The iCub platform: a tool for studying intrinsically motivated learning

Lorenzo Natale, Francesco Nori, Giorgio Metta, Matteo Fumagalli, Serena Ivaldi, Ugo Pattacini, Marco Randazzo, Alexander Schmitz and Giulio Sandini

Department of Robotics, Brain and Cognitive Sciences, Istituto Italiano di Technologia
lorenzo.natale,francesco.nori,giorgio.metta,
matteo.fumagalli,serena.ivaldi,ugo.pattacini,marco.randazzo,
alexander.schmitz,giulio.sandini@iit.it

Abstract. Instrinsically motivated robots are machines designed to operate for long periods of time, performing tasks for which they have not been programmed for. These robots make extensive use of explorative, often unstructured actions in search for opportunities to learn and to extract information from the environment. Research in this field faces challenges that need advances not only on the algorithms but also on the experimental platforms. The iCub is a humanoid platform that was designed to support research in cognitive systems. We review in this paper the chief characteristics of the iCub robot, devoting particular attention to those aspects that makes the platform particularly suitable to the study of intrinsically motivated learning. We provide details on the software architecture, the mechanical design and the sensory system. We report examples of experiments and software modules to show how the robot can be programmed to obtain complex behaviors involving the interaction with the environment. The goal of this paper is to illustrate the potential impact of the iCub on the scientific community at large, but, in particular on the field of instrinsically motivated learning.

1 Introduction

Developmental robotics is a young field of research that attempts to build artificial systems with cognitive abilities (see Lungarella et al., 2003 for a review). In contrast to other, more traditional, approaches, researchers in this field subscribe to the hypothesis that cognition is not hard-coded but that, on the contrary, it emerges autonomously from the physical interaction between the agent and the environment (Weng et al., 2000; Zlatev and Balkenius, 2001). Developmental robotics is a strongly interdisciplinary field that brings together researchers from behavior and brain sciences (psychology, neuroscience), engineering (robotics, computer science) and artificial intelligence, motivated by the conviction that each field has a lot to learn from the others. Roboticists in particular have realized that the real world is too complex to be modeled and too dynamic to hope that static models are of any use. For this reason they have started to seek

inspiration from biological systems and how they deal with the complexity of the world in which they live.

The study of humans and biological systems confirms that nature solved this problem by designing systems that undergo a constant physical and behavioral adaptation. As humans we are probably the most convincing example in this respect, since learning and adaptation is so evident in the first period of our lives. For these reasons developmental approaches to robotics assume that intelligent behavior and cognition emerge autonomously through development.

Experimental evidence in the field of developmental psychology shows that motor and perceptual development happen as a result of the constant physical interaction between the body and the world (Bushnell and Boudreau, 1993; Needham et al., 2002; von Hofsten, 2004). Newborns spend a considerable amount of time exploring the world. They do so by experimenting with their own actions and by exploring correlations between what they do and what is measured by their sensory streams.

Unfortunately today's machines demonstrate learning abilities that are not comparable to the ones that we observe in animals or in infants. Learning in artificial systems is often applied to specific, pre-defined tasks and it requires a certain degree of human supervision for parameter tuning or data preparation. Finally, exploration, acquisition of new data, learning and exploitation are distinct processes. On the other hand learning in humans is something that happens continuously. Children seem to be always actively searching for learning opportunities; they show what has been defined an intrinsic motivation to engage in activities that involve exploring the environment in search for novel things to learn. Intrinsic motivations and curiosity are clearly required ingredients to design machines that are able to autonomously discover not only how to solve certain tasks (like object manipulation or perception), but also how to learn or develop new abilities based on current knowledge (see Chapter **??** and Barto et al., 2004; Kaplan and Oudeyer, 2008; Oudeyer and Kaplan, 2007).

Progress in developmental robotics and intrinsically motivated learning has been somewhat slow. In our opinion this is in part due to the fact that research in these fields requires experimental platforms whose design and construction is difficult. We list here some of the aspects that characterize human development that have deep implications in the design of the hardware and software architecture of such platforms:

- motion during development is at times dominated by exploratory unstructured behaviors (sometimes referred to as "motor babbling");
- failure in performing a task is the norm rather than the exception;
- learning is not cheap, the acquisition of new competences on the contrary require lots of trials and considerable efforts;
- the development of perceptual abilities requires sophisticated interaction with the world;
- perception is intrinsically multimodal and multisensory integration is critical for learning;
- learning is open-ended, i.e. it is an incremental process in which new abilities provide support and opportunities for gathering newer ones.

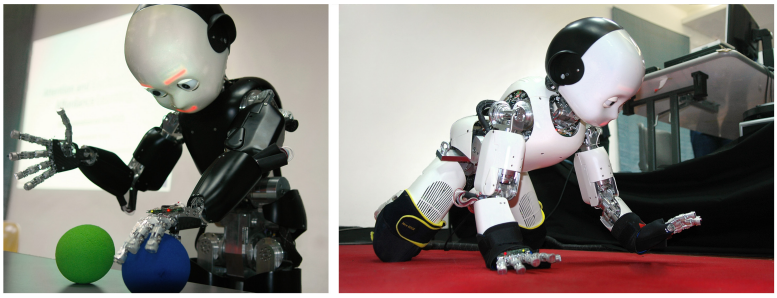

Fig. 1. The iCub.

This implies that developing robots need to be robust and able to operate for several hours (maybe days) in frequent and unpredictable physical interaction with the environment. They need to have articulated mechanical structure that allows interacting with the world in a sophisticated way. For example they should be able to locomote, manipulate objects and use tools. Their sensory system should include diverse and redundant modalities ranging from vision and sound to touch and force. Finally, their software architecture should be flexible enough to allow development of interconnected modules that receive sensory streams, perform control and exchange information. The software should be modular to support re-usability of components so that existing capabilities can work as building blocks to construct newer, more complex ones.

One of the goals of the RobotCub project[1] was the design and construction of a platform that satisfies the aforementioned constraints. The resulting platform is the iCub, a humanoid robot with 53 degrees of freedom and a rich sensory system. The iCub is an open-system: researchers have full access to the details of the platform and can customize it depending on their particular needs[2].

In this paper we provide an overview of the platform, focusing on the functionalities that we consider more interesting for researchers in the field of developmental robotics and intrinsically motivated learning. Section 2 provides an overview of the hardware platform, the mechanical structure, the actuation system and the sensors. Section 3 describes the software architecture and the motivations that have driven its design. Section 4 describes three examples of software modules in which actuators and sensors are used for controlling the interaction with the environment, namely: force control, detection of grasp using the sensors on the hand, and reaching. Section 5 demonstrates a complete behavior in which the robot grasps objects on the table. Finally Section 6 draws the conclusions.

[1] The RobotCub project was funded by the European Commission, Project IST–004370, under Strategic Objective 2.3.2.4: Cognitive Systems.

[2] The iCub software and hardware are licensed under the GNU General Public License (GPL) and GNU Free Documentation License (FDL), respectively.

2 The Hardware Platform

The iCub (Metta et al., 2008; Tsagarakis et al., 2007) was designed specifically for manipulation and locomotion (Figure 1). This is reflected on how the degrees-of-freedom (DOF) are allocated on the robot. Each arm has 16 motors, 3 of which control the orientation of the hand (roll-pitch-yaw) and 4 the shoulder and the elbow. Each hand has five fingers, and is actuated by nine motors. Thumb, index and middle fingers are independently actuated, while the fourth and fifth fingers are connected to a single motor. The joints that are not driven by dedicated motors are mechanically coupled. As usual in these cases the coupling is elastic to allow better adaptations to the objects. A motor on the palm is responsible for controlling the fingers abduction. This DOF is critical to better adapt the hand to objects with various shapes and size. A detailed description of the fingers and their actuation is reported in Figure 2.

The initial design criteria of the legs aimed to allow the robot to locomote by crawling on the floor (see Figure 1). From an early analysis it was determined that five DOF were sufficient. To allow standing and walking it was later decided to add an additional motor at the ankle. Overall in the current design the legs of the iCub have six DOF each.

