International Journal of Humanoid Robotics
Vol. 9, No. 3 (2012) 1250017 (26 pages)
© World Scientific Publishing Company
DOI: 10.1142/S021984361250017X

AUTONOMOUS ONLINE LEARNING OF REACHING BEHAVIOR IN A HUMANOID ROBOT

LORENZO JAMONE*, LORENZO NATALE† and FRANCESCO NORI‡

Department of Robotics Brain and Cognitive Sciences,
Italian Institute of Technology,
Via Morego 30, Genoa, 16163, Italy
**lorenzo.jamone@iit.it*
†lorenzo.natale@iit.it
‡francesco.nori@iit.it

GIORGIO METTA§ and GIULIO SANDINI¶

Department of Robotics Brain and Cognitive Sciences,
Italian Institute of Technology Via Morego 30, Genoa, 16163, Italy
DIST, University of Genoa Viale Causa 13, Genoa, 16145, Italy
§giorgio.metta@iit.it
¶giulio.sandini@iit.it

Received 2 May 2011
Accepted 24 January 2012
Published 17 July 2012

In this paper we describe an autonomous strategy which enables a humanoid robot to learn how to reach for a visually identified object in the 3D space. The robot is a 22-DOF upper-body humanoid with moving eyes, neck, arm and hand. The robot is bootstrapped with limited *a-priori* knowledge, sufficient to start the interaction with the environment; this interaction allows the robot to learn different sensorimotor mappings, required for reaching. The arm-head forward kinematic model and a visuo-motor inverse model are learned from sensory experience. Learning is performed purely online (without any separation between training and execution) through a goal-directed exploration of the environment. During the learning the robot is also able to build an internal representation of its reachable space.

Keywords: Reaching; autonomous online learning; humanoids.

1. Introduction

The goal of the reaching action is to bring the robot end-effector to a specific position in space. This position can be obtained from a given model of the environment, or it can be retrieved from vision; the latter solution requires a kinematic model of the eyes-head system and calibration of the cameras. Moreover, a kinematic model of the arm is needed to plan the appropriate motion. When these models are accurate enough and do not change over time, this approach allows planning and executing reaching actions effectively. On the other hand, in common situations the estimation of the kinematic parameters is difficult and error-prone (consider for example a robot actuated with elastic materials, such as steel tendons). Therefore, it is desirable that the robot is able to: (i) perform this estimation autonomously and (ii) maintain proper calibration over time despite changes in the kinematic structure or sensory system. The problem has been addressed extensively in the literature[1–18]; however none of these works provides a complete and general solution which tackles all the aspects of the problem.

Our contribution in this paper is to describe a learning behavior that allows a humanoid robot to learn autonomously and online the models required for controlling reaching. In our approach the exploration of the state space, which is necessary in any learning system, is *goal-directed*, i.e. it happens during the execution of reaching trials and does not require an explicit distinction between learning and execution. No initial *motor babbling* is needed: the robot learns autonomously (i.e. from the interaction with the environment) and continuously.

We show experimentally that the robot is able to maintain updated the learned kinematics models and adapt them to compensate for perturbations in the arm kinematics or in the visual system. We also present preliminary results showing how the robot can learn a representation of its own reachable space during the execution of reaching trials. To the best of our knowledge, previous works do not provide similar experimental results on real robots.

In the following, we discuss the lessons we took from human development in designing our system (Sec. 2) and we review the state of the art (Sec. 3). Then, in Sec. 4 we present James, the platform used in our experiments, while in Sec. 5 we illustrate our approach in detail. Finally, in Sec. 6 we show the experimental results and in Sec. 7 we report our conclusions.

