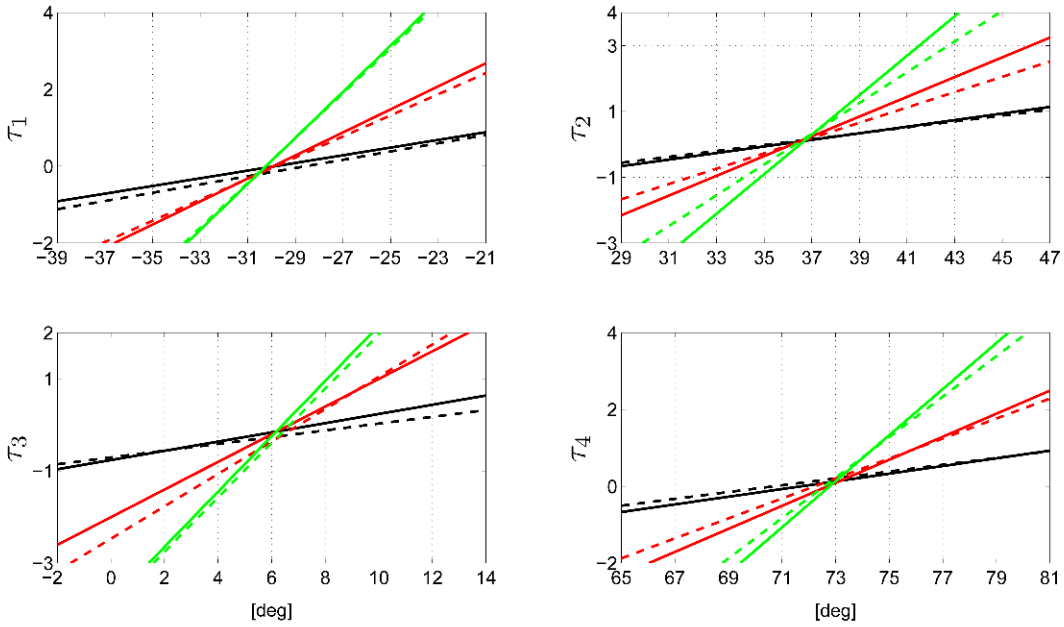


Figure 5.7: Impedance control: desired (solid lines) and measured (dashed lines) stiffness resulting from the application of the impedance controller Eq. 5.2.18 on four joints of the right arm in $q_d = [-30, 37, 6, 73]^\top$. Three different stiffness have been simulated: $k_d = 0.1\frac{Nm}{rad}$ (black lines), $k_d = 0.3\frac{Nm}{rad}$ (red lines), $k_d = 0.6\frac{Nm}{rad}$ (green lines).

Referring to Eq. 5.2.17 a desired K_d have been set for the shoulder and elbow joints equal to $K_p = 0.6[\frac{Nm}{rad}]$. Rearranging the points of Figure 5.5 together with encoder data, the validation of the method has been performed.

In Figure 5.7, the experiment have been repeated assigning different values to the desired stiffness K_d. It is remarkable the possibility to vary this parameter through high level software commands.

6

Hardware and Software Architecture

6.1 YARP

YARP (Yet Another Robot Platform) is an open-source software framework that supports distributed computation under different operative systems (Windows, Linux, Mac) with the main goal of achieving efficient robot control (Metta *et al.*, 2006). Yarp is a set of open source, OS-independent libraries that support hardware and software modularity.

Hardware modularity is obtained by defining interfaces for classes of devices in order to wrap native code API. In this way, a change in hardware requires only a change in the API calls. In this sense, YARP facilitates code reuse and modularity by decoupling the programs from the specic hardware (using Device Drivers) and operative system (relying on the OS wrapper given by ACE (Schmidt, 2003; Schmidt & Huston, 2002)).

Yarp supports also software modularity, providing an inter-process communication protocol based on ports, which allows the user to subdivide the main task of the robot in simple, reusable modules, each of them providing specific functionalities (e.g. object tracking, grasping etc). It provides an intuitive and powerful way to handle inter-process communication (using Ports objects, which follows the Observer pattern (Gamma *et al.*, 1994)). The user application is then obtained by interconnecting at run-time these software modules, which can also run on different machines on a common network, in order to obtain more complex behaviors.

Moreover, achieving visual, auditory, and tactile perception while performing elaborate motor control in real-time requires a lot of processor cycles. The only practical

way to get those cycles at the moment is to have a cluster of computers.
Furthermore, YARP provides mathematical (vectors and matrices operations) and image processing (basic Image class supporting IPL and OpenCV) libraries.
This software framework helped us to construct iCub virtual force sensors and control system as a collection of interconnected independent modules, running on different machines and exchanging data and control signals. The first reason to do so is a practical one: one single CPU, although powerful, can never be enough to cope with more and more demanding applications. Then, smaller subsystem are easier to be maintained and updated, and their employment makes the overall system cleaner. Moreover, when general enough, a single module can be connected to multiple modules, and reused in different contexts.

6.2 iCub

This section presents the electronic hardware components of the iCub robot (which has been introduces in Chapter 2). The main hardware components are presented section in 6.2.1, while the way the software is interconnected is shown in section6.2.2.

6.2.1 The iCub Hardware Architecture

A cluster of standard PCs (Intel Core2 Q9550@2.83Ghz) and a Blade system (Primergy RX200 server with 6 additional blades, Intel Xeon @2.00GHz) are interconnected through a 1GB ethernet and constitute the core of the brain of iCub. A server (Intel 1630 with double xeon5520@2.26Ghz) allows the connection between these computers and user PCs. These machines are generally dedicated to the high-level software computation, which is more demanding (e.g. coordinated control, visual processing, learning, cartesian interfaces). Low-level motor control is implemented on the DSPs embedded on the boards which are present on the robot body. All the high level software have been written using YARP (Metta *et al.*, 2006), as will be shown in section 6.2.2. Low level code runs on Freescale DSP56F807 which is built with Freescale CodeWarrior Development Studio, which is a complete integrated Development Environment (IDE) that provides a highly visual and automated framework to accelerate the development of embedded applications (Freescale, 2010). Table 6.1 shows the main