Three motors control the orientation of the robot at the level of the torso. This turns out to be an important feature because it significantly extends the workspace of the arms. The iCub can bend forward to reach for a far object or turn the attention to different areas while maintaining the arms within their optimal workspace. The head comprises a three DOF neck (tilt, pan and roll) and cameras that can rotate around a common tilt and independently around pan axes to control the gaze direction and the angle of vergence.

Another design choice was to shape the robot as a three-and-a-half year old child (approximately 100 cm high). The overall weight of the robot is 22 Kg. The mechanics and electronics had to be optimized to fit in the available space. The actuation solution adopted was a combination of a harmonic drive reduction system (100:1 ratio for all the major joints) and a brushless frameless motor. The harmonic driver gears provide zero backlash and high reduction ratios in small space with low weight, while the brushless motors guarantee appropriate performance in terms of speed, torque and robustness. An important decision in this respect was to use frameless motors. These motors are provided without housing and bearings in separate components that can be integrated directly inside the structure of the robot thus minimizing size, weight and dimensions. In the majority of the cases torque is transmitted from the motors to the joints using steel tendons routed in complex ways via idle pulleys. Most of the motors are thus placed closer to the body and away from distal links. The motors of the shoulder (placed in the torso) and the hand (placed in the forearm) are examples of this. One of the advantages of this solution is that the robot has lower inertia and as a consequence turns out to be easier to control and safer during interaction with humans.

Clearly the decision to use electric motors was dictated by practical considerations, considered that at the beginning of the project (and still now, at the time

of writing) it was the only solution that met the specifications in terms of speed, torque and size. The choice of the reduction system originated from similar considerations. Electrical motors, especially when equipped with large reductions, are difficult to backdrive, and in general are easier to be controlled in position mode. This is a clear limitation for the targeted research and applications[3], so we equipped the robot with various sensors that allow measuring contact forces (torque and tactile sensors) and we implemented torque and impedance control. Section 4 in this Chapter shows some results on these topics.

The robot is endowed with a complete sensory system, which includes (Figure 3):

- Vision: two color cameras attached to the eyes. We employ commercial Dragonfly cameras from Pointgray[4]. Images are acquired by the PC104 computer mounted on the head and streamed to the network at the maximum rate of 60 Hz and resolution of 640×480 pixels.
- Sound: two microphones are mounted on the head (condenser electrets microphones). A mechanical structure similar to the human pinnae produce spatial filtering that can be used for sound localization (Hörnstein et al., 2006).
- Inertial sensor: the Xsense MTx sensor from Xsens Technology[5] contains three gyroscopes, three linear accelerometers and a compass. This sensor is mounted inside the head and provides absolute orientation and angular acceleration (the signal from the compass is quite noisy and in practice of scarce utility, probably because of the interference with the mechanical structure of the robot).
- Forces and torques: we mounted custom 6 axis force/torque sensors in the arms and legs. These sensors are mechanically similar to commercial sensors usually used in robotics (in particular the ATI Mini-45 sensor[6]). Our sensor embeds the electronics that perform signal conditioning and A/D conversion. Signals are transmitted to a canbus network at the maximum frequency of 1 kHz. In addition we have been working to embed joint level torque measurement in the shoulder (Parmiggiani et al., 2009).
- Proprioception: all motors and joints are equipped with angular encoders. It is therefore possible to recover the position of all joints, including those that are elastically coupled to the actuation.
- Touch: recent versions of the iCub mount tactile sensors on the hands. These sensors are based on capacitive technology and provide contact and pressure information. Overall each hand has 108 sensors, 12 in each fingertip and 48 on the palm (Schmitz et al., 2010).

[3] In position control potentially large forces are produced to achieved a desired position. This is dangerous when unexpected interaction with the environment occurs because the robot is learning, exploring or interacting with humans. In this scenario, as explained in Section 4.1, force control is a preferable approach.

[4] Point Grey Research, Inc.: http://www.pointgray.com

[5] Xsens 3D Motion Tracking: http://www.xsens.com

[6] ATI Industrial Automation: http://www.ati-ia.com

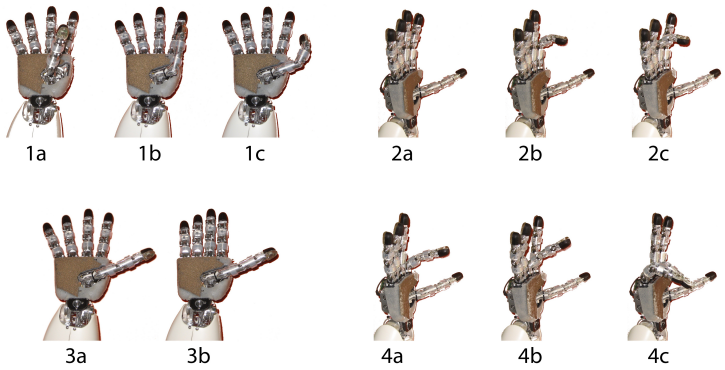

Fig. 2. Hands details. All fingers have three moving phalanges. The middle and distal phalanges are mechanically coupled to bend in a natural way. The coupling is elastic to allow better adaptation to the objects. The thumb has an additional joint that allows it to rotate at the base to oppose different fingers. Pictures 1a, 1b and 1c demonstrate the degrees of freedom of the thumb; in particular picture 1c shows the coupled motion of the middle and distal phalanges. Pictures 2b and 2c show the coupled motion of the proximal phalanges of the index and middle finger respectively (compare with 2a). Index and middle fingers can rotate at the level of the proximal phalanges (4a and 4b). All the phalanges of the fourth and fifth fingers are coupled together and are actuated by a single motor (4c). Finally 3a and 3b demonstrate the abduction of the fingers.

The sensory system is one of the key features of the iCub. The availability of a rich sensory system makes it possible to use the robot to study algorithms that exploit and integrate different sensory modalities (e.g. sensory fusion, multimodal calibration and multimodal perception to mention a few). The force and tactile sensors on the limbs enable the implementation of control strategies to monitor and regulate the interaction forces between the robot and the world. This is crucial for applications that involve autonomous exploration and learning or the interaction with humans.

3 Software Architecture

Very often academic research laboratories need robotic platforms as tools for testing and experimenting new ideas or algorithms. The robotic platforms that are built within these laboratories are not the main goal of the research, but are considered as useful tools whose development is just a preliminary effort before the real work. When the research focuses on the robot itself (e.g. when it involves the study of mechanical solutions, actuators or sensors), it only produces results that rarely go beyond the status of prototypes. In fact the research community gives reward mainly to publication of new ideas or principles. This

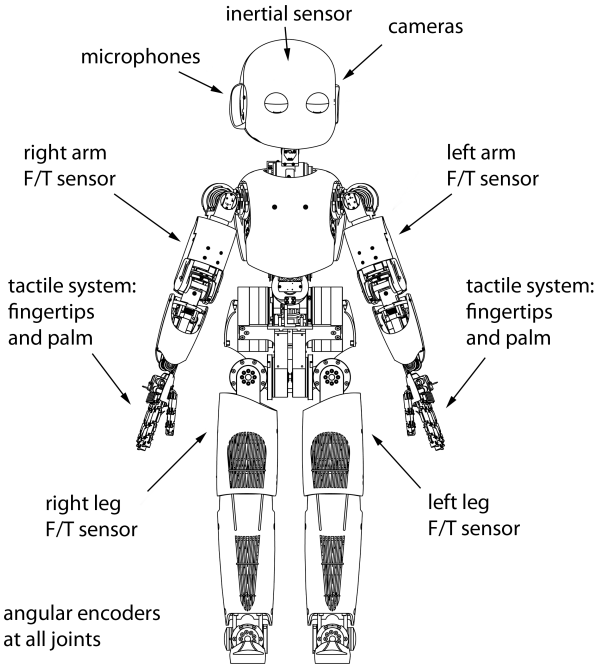

Fig. 3. The iCub sensory system.

happens for good reasons, and in this paper we do not want to criticize this approach. It is true, on the other hand, that this trend has several drawbacks that make research in robotics suffer in different aspects. The first we discuss here is the *lack of off-the shelf solutions for common problems*: with some notable exceptions (OpenCV[7], OROCOS[8] and more recently ROS[9]) it is difficult to find good implementations of existing algorithms, especially that work on a given platform and for applications in humanoid robotics. Often a researcher working on a specific topic has to start from scratch the development of even the most basic functionalities (like control of attention or reaching). This clearly slows down research that involves the implementation of complex behaviors (like grasping or human robot cooperation) and contributes to a second fundamental problem: the *lack of a scientific methodology* that, by comparing different techniques, allows the identification and promotion of better algorithms.