2. Lessons from Biology

In general, biological systems do not learn “from scratch”; on the contrary they are endowed at birth with a certain amount of knowledge and abilities.[19,20] Newborn human infants also possess a repertoire of coordinated movements that are probably exploited to start the interaction with the environment.[21] For example, during pre-reaching the presence of the Asymmetric Tonic Neck Reflex (ATNR) might have the crucial role to (i) allow babies to see their hands and (ii) help investigating the relationship between vision and proprioception.[22,23]

A primitive form of eye-hand coordination is present in newborns from the first days of life: experimental results report extension movements of the arm toward fixated objects (goal-directed movements).[24] Indeed, it has been hypothesized that humans employ a gaze-centered frame of reference for reaching control,[25,26] even in the case of whole-body reaching.[27]

Until four months, the reaching motion seems to be just “ballistic”, as trajectory correction is absent.[28,29] A noisy command generation (related to immature muscle control) is observed[30] which may improve the exploration required for motor learning. Visually guided reaching is used to correct the movement from five months[22,31] with performance that improves during developement.[32] At about six/eight months infants reach consistently for visually identified objects, rarely missing the target.[33]

Following recent neuroscience results,[34] it seems that when humans use a tool for reaching the internal model of their body (i.e. the *body schema*) is updated as if the hand were moved to the tip of the tool, allowing successful reaching. Moreover, humans can adapt to spatial perturbations when moving their hands under visual control. This adaptation mechanism affects internal kinematic models of the body by exploiting the perceived errors in the movement[35]; for example, they can learn how to control hand movements in a rotated reference frame.[36]

In addition, neurophysiological evidence shows that a representation of what is reachable or not is somehow encoded in the human brain and is based on motor information.[37,38]

Summarizing, these observations provide interesting suggestions for the realization of reaching in robots:

- a limited *a priori* knowledge helps to initiate learning;
- noisy command generation may help the initial exploration;
- gaze anticipates reaching: fixation of the target object is achieved before reaching;
- the reaching action combines two phases: an “open-loop” (ballistic) movement and a “closed-loop” (corrective) movement.

On the other hand, the abilities shown by humans suggest interesting ways to benchmark the adaptive abilities of the robots we design by measuring their abilities to:

- adapt to changes in the kinematic structure;
- adapt to perturbations in the visual system;
- build a representation of the reachable space.

3. State of the Art

Early works in robotics have proposed to perform gaze control before reaching[3–6]: after the fixation of the target is achieved, the target position is encoded with the current head motor configuration, which is then used as a reference for reaching. Then open-loop reaching is realized by exploiting a learned model of the head-arm forward kinematics. In Ref. 4, learning of this model is performed during a training phase separated from the subsequent execution phase, while in Refs. 3 and 5 the model is updated during action execution: in Ref. 5 this is achieved by redirecting the robot gaze to the end-effector after unsuccessful reaching movements. While Refs. 3 and 4 do not provide quantitative results, in Ref. 5 experiments with a robotic setup are reported; the learned model maps two eye/head control parameters to two arm control parameters (a 2D nonredundant mapping). In a later version of the work[6] the authors add control of vergence (i.e. control of depth). Recently this approach has been successfully extended to redundant manipulators,[11] although in the case of a 2D visual space. A different strategy is investigated in, Ref. 10, in which the target position is encoded with Cartesian coordinates computed from vision (thus requiring calibration of the cameras): the forward model of the arm is learned through motor babbling and then derived to obtain the Jacobian, which is used to control reaching. This paper reports an experiment with a real robot in which, however, the head is maintained stationary. In a simulated experiments this limitation is removed by adding the measurement from the gyroscope, at the cost of a larger learning space. The advantage of realizing gaze control before reaching is that calibration of the cameras is not necessary and no additional transformations are needed to account for neck movements: the head-arm forward kinematics can be learned as a single motor-motor mapping. We favor this approach in our work.

Traditionally reaching has been achieved in closed-loop, by directly employing in the control the measure of the distance between the hand and the target in the visual space (visual servoing[39]). With respect to this solution, however, open-loop control has the following advantages: it can be executed even if the hand is initially outside the visual field and it does not suffer from velocity limits imposed by visual feedback delays. However, it always performs with some residual error, as it crucially relies on the accuracy of the kinematic model; visual feedback is necessary to reduce the hand positioning error to zero. The work in Ref. 8 proposes to correct positioning errors after the open-loop reaching by using visual feedback (as originally suggested in simulation in Ref. 1); however, only the open-loop controller is learned from sensory data. Following this idea we proposed a system in which both the open-loop and the closed-loop controller are learned from experience and used for control[13]: learning is performed offline using sensory data collected during a training phase in which the robot executes random movements. Redundancy in both the head and the arm is considered.