hardware components constituting the iCub *sensori-motor* system[1].
Motors and encoders are connected on the motor control boards, which provide the current necessary to move the motors, acquire the encoders position and allow the basic calculation for precise motion control (e.g. the coupling of the motors of the iCub shoulders, shown in Section 5.3, is performed on the DSP). The motor control boards and the force/torque sensors are connected through CAN-bus lines. In particular, seven CAN-Bus lines allow the communication between the motor control boards, force sensors (if present), and a central control unit. This network of CAN lines converges to an electronic board, namely the *cfw2*. This board manages the flow of information between the sensori-motor system of the robot, and its central control unit. In particular, the *cfw2* board is characterized by a set of 10 can bus connections, 2 firewire ports and 2 microphone signal conditioning, whose information flow on a unique pc104 standard. It implements a fast CAN communication (full send and receive bandwidth: 6-8 messages/1ms). The custom CAN protocol used by the DSP boards provides sanity check messages. Every fixed amount of time (currently 5 seconds) each DSP board is expected to broadcast a message describing its current state (sensors status, external faults, communication failures, overloads etc. etc.) thus allowing the main control unit, the *PC104* to have a complete description of the entire *sensori-motor* system. The *cfw2* boards also provide ports for the acquisition of camera data and for the 3DOF orientation tracker, through Firewire ports.
A PCI interface allows the communication between the *cfw2* and the *PC104*. The *PC104* board mounts the central control unit of the robot, actually an Intel Core 2Duo $@2.16Mhz$. Here runs the low level software interface to the *cfw2*, and thus to the data flowing through the CAN networks and the Firewire ports. The YARP interfaces for the devices are here used to allow the communication with the boards, as will be shown in 6.2.2 Finally, the *pc104* is connected to a network of cluster and computers to allow the communication between the robot and the user, where an Intel 1630 performs server operation.

[1]Table6.1 shows only the main hardware components (motors and sensors) which have been used within this work, and the main components of the of the robot perceptive system

Sensor/Actuator	Brand	Model	Position	Main Specifications
Brushless DC motor	Kollmorgen	RBE-01210-A Frameless	Shoulder, Elbow, Legs, Torso	$Tc = 0.115Nm$, $Ic = 5.41A$
Motor Control Board	Custom IIT	BLL - BLP	Torso, Legs	2 DC Brushless motor control boards
Brushed DC Motor	Faulhaber	DC Micromotors 12xx...G Series	Hands, Head	Motor, gear box, encoder; $0.2 - 0.6mNm$
Motor Control Board	Custom IIT	MC4 - MCP	Torso, Arms, Head	4 DC Brushed
Absolute Encoder	Honeywell	SS495A1	Joint Axis	Hall effect sensor, $8.7mA$, $4.5 - 10.5Vcc$, $p.1.3mm$
3 DOF Orientation Tracker	Xsens	MTx - Miniature 3D inertial tracker	Head	12bits, $1.7 - 5g$,
6-Axis Force/Torque Sensor	Custom IIT	-	Arms and Legs	16bit, Microchip 16Fxx Micro-Controller,16 bit A/D converter @1kHz
Cameras	Point Gray	DR2-03S2C-EX-CS - Videocam Dragonfly2 color extended version	eyes	1640x480, 1/3, CCD

Table 6.1: The hardware components of the iCub *sensori-motor* system.

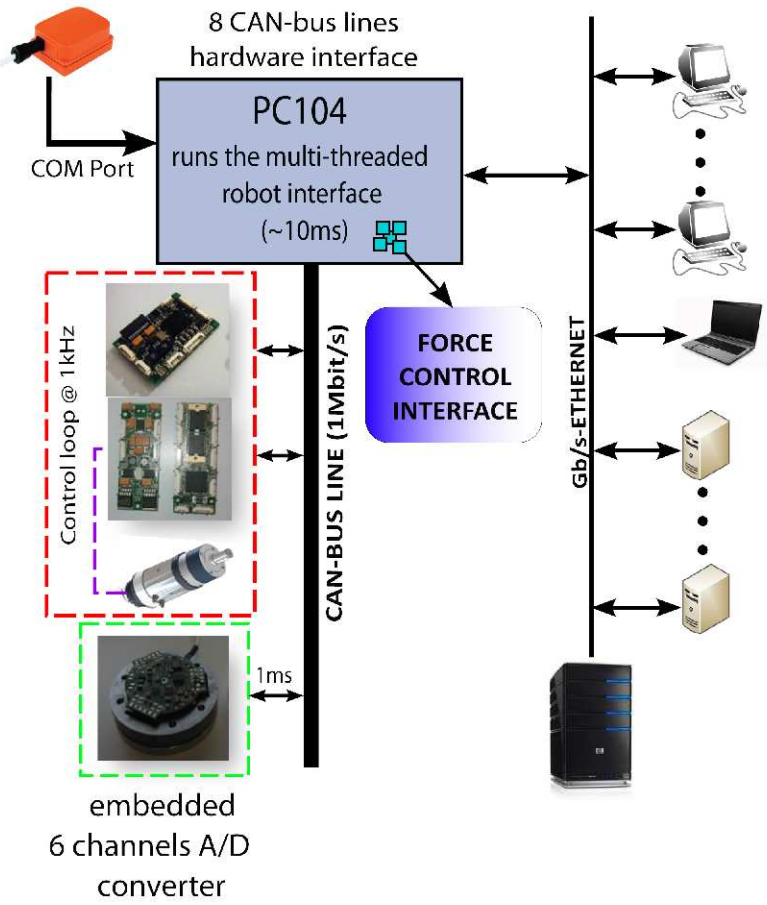


Figure 6.1: iCub robot hardware architecture.

6.2.2 The iCub Software Architecture

The iCub software architecture comprises a set of YARP executables, typically connected by YARP ports. Each executable, which is called *YARP Module*, or just *module*, runs on a PC, a blade or on the *pc104*, which is connected to the same local network. The iCub software architecture is a repertoire of *modules* which communicate through ports, to constitute the cognitive architecture of the robot. At the basis of this cognitive architecture, one *module* is dedicated to the communication and the sharing of the information with the robot hardware.
As shown in 6.2.1 the robot iCub is constituted with a repertoire of very basic motion and sensing capabilities, such as encoders, 3DOF Orientation Tracker, Force/Torque sensors, cameras, motors. The motion of the motors and the measurement of the sensors are interconnected both from the low and high level. On the low level, the hardware connection and the DSP boards define the correspondence between the motor and its associate encoder. On an higher level, a module called *iCubInterface* wraps the information coming from the motor control boards into the parts of the robot. As an example, the robot can associate the motor to move and the encoder to read directly on its motor control board, while the *iCubInterface*, defines that the information of motor i, coming from the CAN network j, moves the joint h of the part k. In this sense, the *iCubInterface* manages the information which flow through the *cfw2* board and opens ports to share these information with the user and other modules. It allows the user to monitor the state of the boards, to send commands and to read these information. Through these ports, the user can communicate with the robot.