The iCub software architecture intends to mitigate these problems. We review here the key design choices that drove its development: ease of use, modularity and scalability.

[7] Open Computer Vision Library: http://sourceforge.net/projects/opencvlibrary
[8] Open Robot Control Software: http://www.orocos.org
[9] Robot Operating System: http://www.ros.org

3.1 Ease of use

One of the design choices of the software platform was to reduce as much as possible the learning curve for new users. We tried to minimize the time a new user would have to spend in order to get accustomed not only with the software itself, but also with the developmental environment. Particular attention was taken to avoid forcing people to a particular development environment, be it the operating system, compiler or IDE. The iCub software is fully working on Windows and Linux (and with minimal effort on MacOS), and more importantly, a mix of the two. Users can take advantage of the platform that best suits their needs and skills. For examples psychologists find it more natural to use Windows, and it is not unusual to have to interface the robot software with off-the-shelf devices that are supported only in the Windows operating system (i.e. an eye tracker). Linux, on the other hand, appears the natural choice of people that have a more technical background. Similar considerations apply to the development environment. In addition, and to a more limited extent, the software provides interoperability with other languages like Python, Java and Matlab. Winning choices in this respect have been the use of open source tools like CMake[10], SWIG[11] and ACE[12].

3.2 Modularity

The iCub software architecture was built on top of YARP (Fitzpatrick et al., 2008). YARP is an open source software middleware that support code reuse and development in robotics. In YARP software is organized as *modules* (usually executables) that communicate using objects called *ports*. Ports are entities that receive or transmit data with each others. Modules are executed on a set of computers (*nodes*) and cooperate through ports. On top of this, YARP allows creating devices that have a standard C++ interface and support remotization of these devices across the network. Code development is intrinsically modular, functionalities are added to the system as modules that have a specific interface. With time modules that become obsolete can be easily replaced with new ones that offer the same interface. Clearly the advantage of a distributed architecture is that it is easily scalable; as long as resources are available new modules can be added to the system. Connections between modules can be created and destroyed dynamically, so the system does not need to be stopped when new modules are added or, for any reason, moved across machines. The other advantage of YARP is that the system can be made of heterogeneous nodes (i.e. nodes running different operating systems).

The robot offers a YARP interface to communicate with its motors and sensory system. All the users need to know is how to use the interface to send commands to the robot and access the sensory information. The complexity of

[10] Kitware, Cross Platform Make: http://www.cmake.com
[11] Simplified Wrapper and Interface Generator: http://www.swig.org
[12] The ADAPTIVE Communication Environment: http://www.cs.wustl.edu/~schmidt/ACE.htm

the networking, low-level hardware and device drivers are completely hidden to users. This is beneficial because it allows non-experts to use the robot and it avoids that changes to the low-level hardware have catastrophic impacts on the user code.

An good example of modularity is the iCub simulator (Tikhanoff et al., 2008). This software component is implemented using the ODE dynamics engine[13] and it simulates all the sensors, actuators and degrees of freedom of the iCub to provide the same software interface of the real robot (i.e. the simulator and the real robot provide compatible YARP ports). Software modules can be developed using the simulator and later plugged to the real robot without effort. It is important to point out that the simulator was not designed to simulate the dynamics of the robot with accuracy. Nonetheless it can be a fundamental tool for fast prototyping and testing, in particular of learning algorithms that are time consuming and require that large number of experiments are performed.

We mentioned modularity as a requirement for long term development. In the iCub architecture modules are loosely coupled and are easily developed by different people with minimal interference. At the time of writing the iCub community is made of roughly 40 active developers and a larger number of users scattered in different parts of the world. Modularity makes it possible for all these developers to coexist and cooperate within the same project.

3.3 Scalability: from Modules to Applications

The iCub software architecture defines the concept of *application* as a collection of modules that achieve a particular functionality. Examples of applications are the attention system (Ruesch et al., 2008), the reaching behavior (Pattacini et al., 2010), or the exploration of object affordances (Montesano et al., 2008) (other applications are reported in Table 1 in Section 6). Applications are obtained by instantiating a set of modules. Since modules can be configured depending on the context, an application should also store the information about how these modules are configured. In other words, applications are somewhat abstract entities that consist of a list of modules and instructions for running them.

Unfortunately, the execution of an application that is fragmented in many modules can be complicated and time consuming, especially for people who have not participated in its development. Good documentation can partially mitigate this problem, but the effort required to run a certain application has a strong influence on the probability that the application will be re-used. The risk is that developers do not perceive the advantages of a modular approach, and start developing monolithic applications. Modularity can prevent scaling from simple experiments involving a few modules to complex behaviors obtained by instantiating many modules (and maybe other applications).

The iCub software contains a *manager* that simplifies running applications. Each application is associated to an *application descriptor* (i.e. an xml file) that contains all the information required to instantiate it, including: the modules

[13] Open Dynamics Engine: http://www.ode.org

to run, the set of parameters, the name of the machines on which modules will be executed, and how modules should be connected. From this information the manager is able to execute all modules on the desired machines, configure them appropriately, and establish connections between the ports involved. In other words the manager automates common tasks like starting and stopping an application, or monitoring its status. Like modules, applications can be uploaded in the repository, documented and shared across developers. The idea is that the users do not instantiate modules individually, but rather from the application descriptors available in the repository (this process is exemplified in Figure 4).

Another limit to the growth of the system is imposed by the delay introduced by the communication between modules. Latencies are heavily dependent on the computing infrastructure and for this reason are difficult to estimate. YARP was implemented and designed to be efficient and minimize overheads, and so far it has demonstrated to behave well in this respect (we have measured that the delay introduced by a port is below 300 μs for small packets, see Natale, 2009). An important design principle driving software development in iCub is to avoid introducing coupling between the timing of the modules. The reason for this is that we want to avoid that the performance of slow modules have a negative impact on the others. Among the different communication paradigm available in YARP, we favor "streaming" communication in which no synchronization between sender and receiver is required or enforced. When large packets are sent to multiple receivers (i.e. packets containing images) we use the *multicast* protocol to minimize the associated overhead.

Finally a clear limitation to the scalability of the system is dictated by the available resources: computing power and network bandwidth. In developing modules and applications it is important to monitor the available resources and avoid saturating them. Our experience has shown that with some attention it is possible to create applications made of a large number of modules (our largest applications involve running at least 20 interconnected modules, e.g. "Imitation, Learning Object Affordances" or "Objects and Actions Learning" in Table 1). In addition since technology has been progressing at a steady pace in these aspects we believe this is unlikely to be a problem in the near future.

4 Examples of Software Modules

In the previous sections we have described the iCub hardware and software platforms. In this section we describe some software modules that exploit the available sensors and actuators to control the interaction with the environment. These and other modules are available open source and documented in the iCub software repository[14]. We report data from experiments with the goal of demonstrating some of the current capabilities of the robot that appear more relevant with respect to the topic of this paper.

[14] http://icub.org: iCub Software Documentation.

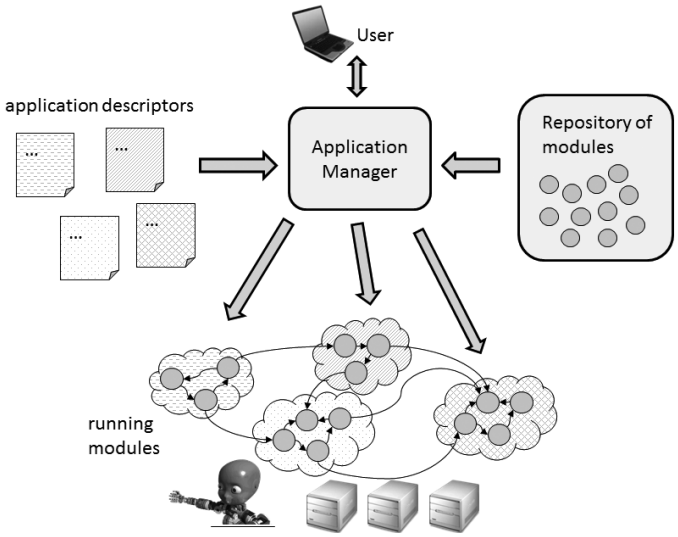

Fig. 4. Managing applications and modules.