Planning the reaching movement either in closed-loop or in open-loop requires an inverse model. We group the approaches to this problem in two families: approaches in which the forward model is learned first and then inverted [2,12,17,40] and approaches in which an inverse model is directly learned from data.[7,9,16] In general, in the case of redundant robots, the forward model provides a richer description of the system (i.e. it includes the redundancy) and gives more opportunities for control: several methods can be used (e.g. extended Jacobian,[41] augmented Jacobian,[42] local minimization through null space projection,[43] numerical optimization[44]) to exploit the redundancy in order to solve a secondary task (i.e. respect additional constraints) while performing the main one. Possible choices for such a secondary task can be joint limits avoidance, obstacle avoidance, energy optimization or many others. Recent works have applied these techniques in the case of learned models.[17,40] Nevertheless, the model inversion can generate instability problems if the Jacobian matrix is ill-conditioned: large joints velocities are generated when the system is close to Jacobian singularities. On the contrary, the problem is avoided when the inverse model is learned directly from the gathered velocity data (which is physically feasible by definition, as they are generated by the robot motion), as proposed in Refs. 7 and 16. In this case the drawback is that only a single solution of the inverse problem is learned. The particular inverse solution that is learned depends on the nature of the training data collected by the robot. If data is generated according to a special redundancy resolution scheme (as in Ref. 7) then the mapping from task space velocities to joint space velocities becomes unique, and the learned inverse solution shows the same resolution of redundancy. On the contrary, if they do not present any special structure the mapping is not unique: there could be multiple joint space velocities mapping the same task space velocity. However, a specific choice of the input representation and the use of a spatially localized learning network allow to learn a valid inverse mapping also from unstructured training data, as explained in Ref. 45 and recalled in Ref. 7 on page 13. In this case the inverse solution approximated by the learning network is a local average of the solutions experienced by the robot (i.e. the solutions described by the training data), which is still proved to be a valid solution.

Learning kinematic models for reaching is a particular case of learning the body schema, a problem which has been addressed in recent works.[14,15,18] In Refs. 14 and 15 algorithms for learning the kinematics of robotic structure using visual information are proposed. In Ref. 18 the authors introduce a method based on active learning. These works share the same limitation: in all cases learning is performed following or during an exploration phase for which the robot has been specifically programmed. Indeed, the separation between exploration and exploitation is somehow present in most works we introduced, namely Refs. 1, 4, 7, 9–18. This approach has two problems. First, at some point the robot (i.e. learning system) should decide to stop learning and activate the controller that uses the learned model. Besides the problem of individuating the time at which the switch has to be performed (how can the robot say that it has learned enough?), such a learned model would be no longer useful if the system changes; even if the learned model is then refined online (as it is the case for many recent works), the initial exploration could take too much time in high-dimensional space. Moreover, different controllers are needed to drive the robot during the two phases as the goal changes (i.e. during the exploration phase the goal is just to gather data for learning). This is neither intuitive nor practical. Indeed, it would be desirable that the exploration of the learning space was goal-directed; a clear advantage is that this guarantees that the exploration is driven toward the areas of the state space in which reaching is performed. This helps reducing the size of such space since it focus exploration on those parts of the state that are more important for the task. This could allow a faster convergence of the estimation error in those areas, as we show in Ref. 46 for a similar problem; if the robot performs actions in other regions of the motor space, the learned model is updated online.

On the other hand, the other works we introduced[2,3,5,6,8] apply online learning only to some parts of the whole system (e.g. they learn only the open-loop control) and they generally present results on simpler systems (e.g. not redundant).

A central purpose of our work is to merge exploration and exploitation in a unique behavior, achieving pure online learning; this is also the main improvement with respect to our previous work.[13] Moreover, we provide novel experimental validation to demonstrate: (i) adaptation to kinematic modifications, (ii) adaptation to perturbations in the visual system, (iii) incremental building of reachable space representation. Concerning the latter point, recent studies have tested numerical methods in order to build a representation of the robot reachable space [47,48]: the main difference of our approach is that we encode positions in space with head motor coordinates (i.e. the same encoding we use for reaching control) and we build the reachable space during an active and goal-directed exploration of the environment. Moreover, we create a compact representation which can be easily updated and used in real-time.