6.2.2.1 iCub Modules

The software architecture of the iCub robot is made of YARP modules. A module is an executable that performs a specific task. Moreover, its interface is defined in terms of YARP ports. YARP allows to connect networks of modules, which constitute the behaviors of the robot. Figure 6.3 depicts an example of robot software architecture. The immediate purpose in developing the software architecture is to create a core software infrastructure for the iCub so that it will be able to exhibit a set of target behaviours for an Experimental Investigation (see `http://eris.liralab.it` (2010)). This

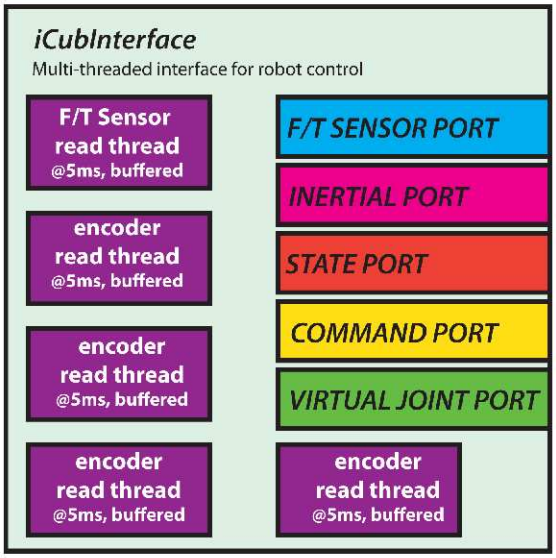


Figure 6.2: Software interface, namely *iCubInterface* that allows the communication with low level devices (API) and allows the communication with the user through interfaces.

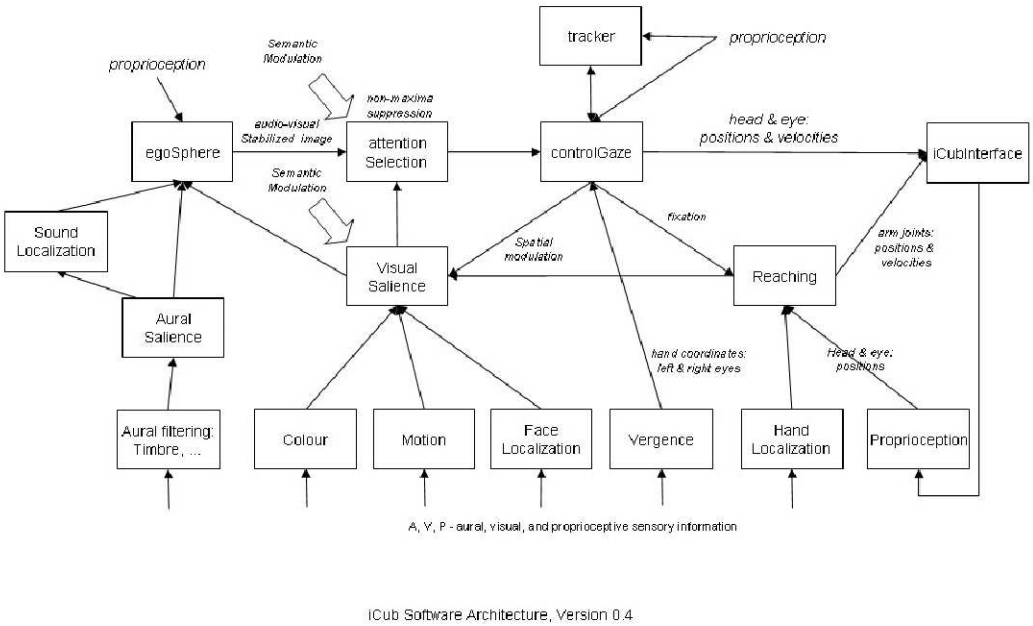


Figure 6.3: An example of YARP modularity for building behaviors.

picture refers to the last version of the iCub software architecture and is here reported to give an idea of the incredible possibilities given by the YARP framework. Next sections show how we have adopted the modularity granted by the YARP middleware, to enlarge the sensorial system of the iCub. We in fact defined a set of modules to perform force control and gravity compensation. More specifically, this framework enlarge the perception of the interaction forces, allows to obtain *virtual joint torque measurements* and, through low level implementation of appropriate control strategies allow to obtain compliant behaviors of the platform, while performing tasks.

6.3 Force Control

It has been shown in Chapter 4 that the iCub robot, version $v1.0$ to $v1.3$ is not provided with joint level torque sensors. Starting from version $v1.1$, proximal F/T sensors have been employed, one for each arm and leg. These sensors allow a great variety of information, and enlarge the tasks the robot can perform, but in particular improve the perception of the robot of the outer world. It has been shown in Chapter 3 and 4 how proximal FTSs can be exploited, together with the information of an artificial skin (see Cannata *et al.* (2008a)) to have a precise, rather than complete and distributed information about the interaction occurring during the tasks the robot performs. Moreover, from these information, it is possible to retrieve torque measurements. Torque measurements at joint level is very important for robots. They allow to obtain active compliance. Active compliance is of fundamental importance for a humanoid robot. A humanoid robot, can not be purely position controlled. It is supposed to perceive its surrounding environment, in terms of force and touch perception. It should be compliant when it explore its surrounding. In this framework, in fact, it is necessary that the robot regulates its motion behavior at joint level, as a function of the perceived force input.