4.1 Force Control

Force control is a well known technique that is commonly used to better control the interaction between the robot and the environment, especially in the presence of uncertainties (Sciavicco and Siciliano, 2005). Recently force control has obtained renewed importance since researchers and industries have started to propose applications that see robots working in close interaction with humans. Safety and uncertainty handling have pushed the need for robots that can control or limit the amount of force they exert on the external world (De Santis et al., 2008; Schiavi et al., 2009). Among the possible solutions that allow achieving safety (light weight designs, intrinsically compliant actuations, see Haddadin et al., 2009; Zinn et al., 2004), force control has the advantage that it does not require an increase in the complexity of the system.

We employ 6-axis force and torque (F/T) sensors, integrated within the two arms and legs. The F/T sensors of the arms are placed in the upper arm, between the shoulder and the elbow, while those employed for the legs are placed between the hip joints and the knee (see Figure 3). These F/T sensors employ semiconductor strain gages (SSG) for measuring the deformation of the sensing elements.

Commonly force sensors are placed at the end-effector. The adopted solution however has some advantages, in particular it allows to:

- estimate forces and torques due to the internal dynamic of the links;
- measure external forces exerted on the whole arm;
- derive the torques at each joint.

The drawback is that, if not compensated for, dynamic forces due to the links are detected as external forces (something that does not happen when the F/T sensor is placed at the end-effector). To properly compensate forces acting on the whole arm, we need to know their point of application. Since the latter information is not available without tactile feedback in this work we assumed that all forces are applied at the end-effector (to remove this hypothesis we have recently installed a distributed tactile sensing system on the arms, see Del Prete et al., 2011; Schmitz et al., 2011). In addition we suppose also that the contribution of the internal dynamics of the manipulator are known and compensated for (see for example Murray et al., 1994). Under these hypotheses the actual external wrench and the measured F/T vector are related with pure kinematic relations, and it is possible to estimate the corresponding joint level torques.

With reference to Figure 5, the output of the F/T is a wrench ${}^{s}F_s \in \mathbb{R}^6$ represented in the sensor reference frame $< s >$. We can compute an equivalent wrench that, applied to the end-effector, produces the same effect. This clearly depends on the vector $p_s^e \in \mathbb{R}^3$ – the relative distance of the center of the sensor reference frame $O_s \in \mathbb{R}^3$ with respect to the position of the end-effector – and on the rotation matrix relating $< s >$ with $< e >$. Formally:

$$ {}^{b}F_e = T_e^b H_s^e \; {}^{s}F_s \tag{1} $$

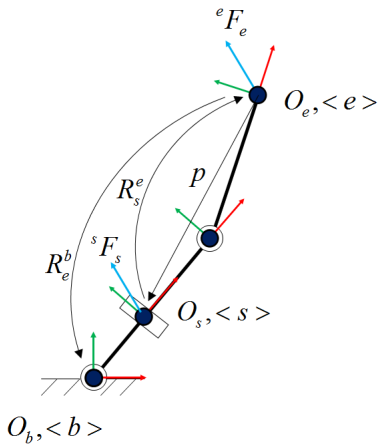

Fig. 5. Schematic representation of the reference frames and coordinate transformations involved in the computation of the joint torques from the measures of the F/T sensor. See text for details.

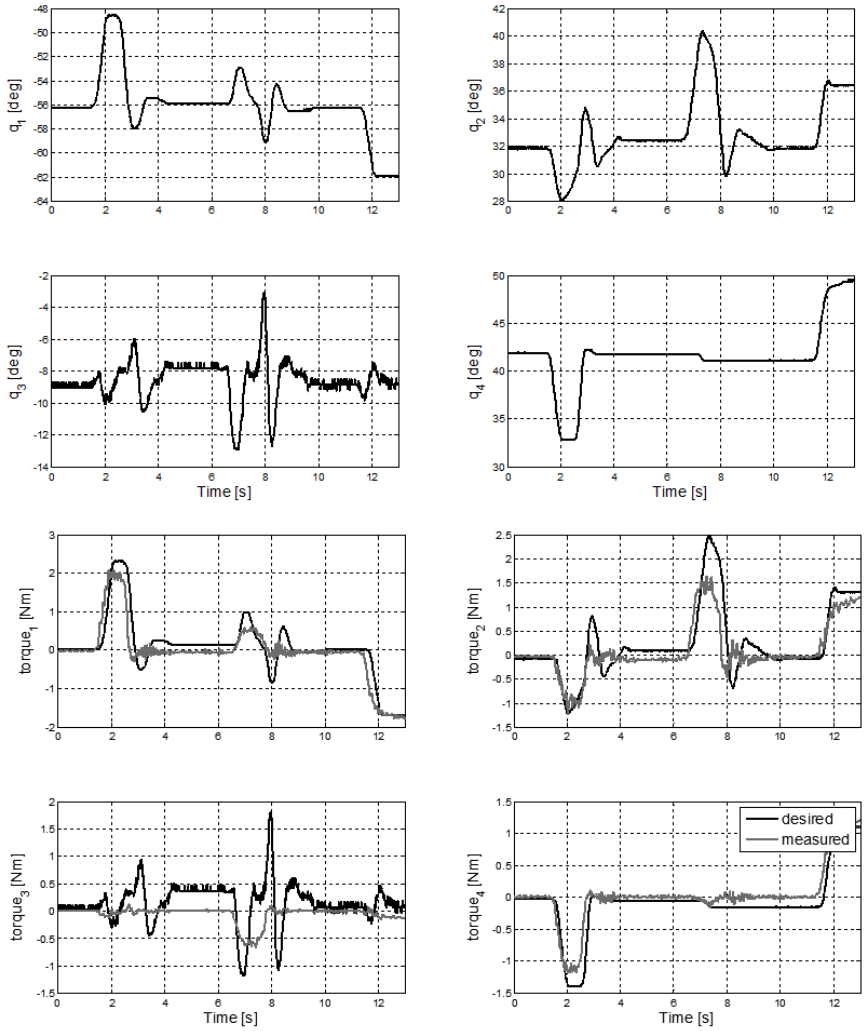

Fig. 6. Impedance controller during interaction with an external disturbance (only the four shoulder joints are considered). The external disturbance is applied to the arm at roughly $t_1 \approx 2\ s$, $t_2 \approx 4.5\ s$ and $t_3 \approx 12\ s$. Top plots: displacement of each joint during interaction (q_1, q_2, q_3 and q_4). The solid line shows the encoder value. During the whole experiment the reference value requested to each joint is maintained stationary. The compliant behavior of the arm is demonstrated by the fact that during the interaction with the disturbance the joints are allowed to move away from the reference position. Bottom plots: requested (black line) versus actual (gray line) torques at the joints ($torque_1$, $torque_2$, $torque_3$ and $torque_4$). The requested torques increase when the joints move away from the desired position and attempt to restore the desired position of the arm (forces are proportional to the displacement).