4. The Humanoid Robot

As in our previous work,[13] the experiments described in this paper have been carried out on the robot James.[49] James is an upper body humanoid with 22 DOFs (see Fig. 1). It is actuated using rotary DC motors, whose angular position is measured by magnetic incremental encoders. Torque is transmitted to the joints by belts and stainless-steel tendons. Tendon driven systems are getting popular in robotics

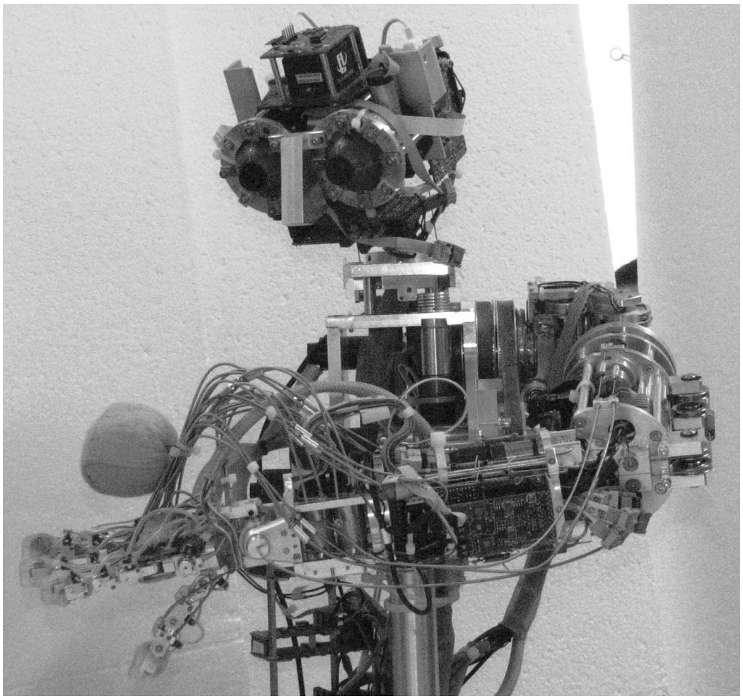


Fig. 1. The humanoid robot James.

because they allow sophisticated routing of the actuation to reduce the weight and inertia of the mechanical structure. Nevertheless, tendons also introduce a nonlinear elasticity which is difficult to model; this further justifies the use of machine learning in our work.

The head structure has a total of seven DOFs. Four motors are used to independently actuate the pan and tilt movements of the left and right eyes. Even though the eyes can be moved independently, our strategy was to couple their movements so to achieve a more human-like motion: we used common tilt, vergence and version. Two CCD digital cameras (PointGray Dragonfly remote head) are mounted into the eyeballs, providing 320×240 pixels images at 15 fps. The neck has three degrees of freedom, which allow rotation (i.e. yaw), elevation/depression (i.e. pitch) and adduction/abduction (i.e. roll) of the head (see Refs. 40, 50 and 51 for a detailed description of the neck structure and control). The arm has seven DOFs: three of them are located in the shoulder, one in the elbow and three in the wrist. The hand has five fingers and is under-actuated with a total of eight DOFs. Tactile sensors have been realized and mounted on the internal side of the fingers (see Ref. 49 for details).

For the experiments reported in this paper, a green ball (2 cm radius) has been attached to the robot wrist as a marker for the hand, in order to simplify its visual recognition: the ball center can be localized with pixel precision using standard image processing, and constitutes the robot end-effector for the experiments. From here on, when we refer to "the hand" we mean the center of the ball. We actuate four DOF of the arm (adduction/abduction, elevation/depression and rotation of the shoulder, flexion/extension of the elbow) and three DOF of the head (vergence of the eyes, yaw and pitch of the head), so the motor variables relevant to understand the remaining of the paper are:

- $\mathbf{q}_{\text{arm}} \in \mathbb{R}^4$
- $\mathbf{q}_{\text{head}} \in \mathbb{R}^3$