In this section, we show that the framework which has been introduced in sections 6.1 and 6.2 can also be used to build and improve the perceptual architecture, where possible, from an hardware and software point of view, but also from a computational point of view. It has been used the iCub software framework to create an architecture which enlarge the perceptual capabilities of the robot, in terms of touch and force perceptual capabilities. We have thus built modules for:

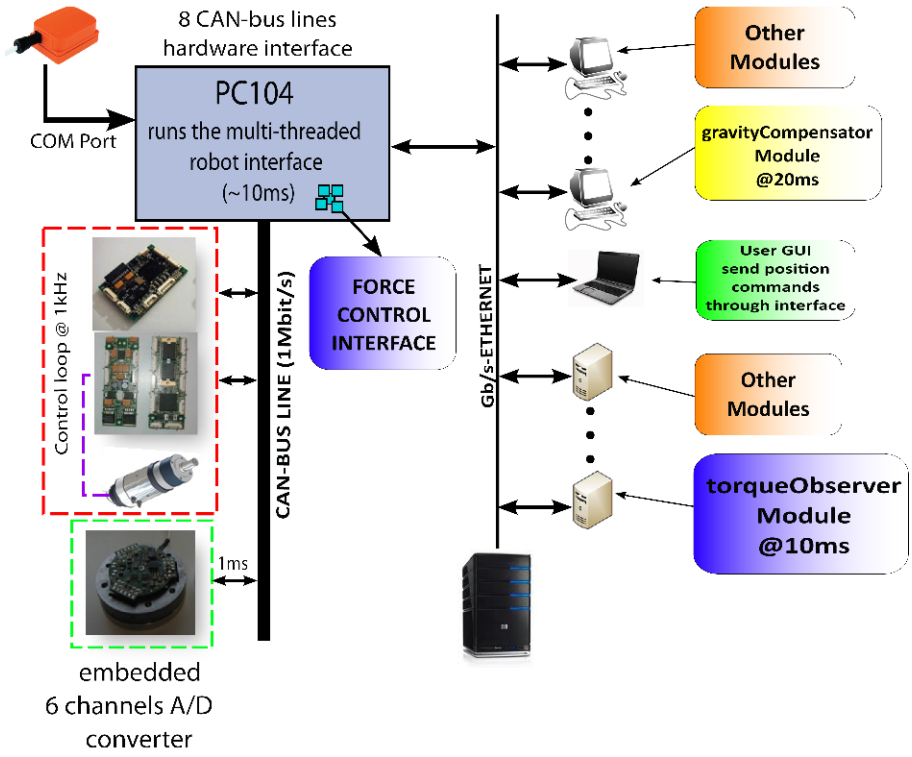


Figure 6.4: Hardware architecture with specification of the modules and devices involved for the implementation of *virtual joint torque measurements* and control.

- contact detection: the robot should detect critical situation of contact with the environment.
- virtual joint torque measurements: joint torque measurements allow the implementation of joint compliance and torque regulation.
- gravity compensation: necessary to improve the performances of motion and reaching tasks in terms of trajectory tracking and steady state position error

these modules are described in next sections in terms of implementation issues, assumptions, reliability and performances (timing and delays).

6.3.1 Contact Detection

Contact detection is the basic perceptual behavior the robot is supposed to have. Basic contact detection is achieved by defining a threshold on the robot model error whichas-

sures that the robot is not interacting with the environment. Different algorithms have been developed in literature to achieve this kind of task. Most of them are complex algorithms which have the goal to reduce the interval over which the probability of a contact is high.
The iCub robot has been provided with a basic contact detection algorithm which is based on a threshold which represents a confidential interval, determined by a statistics of the difference between the model and the actual measurements of the FTSs, such that:

$$\begin{cases} \text{if } \|w_{measured} - w_{model}\| > \eta \Rightarrow \xi = 1 \\ \text{if } \|w_{measured} - w_{model}\| \leq \eta \Rightarrow \xi = 0 \end{cases} \tag{6.3.1}$$

being ξ a boolean number defining whether a contact occurs ($\xi = 1$) or not ($\xi = 0$).

6.3.2 Providing Virtual Torque Measurements

The iCub versions from $v1.1$ to $v1.3$ mount 6-axis F/T sensors on each arm and leg. Their position is proximal with respect to the position of the end-effector. As an example, the F/T sensors which are mounted on the arm are placed immediately after the $3DOF$ spherical joint of the shoulder. On the legs, it is placed close to the $2DOF$ revolutionary joint of the hip.
The aforementioned versions of the iCub lack of joint torque sensors, and thus, a *virtual torque measurement* should be provided in order to send this information to the user and to the control boards.
We define here a *virtual measurement*, a data which flows through the CAN-bus network of the robot, whose source of information is not one of the sensors of the robot (actually a joint torque sensor, in this case) but it is predicted outside the actual robot hardware architecture, through software estimation. The *virtual measurement* is thus not directly acquired by one hardware component which is dedicated to perform the measurement of physical quantity. These sets of value, in fact, are directly related to one or more measurements which are performed by real sensors, which are part of the robot hardware architecture. In other words, these sensors does not give a direct information about the quantity to be estimated. Nevertheless, through these sensors and through an estimation of the robot parameters, it is possible to have an estimation of the actual quantity which lacks of a proper sensor measurement.

Virtual joint torque measurements follow this principle. In chapter 3 we have shown the formalism and the assumptions which are at the base of this method. In short, starting from the measurements of a $3DOF$ orientation tracker and joint encoders, we can obtain the *virtual measurements* of links frame velocity and acceleration, given the links length. On the other side, exploiting the measurements of 6-axis F/T sensors and with an estimation of the links dynamical parameters (e.g. through a CAD model), we can address the problem of the estimation of the actual joint torque, through a model based approach based on enhanced oriented graphs (see section 3.3).
The computation of the torques of a humanoid robot requires, of course, a lot of effort in terms of CPU demand, and also require an abstract layer to reduce the complexity of the code. These calculation cannot thus be performed on DSP, where only C code is allowed, and whose resources in terms of performance and .... are limited. We thus decided to perform these calculation on the blades which are physically connected to the robot through a 1GB Ethernet network. Moreover the *virtual measurements*, which are evaluated in a module running outside the *pc104* (see section 6.2.1) have a direct connection to the CAN-bus network through a dedicated connection with the *iCubInterface*, through YARP ports.
The *iCubInterface* module runs a thread which interacts with the low level CAN API. This thread generates the CAN messages to send to the boards through the CAN-Network. The messages contain both the ID of the board which sends the messages and the ID of the board which will receive it. It is this only required to the motor control boards to know the ID of the measurement they have to exploit, that the approach becomes independent on the sensor employed to perform the measurement. In other word, if a joint torque sensor is present, a data acquisition board will send through the can bus a message with a proper ID. Otherwise, a module running on an external machine will send the *virtual measurement* to the *iCubInterface*, which will convert this message to a CAN bus message with proper ID.
It thus become sufficient that the motor control boards which receive the messages have knowledge of the ID of the board which sends the message they are waiting for, in order to have the torque data. In this way, the method become transparent at the sensor level, from the point of view of the boards. This means that if the sensor is actually present, the motor control board will read its measurement, and no need to have a *virtual torque measurement*. If the robot lacks of the actual sensor, the *virtual*