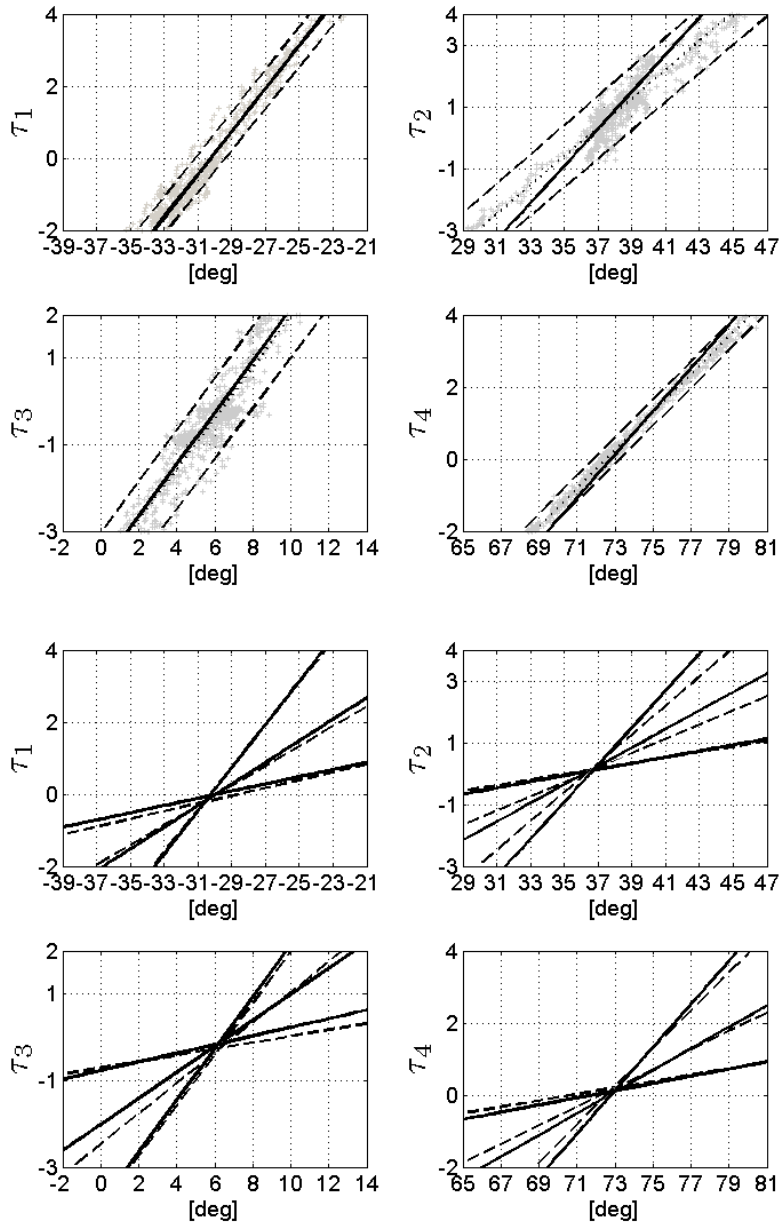

Fig. 7. Measuring the stiffness of each joint (only the four shoulder joints are considered). In these experiments the virtual springs have a constant equilibrium point at $q = (-30, 37, 6, 73)$ *degrees*. Top rows: all joints maintain a certain stiffness, the plots show the values of torques versus position. Crosses represent measured values, dashed lines 95% confidence interval for the measured stiffness, and least square linear fit. Solid line is the value of K used in the controller. Bottom plots: in this case we tested three different values of K. We plot the ideal response that would be obtained using the desired value of K (solid lines) versus least squares fit (dashed lines).

in which ${}^{b}F_e \in \mathbb{R}^6$ is the external force (represented in the reference frame $< b >$) and H_s^e and $T_e^b \in \mathbb{R}^{6\times 6}$ are defined as:

$$H_s^e = \begin{bmatrix} R_s^e & 0 \\ -S(p)R_s^e & R_s^e \end{bmatrix}, \tag{2}$$

$$T_e^b = \begin{bmatrix} R_e^b & 0 \\ 0 & R_e^b \end{bmatrix}. \tag{3}$$

Here $R_a^b \in \mathbb{R}^{3\times 3}$ represents the rotation matrix from $< a >$ to $< b >$, and $S(.) \in \mathbb{R}^{3\times 3}$ is the matrix operator of $p\times$.

From ${}^{b}F_e$ it is straightforward to compute the joint level torques:

$$\hat{\tau} = J^T(q)\ {}^{b}F_e, \tag{4}$$

where $J(q) \in \mathbb{R}^{6\times n}$ is the Jacobian of the n-joints manipulator, and q is the vector of joint positions. Given the value of $\hat{\tau}$ corresponding to the current reading ${}^{b}F_e$, the following control strategy:

$$u = PID(\hat{\tau} - \tau_d), \tag{5}$$

employs a proportional–integral–derivative controller (PID) to compute the motor command $u \in \mathbb{R}^n$ that achieves a desired value of joint torques τ_d (which, in turn, produce a corresponding net force exerted by the arm at the end effector).

A simple way to demonstrate force control is to implement an impedance controller[15]. At the joint level this is easily achieved by computing τ_d as in:

$$\tau_d = -K(q - q^\star), \tag{6}$$

being $K \in \mathbb{R}^n$ the vector of virtual joint stiffnesses. This controller simulates virtual springs attached to each joint i, with stiffness K and equilibrium point at $q_i^\star$.

When the value of the stiffness K is low the arm exhibits a "compliant" behavior, as demonstrated in the following experiment. The controller maintains $q^\star$, or equivalently, a certain position of the arm. We apply disturbances by means of variable forces applied at the end-effector. Forces produce a displacement of the end-effector; depending on the stiffness K the controller tries to oppose the external disturbances with a restoring force proportional to the displacement (Figure 6). This situation is similar to what happens when the arm interacts with the environment. In theory if K is large enough $q \to q^\star$ and we can use this controller to achieve any desired configuration of the arm. In practice, however, we would like to use small values of the stiffness K so to reduce the effects of unwanted collisions. To better validate the control system, in Figure 7 we report the plot of torque versus displacement with constant and variable values of the parameter K (respectively left and right).

[15] An impedance controller drives the arm by simulating virtual springs attached between the arm and a desired equilibrium point.

4.2 The Sensors on the Hand: Grasp Detection

The sensory system of the iCub allows us to determine when the fingers get in touch with an object. To this purpose we can exploit the encoders in the joints of the fingers and the tactile sensors on the fingertips and palm. Let us first focus on the former approach.

We have implemented a mechanism to perform contact detection using the springs mounted on the phalanges of the hand. Due to the elastic coupling between the phalanges (see Section 2 and, in particular, Figure 2) the fingers passively adapt when they encounter an obstacle (i.e. when they touch the surface of an object). The amount of adaptation can be indirectly estimated from the encoders on the joints of the fingers. In other words, the idea is to measure the discrepancy between the finger motion in presence of external obstacles (e.g. objects or the other fingers) and the one that would result in normal operation (in absence of obstacles/free movement). In a calibration phase we estimate the (linear) relationship between the joints of the fingers in absence of contact. In normal operation, we detect contact by comparing how much this model fits the current encoder readings.

Let us group together those joints of the fingers that are actuated by the same motors and that are coupled with elastic elements. For each of these groups we define generic n dimensional vectors $q \in \mathbb{R}^n$, each collecting the values of all the n joints that are actuated by the same motor (e.g. for the thumb, index and middle distal phalanges $n = 2$ each, while for the ring and small fingers $n = 6$, as explained in Figure 2). For each motor, the following parametric equation, r, models the mechanical coupling:

$$r : q = k_0 + k_1 \times \ \ s, s \in [s_{min}, s_{max}] \tag{7}$$

where s is the free parameter to be chosen in $[s_{min}, s_{max}]$. In a calibration phase we can fit this model to a set of joint positions recorded in absence of external

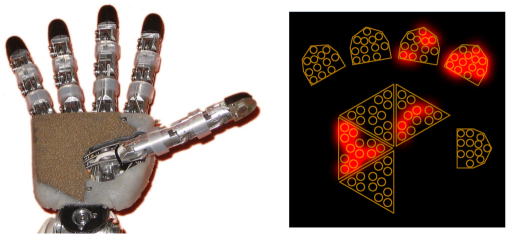

Fig. 8. The skin system on the hands consists of units of 12 capacitive sensors each: 5 units are installed in the fingertips and 4 cover the palm. Overall 108 sensors are available on each hand. Left: a picture of the hand. Right: schematic representation of how the units and sensors are distributed on the fingers and palm.

forces. Once we have determined the values of the parameters k_0 and k_1 we can determine if a given value of encoders q fit the mode in Equation 7. The idea is that the higher the effect of the external perturbation to the fingers the larger the error with which the model in Equation 7 predicts the value q. This algorithm was implemented in a software component that is available in the iCub software repository, along with a more detailed description of the algorithm[16].

The hand is equipped with a skin system made of interconnected units providing a total of 108 sensing elements distributed as in Figure 8. The details of the technology are reported elsewhere (Schmitz et al., 2008, 2011).

We now report a series of grasping experiments in which we show that the sensors of the hand (joint encoders and tactile system) can detect when the fingers touch an object. The robot was programmed to perform a series of grasping actions on different objects. The objects were placed on the same position and grasping was completely preprogrammed. We collected the information from the grasp detector (the output of the module, i.e. the error between the measured data and the model) and the output of one of the taxel of the fingertips (i.e. the taxel whose activation was stronger). As a reference we also collected data when the hand performed the same movement but in absence of an object. Figure 9 reports the data we collected averaged across 20 consecutive trials. The plots clearly show that both signals allow detecting when the finger gets in touch with the object. More details on this experiment are reported by Schmitz et al. (2010).

4.3 Reaching

We here describe a software component that controls the position and orientation of the hand[17]. Given a target position and orientation of the hand in the Cartesian space, a first stage of processing employs a non linear optimization technique to determine the arm joints configuration q_d that achieves the desired posture. The second stage of processing consists in a biologically inspired controller that computes the velocity $\dot{q}$ of the motors to produce a human-like quasi-straight trajectory of the end-effector. The details of this algorithm are reported by Pattacini et al. (2010) so we report here a brief description of its functionalities.

The solver module computes the value of the joint encoders $q^\star \in \mathbb{R}^n$ that achieves a given position $x_d \in \mathbb{R}^3$ and orientation $\alpha_d \in \mathbb{R}^3$ of the end-effector while, at the same time, satisfies a set of given constraints expressed as inequalities. Formally, this can be expressed as:

$$\begin{aligned} q^\star &= \arg\min_{q\in\mathbb{R}^n} \left(\|\alpha_d - K_\alpha(q)\|^2 + \lambda \cdot (q_{rest} - q)^T W (q_{rest} - q) \right) \\ \text{s.t.:} &\begin{cases} \|X_d - K_x(q)\|^2 < \epsilon \\ q_L < q < q_U \end{cases} \end{aligned}, \tag{8}$$

[16] *graspDetector*: see iCub Software Documentation, http://icub.org
[17] *iKinArmCtrlIF*: see iCub Software Documentation, http://icub.org

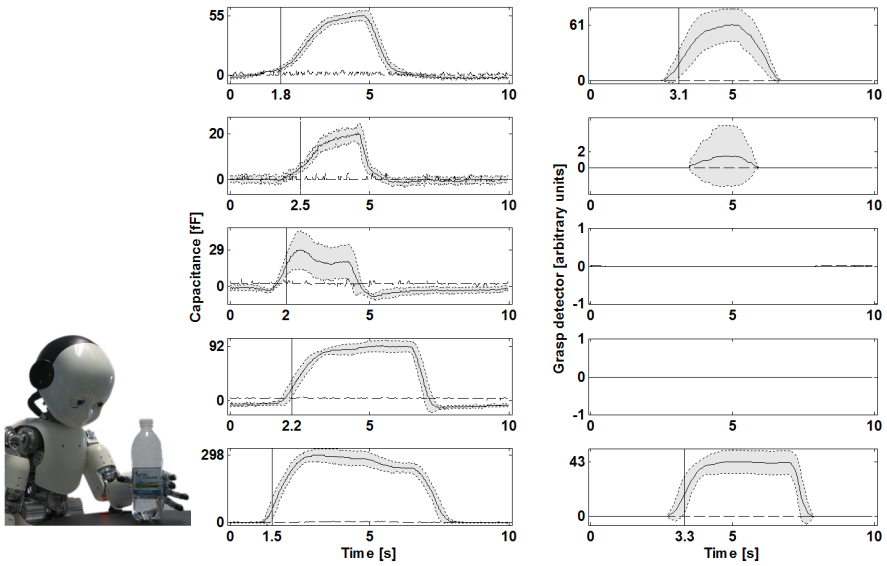

Fig. 9. A plot of the fingertips response (left column) and grasp detector response (right column) while grasping a plastic bottle. Rows corresponds to different fingers. From the top to the bottom: thumb, index, middle, ring and little finger. The dashed line is the response when the grasp action is performed without object. The solid line is the average response in 20 trials. The shaded region is the standard deviation in 20 trials. The horizontal axis reports also the detection time instant (vertical segment): this instant has been obtained on the basis of a threshold chosen on a 95% confidence interval. Data from other objects is reported in Schmitz et al. (2010).

where K_x and K_α are the forward kinematic functions that respectively compute position and orientation of the end-effector from the joint angles q; q_{rest} is a preferred joint configuration, W is a diagonal matrix of weighting factors, λ is a positive scalar (< 1) and ϵ a small number. The cost function in Equation 8 requires that the final orientation of the end-effector matches the desired value α_d and that the arm joints are as close as possible to a preferred "resting value" q_{rest}. The weights W determine which joints receive more importance during the minimization, whereas the scalar λ determines the overall importance given to this part of the cost. In addition the solution to the problem has to comply with a set of constraints. We here enforce that the position of the end-effector is arbitrarily close to the desired value X_d [18]. Other constraints can be imposed

[18] In doing so we ensure that this constraint receives higher priority in the minimization (Pattacini et al., 2010). This part of the task is fulfilled with a precision up to the value of the constant ϵ that is selected to be practically negligible (in our case $10^{-6}m$).

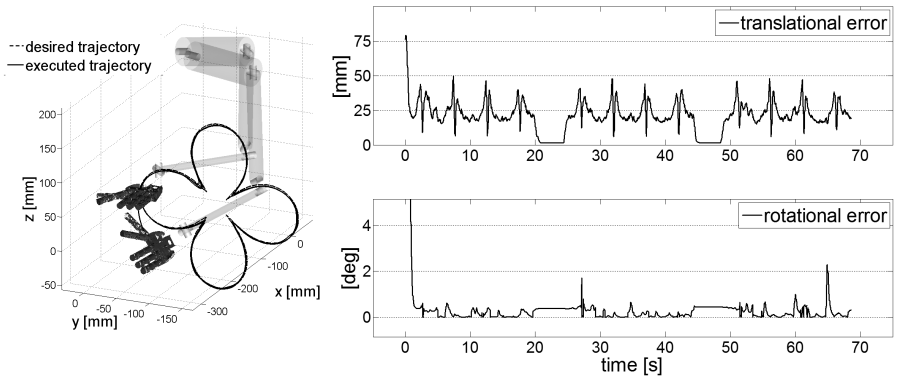

Fig. 10. Tracking a desired trajectory with a lemniscate shape in the operational space. Left: trajectory in Cartesian space. For better understanding the figure shows two configurations of the arm: the lower one depicts the starting pose, whilst the upper one shows the commanded hand orientation during the task. Right: tracking error during the same experiment.

as well, here for example we require the solution to lie between lower and upper bounds of physically admissible values. To solve the minimization problem we employ an Interior Point Optimization Technique, in particular we use the *Ipopt* libray, a public domain software package designed for large-scale non linear optimization (Wächter and Biegler, 2006). As demonstrated in Pattacini et al. (2010) this technique has several advantages, such as, speed, automatic handling of the singularities and possibility to incorporate complex constraints as inequalities in the problem.

The controller module is responsible for computing a series of joint space velocities $\dot{q}$ that drive the arm from the current configuration q to the desired final state $q^\star$ computed by the solver in the previous step. The approach we follow in this case is similar to the Multi-Referential Dynamical Systems approach (Hersch and Billard, 2008). Two dynamical systems operate in joint and task space with the same target position and generate desired commands for the arm. The coherence constraint between the two tasks is enforced with the Lagrangian multipliers method. This can be used to modulate the relative influence of each controller. The joint level controller produces straight trajectories in joint space, while the task level controller produces straight trajectories in the task space. Both controllers have their own advantages, the former allows to avoid joint limits, while the latter make sure the arm follows rectilinear bell-shaped trajectories in the task space. Instead of the second order dynamical systems proposed by Hersch and Billard (2008), we use a third order system whose coefficients are tuned to better approximate a minimum jerk profile (Flash and

Hogan, 1985) . This allows production of smoother trajectories of the arm, both in the joint and task space (Pattacini et al., 2010).