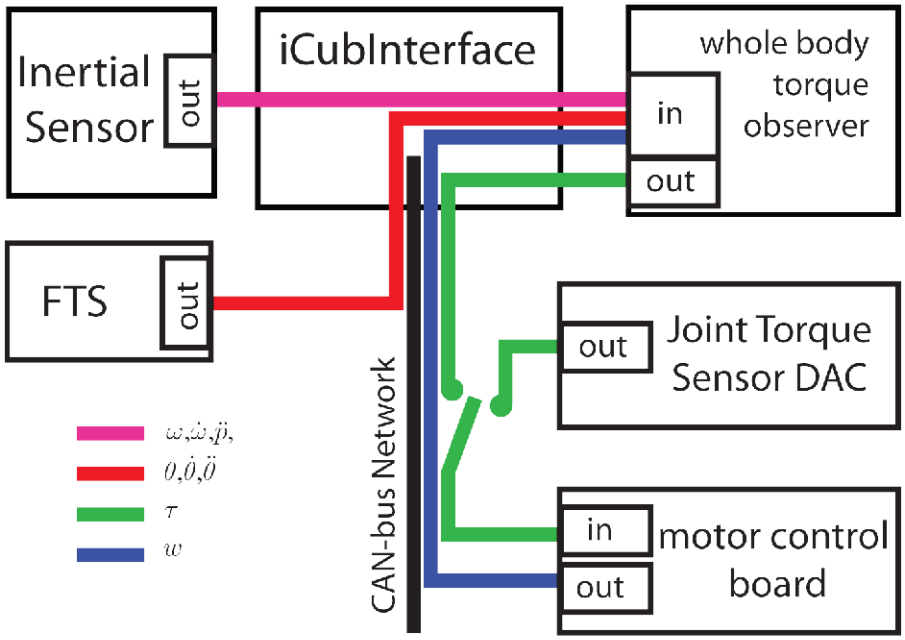


Figure 6.5: Scheme of the software architecture that allow to perform the enrichment of the iCub haptic sensory system. The *wholeBodyTorqueObserver* module takes information from the inertial sensor and FTSs to perform the computation of joint torque to send to the motor control boards, that perform the control. The low level software architecture allows to chose the messages to read between the *virtual measurements* and actual measurements from joint level torque sensors.

measurement employed.

The Whole Body Torque Observer The YARP module which perform the *virtual torque* estimation has been called *WholeBodyTorqueObserver* Fumagalli (2010). This module employs the methods provided by the *iDyn* Ivaldi *et al.* (2010) library to perform the computation of the *virtual joint torque measurements*, so to provide an estimation of the actual joint torque on 32 over 53 degrees of freedom of the iCub robot. The iDyn library is a set of classes which follow the rules presented in section 3 to estimate torques, to calculate gravity contribution and, more in general, to perform inverse dynamic calculation. This library also provides classes which specifically wrap the joints of one part of the robot (e.g. the iCub arm), more parts together (e.g. the upper torso joints, which include the head and the two arms), and even the entire structure

of the robot (two $6DOF$ legs, two $7DOF$ arms, a $3DOF$ torso and a $3DOF$ head). It allows inverse dynamic calculation performed over multiple branched chains, and specifically it provides classes for the iCub robot, as reported in section 4.3.
As previously mentioned, the *wholeBodyTorqueObserver* module performs the estimation of the actual joint torques, employing methods for inverse dynamic calculation which have been adapted to multiple branched chains of links. Moreover, it uses data from the $3DOF$ orientation tracker placed on the iCub head to initialize the kinematic flow of information. The inertial sensor in fact, is necessary to have a measurement of the absolute Cartesian velocity and acceleration, both linear and revolutionary, to calculate the other links kinematic quantities. On the other hand, the module take the data from the four F/T sensors to perform wrench computation, and thus also to compute the *virtual torque measurement*.
The evaluated torque data are then sent to the *iCubInterface*, which provide specific ports dedicated to the *wholeBodyTorqueObserver* module for virtual torque data acquisition.
Finally, the *iCubInterface* module send the information to the motor control boards, as presented previously in section 6.3.2.

6.3.3 The Gravity Compensator

Another module which is required to achieve high level position tasks with force feedback is the so called *gravityCompensator*. This module has the task to perform an estimation of the gravitational component of the robot dynamic. The estimation of this quantity is required to improve the performance of the position control when force feedback is activated. In particular it reduces the steady state position error when the robot joints control modality is the impedance control mode, following the formulation presented in section 5.2.3.

In this context, the gravitational component is given to the low level control boards as a reference torque the controller should track. To allow this, encoder values rather then the absolute orientation of the robot position (which is obtained by the values of a

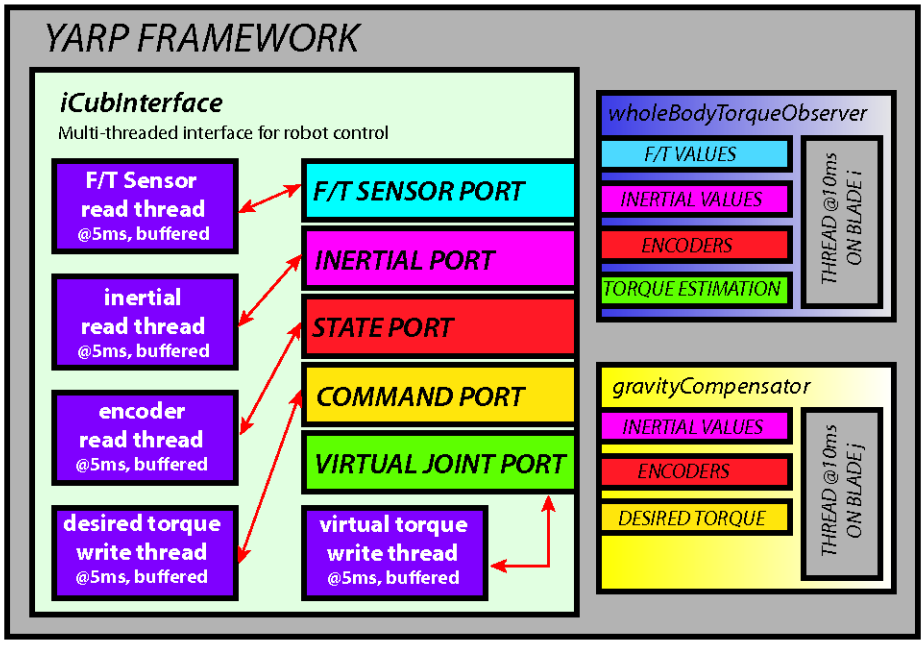


Figure 6.6: Overall software modules that allow the enrichment of the information to the iCub robot.