To simplify the use of the *Reaching Controller* YARP defines an interface that specifies methods for task space control of a robotic structure. The purpose is twofold: 1) it achieves better modularity and 2) hide the implementation details of the controller behind a set of immutable interfaces. Examples of these methods are: *go_to_pose(), get_pose(), set_trajectory_time()*[19]. The functionalities of this module are also available through the *iKin* library; this is a general purpose kinematics library that can be configured to represent the forward kinematics of any serial link.

Figure 10 shows the Cartesian position of the end-effector while tracking a desired pose in the operational space; in this particular example 10 degrees of freedom are controlled (7 for the arm along with the pitch, the roll and the yaw joints of the torso): one cycle of the lemniscate-shaped desired trajectory in front of the robot frontal plane was executed in 20 seconds, whereas the time control gain T for point-to-point movements was set to 0.5 seconds. Our experiments show that the controller can easily run in real-time and has good response in terms of accuracy and smoothness.

5 An Integrated Behavior

We present an example of a grasping behavior that was implemented on the robot using modules in the repository as building blocks. The earliest stage of the visual processing is the attention system (see for example Ruesch et al., 2008), which consists in detecting regions in the (visual) space towards which directing gaze. A commonly adopted solution is to employ a set of filters (each tuned to features like colors, orientations and motion) and combine their output to obtain a saliency map. This saliency map is then searched for local maxima which correspond to "interesting" regions in the visual scene. The gaze of the robot is finally directed towards these saliency regions using a certain criteria (in this case using a "winner-take-all" approach, but other strategies like random walk could be thought). Each feature can be given more importance: in this case it was decided to give more priority to motion so that moving objects are more likely to attract the attention of the robot. The gaze control module controls the motor of the head to bring salient regions at the center of the cameras. A low level segmentation algorithm groups together areas in the images that have uniform color and extract the center of the area that is closer to the center of the image. The process described above exploits visual information from one camera, and extracts only the location of the target in image coordinates. Since the extraction of 3D information is difficult and easily imprecise we decided to take a pragmatic approach and compute the missing information from the assumption that all relevant objects lay on a table whose height is determined by touch. With this

[19] A complete list of available methods is available on the documentation page of the *ICartesianControl* interface: http://icub.org, YARP documentation.

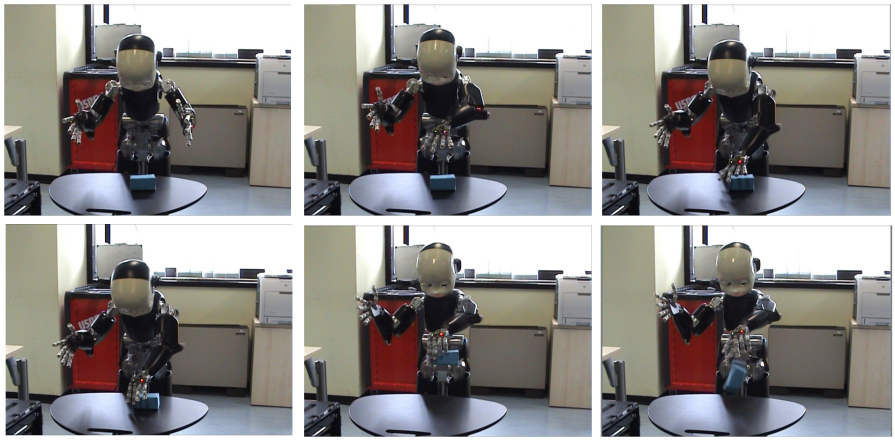

Fig. 11. A grasping sequence. From top-left to bottom-right, an object is placed on the table, the robot moves the hand above the object and closes the hand. When a successful grasp is detected the robot lifts the object and eventually drops it.

assumption the visual processing module computes the Cartesian position of the target with respect to a known reference frame.

The reaching subsystem receives the Cartesian position of the target object and computes the trajectories of the joints to achieve the desired posture. Once the hand is above the object the finger closes, with a predefined movement, until the robot detects the contact with the object. If grasping is successful the robot lifts the object, otherwise it opens the hand and brings the arm back to the initial position. The whole sequence is exemplified in Figure 11.

This behavior, although already sophisticated, makes several simplifying assumptions about the locations of the objects and, to a certain extent their shape. In the context of this paper, this experiment shows an example of how basic modules can be integrated to perform a meaningful behavior that involves the interaction between the sensory system and the controllers of different body parts. Finally, this behavior could be, itself, a building block for an even more complicated behavior involving the interaction with the environment (for example learning of affordances or human robot cooperation to mention a few).

6 Conclusions

The design of the robot started with difficult constraints. The targeted research area required the robot to have a certain degree of complexity to allow sophisticated interaction with the environment. Fitting 53 degrees of freedom and all the sensors in a small, integrated platform was one of the most difficult design

challenges. The iCub had to be built in multiple copies, to be used by people that did not participate in its development and likely to have a mixed background ranging from engineering and computer science to biology and psychology. For these reasons the robot had to go beyond the level of a simple prototype but rather be a mature, documented and robust platform. The software had to minimize the learning curve, and be usable by non-expert users. Finally, one of the goals of the project was to create a community of people working on the same robots, sharing results and algorithms.

In the past years the community of iCub users has been rapidly growing around the 20 robots that have been distributed, the mailing list, the summer schools and the use of the software simulator. As a whole the community is contributing to the iCub at various levels: from basic functionalities like routines for calibration, control of attention or grasping to more sophisticated ones like learning of object affordances and learning by demonstration (a representative list of these functionalities is reported in Table 1).

In this Chapter we have provided an overview of the iCub platform. We have covered aspects of the software and hardware architecture that make the platform suitable for research in the fields of developmental robotics and cognitive systems in general. We have also provided a description of the available functionalities, with particular focus on those that are more relevant for controlling the manual interaction with the environment, namely force control and touch-based control of the arms and hands (references to other functionalities implemented on the iCub are in Table 1). These features support the implementation of exploratory behaviors in that they allow safe and prolonged interaction with the environment. The iCub sensory system also provides a wealth of information opportunities for learning. The software architecture is intrinsically modular it allows experimenting with learning architectures in which complex abilities are formed on the basis of simpler ones. Overall we believe these functionalities make the iCub particularly interesting to the community of researchers studying intrinsically motivated learning. As we have demonstrated the iCub is an unique integrated platform that embeds most of what is needed to tackle the problems that are challenging the research community in this field.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreements N° 231500 (ROBOSKIN), N° 214668 (ITALK) and N° 215805 (CHRIS).

Overview of high-level functionalities on the iCub		
Functionality	Description	References
Attention System	Control of attention using visual and auditory cues	Ruesch et al. (2008)
Log-polar Attention System	Control the attention using visual cues in log-polar space	The iCub "contrib" software repository
Reaching	Control the arm to reach for a point in space with the hand	Pattacini et al. (2010)
Force Control	Control the amount of force exchanged between the arm, legs and the environment	Fumagalli et al. (2010)
Skin spatial calibration	Automatic calibration of the tactile system, compute the spatial location of each taxel	Del Prete et al. (2011)
Head calibration	Perform automatic calibration of the head using vision and inertial information	Santos et al. (2010)
Crawling	Locomotion on the legs and arms	Dégallier et al. (2011)
Cognitive Architecture	A cognitive architecture for the iCub robot implementing gaze control	Vernon et al. (2011)
Imitation learning and grasping	The robot learns a grasp model from a first demonstration of a hand posture which is then physically corrected by a human teacher pressing on the fingertips	Sauser et al. (2011)
Kinesthetic teaching	An action acquisition model based on multiple time-scales recurrent neural network and self-organizing maps; the robot learns multiple behaviors through demonstration	Peniak et al. (2011a)
Imitation, Learning Object Affordances	The robot learns a representation of object affordances and uses it to imitation actions	Montesano et al. (2008)
Objects and Actions Learning	Learning under human supervision, includes visual object recognition and actions	The iCub "main" software repository
Object Learning	Replication of the "modi experiment" from developmental psychology literature	Peniak et al. (2011b)
Reaching with force fields and obstacle avoidance	Plan a trajectory in space while avoiding obstacles; obstacles are represented as force fields	The iCub "main" software repository
Body schema: hand detection and localization	The robot learns autonomously to visually identify its own hand and arm	Ciliberto et al. (2011); Saegusa et al. (2011)
Online Learning Machine	An incremental learning algorithm for regression	Gijsberts and Metta (2011)

Table 1. A list of the functionalities that have been implemented on the iCub. The table reports the name of the functionality and a description. When available references to the scientific papers that present the work are reported in the right column.

Bibliography

Barto, A., Singh, S., and Chentanez, N. (2004). Intrinsically motivated learning of hierarchical collections of skills. In *International Conference on Developmental Learning*.

Bushnell, E., W. and Boudreau, J., P. (1993). Motor development and the mind: The potential role of motor abilities as a determinant of aspects of perceptual development. *Child Development*, 64(4):1005–1021.

Ciliberto, C., Smeraldi, F., Natale, L., and Metta, G. (2011). Online multiple instance learning applied to hand detection in a humanoid robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

De Santis, A., Siciliano, B., De Luca, A., and Bicchi, A. (2008). An atlas of physical Human–Robot Interaction. *Mechanism and Machine Theory*, 43(3):253–270.

Dégallier, S., Righetti, L., Gay, S., and Ijspeert, A. (2011). Towards simple control for complex, autonomous robotic applications: Combining discrete and rhythmic motor primitives. *Autonomous Robots*, 31(2):155–181.

Del Prete, A., Denei, S., Natale, L., F., M., Nori, F., Cannata, G., and Metta, G. (2011). Skin spatial calibration using force/torque measurements. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Fitzpatrick, P., Metta, G., and Natale, L. (2008). Towards long-lived robot genes. *Robotics and Autonomous Systems*, 56(1):29–45.

Flash, T. and Hogan, N. (1985). The coordination of arm movements: an experimentally confirmed mathematical model. *The Journal of Neuroscience*, 5(3):1688–1703.

Fumagalli, M., Nori, F., Randazzo, M., Natale, L., Giorgio, M., and Giulio, S. (2010). Exploiting proximal F/T measurements for the iCub torque control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Gijsberts, A. and Metta, G. (2011). Incremental learning of robot dynamics using random features. In *IEEE International Conference on Robotics and Automation*.

Haddadin, S., Albu-Schaffer, A., and Hirzinger, G. (2009). Requirements for safe robots: Measurements, analysis and new insights. *The International Journal of Robotics Research*, 28(11-12):1507–1527.

Hersch, M. and Billard, A. (2008). Reaching with multi-referential dynamical systems. *Autonomous Robots*, 25(1-2):71–83.

Hörnstein, J., Lopes, M., and Santos-Victor, J. (2006). Sound localization for humanoid robots — building audio-motor maps based on the HRTF. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1170–1176.

Kaplan, F. and Oudeyer, P.-Y. (2008). Intrinsically motivated machines. In Lungarella, M., Iida, F., Bongard, J., and Pfeifer, R., editors, *50 Years of AI*, pages 303–314. Springer.

Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G. (2003). Developmental robotics: a survey. *Connection Science*, 15(4):151–190.

Metta, G., Sandini, G., Vernon, D., Natale, L., and Nori, F. (2008). The iCub humanoid robot: an open platform for research in embodied cognition. In *PerMIS: Performance Metrics for Intelligent Systems Workshop*.

Montesano, L., Lopes, M., Bernardino, A., and Santos-Victor, J. (2008). Learning object affordances: From sensory–motor coordination to imitation. *IEEE Transactions on Robotics: Special Issue on Biorobotics*, 24(1):15–26.

Murray, R. M., Sastry, S. S., and Zexiang, L. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA.

Natale, L. (2009). A study on YARP performance. Technical report, Department of Robotics, Brain and Cognitive Sciences, Istituto Italiano di Tecnologia.

Needham, A., Barret, T., and Peterman, K. (2002). A pick-me-up for infants exploratory skills: Early simulated experiences reaching for objects using sticky mittens enhances young infants object exploration skills. *Infant Behavior and Development*, 25(3):279–295.

Oudeyer, P.-Y. and Kaplan, F. (2007). What is intrinsic motivation? a typology of computational approaches. *Frontiers in Neurorobotics*, 1(6).

Parmiggiani, A., Randazzo, M., Natale, L., Metta, G., and Sandini, G. (2009). Joint torque sensing for the upper-body of the iCub humanoid robot. In *IEEE-RAS International Conference on Humanoid Robots*.

Pattacini, U., Nori, F., Natale, L., Metta, G., and Sandini, G. (2010). An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Peniak, M., Marocco, D., Tani, J., Yamashita, Y., Fischer, K., and Cangelosi, A. (2011a). Multiple time scales recurrent neural network for complex action acquisition. In *International Joint Conference on Development and Learning (ICDL) and Epigenetic Robotics (ICDL-EPIROB)*.

Peniak, M., Morse, A., Larcombe, C., Ramirez-Contla, S., and A., C. (2011b). Aquila: An open-source gpu-accelerated toolkit for cognitive and neuro-robotics research. In *International Joint Conference on Neural Networks (IJCNN)*.

Ruesch, J., Lopes, M., Bernardino, A., Hörnstein, J., Santos-Victor, J., and Pfeifer, R. (2008). Multimodal saliency-based bottom-up attention a framework for the humanoid robot iCub. In *IEEE - International Conference on Robotics and Automation*.

Saegusa, R., Natale, L., Metta, G., and Sandini, G. (2011). Cognitive robotics - active perception of the self and others. In *4th International Conference on Human System Interaction*.

Santos, J., Bernardino, A., and Santos-Victor, J. (2010). Sensor-based self-calibration of the iCub's head. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Sauser, E., Argall, B. D., Metta, G., and Billard, A. (2011). Iterative Learning of Grasp Adaptation through Human Corrections. *Accepted for publiction in Robotics and Autonomous Systems*.

Schiavi, R., Flacco, F., and Bicchi, A. (2009). Integration of active and passive compliance control for safe human-robot coexistence. In *IEEE International Conference on Robotics and Automation*.

Schmitz, A., Maggiali, M., Randazzo, M., Natale, L., and Metta, G. (2008). A prototype fingertip with high spatial resolution pressure sensing for the robot iCub. In *IEEE-RAS International Conference on Humanoid Robots*.

Schmitz, A., Maiolino, P., Maggiali, M., Natale, L., Cannata, G., and Metta, G. (2011). Methods and technologies for the implementation of large scale robot tactile sensors. *IEEE Transactions On Robotics: Special Issue on Robot Sense of Touch*, 23(3):389–400.

Schmitz, A., Pattacini, U., Nori, F., Natale, L., and Metta, G. (2010). Design, realization and sensorization of a dextrous hand: the iCub design choices. In *IEEE-RAS International Conference on Humanoid Robots*.

Sciavicco, L. and Siciliano, B. (2005). *Modelling and Control of Robot Manipulators*. Advanced textbooks in control and signal processing. Springer, second edition.

Tikhanoff, V., Fitzpatrick, P., Metta, G., Natale, L., Nori, F., and Cangelosi, A. (2008). An open source simulator for cognitive robotics research: The prototype of the iCub humanoid robot simulator. In *Workshop on Performance Metrics for Intelligent Systems*.

Tsagarakis, N., Becchi, F., Righetti, L., Ijspeert, A., and Caldwell, D. (2007). Lower body realization of the baby humanoid - iCub. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Vernon, D., von Hofsten, C., and Fadiga, L. (2011). A roadmap for cognitive development in humanoid robots. volume 11 of *Cognitive Systems Monographs (COSMOS)*. Springer.

von Hofsten, C. (2004). An action perspective on motor development. *Trends in cognitive sciences*, 8(6):266–272.

Wächter, A. and Biegler, L., T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., and Thelen, E. (2000). Autonomous mental development by robots and animals. *Science*, 291(5504):599–600.

Zinn, M., Roth, B., Khatib, O., and Salisbury, J. K. (2004). A new actuation approach for human friendly robot design. *The International Journal of Robotics Research*, 23(4-5):379–398.

Zlatev, J. and Balkenius, C. (2001). Why "epigenetic robotics"? In Balkenius, C., Zlatev, J., Kozima, H., Dautenhahn, K., and Breazeal, C., editors, *International Workshop of Epigenetic Robotics*, volume 85, pages 1–4. Lund University Cognitive Studies.