$3DOF$ orientation tracker) are required. Considering the equation of a manipulator:

$$\tau = M(\theta)\ddot{\theta} + \eta(\dot{\theta}, \theta) \tag{6.3.2}$$

being $M(\theta)$ the matrix of inertia of the manipulator, and $\eta(\dot{\theta}, \theta)$ the vector of centrifugal, Corioli's terms, and which also includes the gravitational component.
The gravitational term can be found by Eq. 6.3.2 by substituting $\dot{\theta} = 0$. This quantity can thus be used to obtain an estimation of the gravitational term to be used as a reference to the control strategies presented in Section 5.2.2, 5.2.3 and Section 5.2.4.
Following this procedure, the gravitational component thus become a reference torque of the form:

$$\tau_d = \tau_g = \hat{\eta}(0, \theta_j) \tag{6.3.3}$$

being $\hat{\eta}$ the estimation of the gravitational, centrifugal and Corioli's vector of Eq. 6.3.2.
From the point of view of the software/hardware architecture, the reference gravitational component is calculated on a blade. The control modalities allow to use methods which communicate with the control boards through ports of the *iCubInterface*. These values are then read by the boards and interpreted as reference torque of specific control modalities (e.g. impedance and torque modes)Also here, the software connection is presented in figure Figure 6.6 where the yellow boxes are connected together through yarp ports. The module which performs the estimation of the gravitational component runs on a blade Intel Xeon @2.00GHz, where each computation is performed with a rate of 10ms.
When applied to active compliance control strategy, the low level control performs the tracking of the reference torque considering also the torque necessary to contrast the gravitational component.

Conclusions

During the last decade, trends in robotics foster research in the development of capabilities and skills which can make robots autonomous and safe (i.e. not dangerous). The human and robot coexistence in the same physical space require exploration, adaptation and learning of the autonomous system in order to create its knowledge of the effects of an action. The representation of the environment and the perception of the interaction are fundamental in this framework. Force information are of primary importance during the learning phase, as they become part of the experience of the autonomous system. Moreover, these information are necessary during the exploration process, also to prevent dangerous situation due to collision, through control.

In this thesis it has been shown a method which allows, through distributed proximal force sensors, inertial sensors and artificial skin, to increase the perceptual capabilities of the robot iCub. It has been shown that, under some assumptions, the method allows to have an estimation of the external force on any point of the robotic structure and, moreover, it allows to have an estimation of internal forces. These quantities have been used for control purposes. Impedance control and torque control at joint level has been implemented. Backdrivability performances have been risen through a model based approach that canceled the main source of dissipation of the iCub motors.

The software architecture and the modules that allow to perform excellent basic low level compliant behaviors has also been presented.

The lack of the artificial skin over the robotic structure did not allow to perform qualitative experiments of the dynamism of the method. This point will be achieved in near future works.

This thesis focused on the exploitation of force information with the goal of creating a framework for the exploration process. The work shown here is meant to supply for the necessity of increasing the perceptual capabilities of generic robotic systems for

research in autonomous cognitive systems, but also for extending the representation of the generalized forces that arise in a physical human robot interaction scenario.

Improving the Estimate of Proprioceptive Measurements

In this respect, a possible algorithm for computing the better estimate of the kinematics, given the multiple sources, is briefly reported in Alg. 3. Basically, given a set of K kinematics sources ▼, which for brevity we name $\kappa_1, \ldots, \kappa_K$, Alg. 1 is solved K times. At each time k, κ_k is the only kinematic source which is not being removed from the EOG, and then the only ▼ in the graph. The solution of the EOG K times yields a set of conditional estimates $\omega_{j|\kappa_1}, \ldots, \omega_{j|\kappa_K}$, $\forall j$ (analogous considerations hold for $\dot{\omega}$ and $\ddot{p}$), which can be used by classical filters to provide the better estimate (e.g. maximum likelihood filters, Kalman filters *etc*). The analysis and evaluation of the possible filters has not been analyzed in this thesis, but future works might deal with this sort of estimation and improvement of internal robot perception.

Algorithm 3 Fusion of multiple kinematic sources

Require: EOG, $\kappa_k = [\omega_k, \dot{\omega}_k, \ddot{p}_k]$, $k = 1, \ldots, K$
Ensure: $\hat{\omega}_i$, $\hat{\dot{\omega}}_i$, $\hat{\ddot{p}}_i$, $\forall i$
1: **for all** $k = 1 : K$ **do**
2: Attach a node ▼ for κ_k
3: Compute $\omega_{i|\kappa_k}, \forall i$
4: **end for**
5: Compute $\hat{\omega}_i =$ filter* $(\omega_{i|\kappa_1}, \ldots, \omega_{i|\kappa_K})$

* filter is a generic filter for data fusion from multiple sensors

Acknowledgments

I acknowledge almost everyone

References

ALAMI, R., ALBU-SCHAEFFER, A., BICCHI, A., BISCHOFF, R., CHATILA, R., LUCA, A.D., SANTIS, A.D., GIRALT, G., GUIOCHET, J., HIRZINGER, G., INGRAND, F., LIPPIELLO, V., MATTONE, R., POWELL, D., SEN, S., SICILIANO, B., TONIETTI, G. & VILLANI, L. (2006). Safe and Dependable Physical Human-Robot Interaction in Anthropic Domains: State of the Art and Challenges. In A. Bicchi & A.D. Luca, eds., *Procceedings IROS Workshop on pHRI - Physical Human-Robot Interaction in Anthropic Domains*, Beijing, China. 5

BICCHI, A. & TONIETTI, G. (2004). Fast and "soft-arm" tactics [robot arm design]. *Robotics Automation Magazine, IEEE*, **11**, 22 – 33. 9

BROOKS, R.A., BREAZEAL, C., MARJANOVIC, M., SCASSELLATI, B. & WILLIAMSON, M.M. (1999). The cog project: Building a humanoid robot. In *Lecture Notes in Computer Science*, 52–87, Springer-Verlag. 9

BRUNER, J.S. (1968). *TProcesses of cognitive growth: infancy*. Clark University Press (Worcester, Mass). 12

CANNATA, G., MAGGIALI, M., METTA, G. & SANDINI, G. (2008a). An embedded artificial skin for humanoid robots. In *IEEE Int. Conf. on Multisensor Fusion and Integration*, Seoul, Korea. 23, 110

CANNATA, G., MAGGIALI, M., METTA, G. & SANDINI, G. (2008b). An embedded artificial skin for humanoid robots. In *IEEE Int. Conf. on Multisensor Fusion and Integration*, Seoul, Korea. 71

CHELI, F. & E.PENNESTRI (2006). *Cinematica e Dinamica dei Sistemy Multibody*. Casa Editrice Ambrosiana. 32, 35

COLGATE, J.E. (1988). *The control of dynamically interacting systems, Thesis (Ph. D.)*. Massachusetts Institute of Technology, Cambridge, MA, USA. 8

CORMEN, T., LEISERSON, C., RIVEST, R. & STEIN, C. (2002). *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edn. 63

DESANTIS, A., SICILIANO, B., DELUCA, A. & BICCHI, A. (2008). An atlas of physical humanrobot interaction. *Mechanism and Machine Theory*, **43**, 253–270. 5

EDSINGER-GONZALES, A. & WEBER, J. (2004). Domo: a force sensing humanoid robot for manipulation research. In *Humanoid Robots, 2004 4th IEEE/RAS International Conference on*, vol. 1, 273 – 291 Vol. 1. 9

FEATHERSTONE, R. (2007). *Rigid Body Dynamics Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. 41

FEATHERSTONE, R. (2010). Exploiting sparsity in operational-space dynamics. *The Int. Journal of Robotics Research*, **29**, 1353–1368. 41

FEATHERSTONE, R. & ORIN, D.E. (2008). *Handbook of Robotics*, chap. Dynamics, 35–65. B. Siciliano and O. Khatib Eds., Springer. 41, 43, 57

FREESCALE (2010). *CodeWarrior Development Tool*. `http://www.freescale.com/webapp/sps/site/homepage.jsp?code=CW_HOME`. 104

FUMAGALLI, M. (2010). *http://eris.liralab.it/iCub/main/dox/html/group__wholeBodyTorqueObserver.html*. GNU GPL v2.0. 114

FUMAGALLI, M., RANDAZZO, M., NORI, F., NATALE, L., METTA, G. & SANDINI, G. (2010). Exploiting proximal F/T measurements for the iCub active compliance. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Taipei, Taiwan. 9, 67

GAMMA, E., HELM, R., JOHNSON, R. & VLISSIDES, J.M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1st edn. 29, 103

HADDADIN, S., ALBU-SCHAFFER, A. & HIRZINGER, G. (2007). Safety evaluation of phisical human-robot interaction via crash-testing. In *Robotics: Science and System Conference (RSS 2007)*, Atlanta, Georgia. 5

HADDADIN, S., ALBU-SCHAFFER, A., & HIRZINGER, G. (2008a). The role of the robot mass and velocity in physical human-robot interaction - part i: Non-constrained blunt impacts. In *IEEE Int. Conf. on Robotics and Automation*, Pasadena, CA, USA. 5

HADDADIN, S., ALBU-SCHAFFER, A., FROMMBERGER, M., & HIRZINGER, G. (2008b). The role of the robot mass and velocity in physical human-robot interaction - part ii: Constrained blunt impacts. In *IEEE Int. Conf. on Robotics and Automation*, Pasadena, CA, USA. 5

HADDADIN, S., ALBU-SCHAFFER, A., EIBERGER, O. & HIRZINGER, G. (2010a). New insights concerning intrinsic joint elasticity for safety. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. 9

HADDADIN, S., URBANEK, H., PARUSEL, S., BURSCHKA, D., ROSSMANN, J., ALBU-SCHAFFER, A. & HIRZINGER, G. (2010b). Real-time reactive motion generation based on variable attractor dynamics and shaped velocities. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 3109 – 3116. 9

HOGAN, N. & BUERGER, S. (2004). *Impedance and Interaction Control*, chap. 19, 1–23. CRC Press. 8

INC., B.T. (2010). *The WAM arm from Barret Technology*. `http://www.barrett.com/robot/products-arm.htm`. V, 6, 7

ISHIDA, T. & TAKANISHI, A. (2006). A robot actuator development with high backdrivability. In *Robotics, Automation and Mechatronics, 2006 IEEE Conference on*, 1 –6. 8

IVALDI, S., FUMAGALLI, M. & PATTACINI, U. (2010). *Doxygen documentation of the iDyn library*. `http://eris.liralab.it/iCub/main/dox/html/group__iDyn.html`. 114

IWATA, H. & SUGANO, S. (2009). Design of human symbiotic robot TWENDY-ONE. In *2009 IEEE International Conference on Robotics and Automation*, 580–586, IEEE. 9

JANABI-SHARIFI, F., HAYWARD, V. & CHEN, C.S.J. (2000). Discrete-time adaptive windowing for velocity estimation. *IEEE Trans. on Control Systems Technology*, **8**, 1003–1009. 74

KANEKO, K., KANEHIRO, F., KAJITA, S., HIRUKAWA, H., KAWASAKI, T., HIRATA, M., AKACHI, K. & ISOZUMI, T. (2004). Humanoid robot hrp-2. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. 20

KULIC, D. & CROFT, E. (2005). Real-time safety for human - robot interaction. In *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*, 719 –724. 4

KULIC, D. & CROFT, E. (2007). Pre-collision safety strategies for human-robot interaction. *Autonomous Robots*, **22**, 149–164. 4

LUCA, A.D. (2006). Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1623–1630. 9

LUSSIER, B., CHATILA, R., GUIOCHET, J., INGR, F., LAMPE, R., OLIVIER KILLIJIAN, M. & POWELL, D. (2005). Fault tolerance in autonomous systems: How and how much. In *In Proceedings of the 4th IARP/IEEE-RAS/EURON Joint Workshop on Technical Challenge for Dependable Robots in Human Environments*, 16–18. 4

MAGGIALI, M., CANNATA, G., MAIOLINO, P., METTA, G., RANDAZZO, M. & SANDINI, G. (2008). Embedded distributed capacitive tactile sensor. In *Mechatronics 2008*, Limerick, Ireland. 23

MATURANA, H. (1970). *Biology of cognition*. Research Report BCL 9.0, University of Illinois, Urbana, Illinois. 12

MATURANA, H. (1975). he organization of the living: a theory of the living organization. 313332. 12

MATURANA, V.F., H.R. (1980). *Autopoiesis and Cognition The Realization of the Living*. D. Reidel Publishing Company, Dordrecht, Holland. 12

METTA, G., FITZPATRICK, P. & NATALE, L. (2006). Yarp: Yet another robot platform. *International Journal of Advanced Robotics Systems, special issue on Software Development and Integration in Robotics*, **3**. 29, 30, 103, 104

METTA, G., SANDINI, G., VERNON, D., NATALE, L. & NORI, F. (2008). The iCub humanoid robot: an open platform for research in embodied cognition. In *PerMIS: Performance Metrics for Intelligent Systems Workshop*, Washington DC, USA. 17, 18

MINGUEZ, J., LAMIRAUX, F. & LAUMOND, J.P. (2008). *Handbook of Robotics*, chap. Motion planning and obstacle avoidance, 827–852. B. Siciliano and O. Khatib Eds., Springer. 4

MISTRY, M., BUCHLI, J. & SCHAAL, S. (2010). Inverse dynamics control of floating base systems using orthogonal decomposition. In *IEEE Int. Conf. on Robotics and Automation*. 9

PARMIGGIANI, A., RANDAZZO, M., NATALE, L., METTA, G. & SANDINI, G. (2009). Joint torque sensing for the upper-body of the iCub humanoid robot. In *International Conference on Humanoid Robots*, Paris, France. 20, 95

PRATT, G. & WILLIAMSON, M. (1995). Series elastic actuators. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 399–406, Los Alamitos, CA, USA. 8

ROBOTCUB.ORG (2010). `http://www.robotcub.org`. 17

SALISBURY, K., TOWNSEND, W., EBRMAN, B. & DIPIETRO, D. (1988). Preliminary design of a whole-arm manipulation system (wams). In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, 254 –260 vol.1. 6, 10

SANDINI, G., METTA, G. & VERNON, D. (2007). The iCub cognitive humanoid robot: An open-system research platform for enactive cognition. *50 Years of AI*, **LNAI 4850**, 359–370. 12, 13, 17

SCHMIDT, D.C. (2003). *The adaptive communication environment*. `http://www.cs.wustl.edu/~schmidt/ACE.html`. 29, 103

SCHMIDT, D.C. & HUSTON, D.H. (2002). *C++ Network Programming: Mastering Complexity Using ACE and Patterns*. Addison-Wesley Longman. 29, 103

SCHMITZ, A., MAGGIALI, M., NATALE, L., BONINO, B. & METTA, G. (2010). A tactile sensor for the fingertips of the humanoid robot icub. Taipei, Taiwan, October 18-22, 2010. 23

SCIAVICCO, L. & SICILIANO, B. (2005a). *Modelling and Control of Robot Manipulators*. Advanced textbooks in Control and Signal Processing, Springer. 10, 40, 41, 51, 63, 66, 90

SCIAVICCO, L. & SICILIANO, B. (2005b). *Modelling and Control of Robot Manipulators*. Advanced Textbooks in Control and Signal Processing series, Springer, London, 2nd edn. 32, 38, 39

SICILIANO, B. & VILLANI, L. (1996). A passivity-based approach to force regulation and motion control of robot manipulators. *Automatica*, **32**, 443 – 447. 9

SICILIANO, B. & VILLANI, L. (2000). *Robot Force Control*. Kluwer Academic Publishers, Norwell, MA, USA. 10, 27

SISBOT, E., MARIN, L., ALAMI, R. & SIMEON, T. (2006). A mobile robot that performs human acceptable motions. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 1811 –1816. 4

SISBOT, E., MARIN-URIAS, L., BROQURE, X., SIDOBRE, D. & ALAMI, R. (2010). Synthesizing robot motions adapted to human presence. *Int. Journal of Social Robotics*, **2**, 329–343. 4

TOWNSEND, W.T. (1988). *The Effect of Transmission Design on Force-Controlled Manipulator Performance*. Massachusetts Institute of Technology, Cambridge, MA, USA. V, 6, 7

TSAGARAKIS, N., BECCHI, F., RIGHETTI, L., IJSPEERT, A. & CALDWELL, D. (2007a). Lower body realization of the baby humanoid - iCub. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, USA. 17, 24, 25, 26

TSAGARAKIS, N., METTA, G., SANDINI, G., VERNON, D., BEIRA, R., SANTOS-VICTOR, J., CARRAZZO, M., BECCHI, F. & CALDWELL, D. (2007b). icub - the design and realization of an open humanoid platform for cognitive and neuroscience research. *International Journal of Advanced Robotics*, **21(10)**, 1151–75. 13, 17, 18, 20, 24, 25, 95

HTTP://ERIS.LIRALAB.IT (2010). *The iCub user main page*. http://eris.liralab.it/wiki/Main_Page. 21, 26, 108

VARELA, F. (1979). *Principles of Biological Autonomy*. Elsevier, North Holland, New York. 12

WENG, Y.H., CHEN, C.H. & SUN, C.T. (2009). Toward the humanrobot co-existence society: On safety intelligence for next generation robots. *International Journal of Social Robotics*, **1**, 267–282, 10.1007/s12369-009-0019-1. 3

WITTENBURG, J. (1994). Topological description of articulated systems. *Computer-Aided Analysis of Rigid and Flexible Mechanical Systems*, 159–196. 41

XSENSMTX (2010). *The MTx orientation tracker user manual*. http://www.xsens.com/en/general/mtx. VI, 27, 28, 67