A cluster of standard PCs (Intel Core2 Duo @2.00 GHz) and a Blade system (Primergy RX200 server with six additional blades, Intel Xeon @2.00 GHz) are interconnected through a 1 GB ethernet and constitute the core of the brain of James. 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. All this software has been written using YARP.[52]

5. Our Approach

The key features that characterize our approach are listed hereinafter:

- fixation of the target object comes first, and the head motor configuration obtained after fixation is used as a frame of reference for controlling reaching;

- the goal of the reaching movement is to bring the hand to the fixation point (i.e. the center of both cameras);
- the reaching controller combines open-loop and closed-loop control;
- the models required for controlling reaching are learned purely online through a goal-directed exploration of the environment;
- the system is bootstrapped with limited *a priori* knowledge, which is then gradually replaced by the one acquired from sensory experience;
- noise is added to the initial motor commands in order to improve the exploration and facilitate motor learning.

The learning behavior described in this paper is independent of the particular learning algorithm used for regression. The only requirements are that the algorithm is fast (as the learned models are used for control purposes), online and spatially localized (the latter requirement is discussed in Sec. 3). In this paper, the system was validated using RFWR (Receptive Field Weighted Regression[53]). This algorithm has been used successfully in similar scenarios for robot learning[7,17] in its recently modified version (LWPR, Locally Weighted Projection Regression[54]). However, the choice of this particular algorithm was dictated by practical considerations. More recent algorithms can be used as well, like for instance Local Gaussian Process Regression (LGPR,[55]).

In the experiments reported in Sec. 6 we show how the learned models that are used for reaching control are updated continuosly during goal-directed reaching movements; in particular, the robot is able to adapt to sudden modifications of its own kinematics and perturbations of the visual perception. Then, preliminary results are provided about learning a representation of the reachable space. To the best of our knowledge, such experimental evaluations are not provided in previous works. The following subsections describe the different components of the system in detail, namely:

- the gaze controller;
- the open-loop controller;
- the closed-loop controller;
- the goal-directed exploration;
- the representation of the reachable space.

5.1. *Gaze controller*

The head is controlled with a simple proportional controller in order to fixate either the hand or the object. To the sake of clarity we will refer to both entities as the "target" (i.e. target of the gazing action) to explain how this controller works. Then, in the remaining of the paper, when we use the verbs "to gaze at" or "to fixate" we mean that this controller is activated.

If the target is visible (i.e. inside the image plane) head joints velocities are generated as follows:

$$\dot{\mathbf{q}}_{head}(t) = -G\mathbf{x}(t), \tag{1}$$

where $G \in \mathbb{R}^{3\times3}$ is a positive definite gain matrix and the position of the target $\mathbf{x} \in \mathbb{R}^3$ is defined as follows:

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} u_L - u_R \\ \dfrac{u_L + u_R}{2} \\ \dfrac{v_L + v_R}{2} \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ \dfrac{1}{2} & \dfrac{1}{2} & 0 & 0 \\ 0 & 0 & \dfrac{1}{2} & \dfrac{1}{2} \end{bmatrix} \begin{bmatrix} u_L \\ u_R \\ v_L \\ v_R \end{bmatrix} \tag{2}$$

being u_R and v_R the coordinates of the target on the right image plane and u_L and v_L the coordinates of the target on the left image plane.

The goal of the controller is to reduce to zero $[u_L\ u_R\ v_L\ v_R]^T$, which entails bringing the target in the center of both cameras (i.e. the fixation point). However, since $v_L = v_R$ (perceived targets have the same vertical position on both images) it is sufficient to reduce to zero $[u_L - u_R\ \frac{u_L+u_R}{2}\ \frac{v_L+v_R}{2}]^T$. This condition always holds in our system since we do not actuate the independent tilt of the eyes. In particular, the two cameras have been mechanically calibrated so as to guarantee that the alignment condition $(v_L = v_R)$ holds.

If the target is not visible a stereotyped motion strategy (i.e. random left-right and up-down movements of the neck) is used to detect it; then controller 1 is activated.

After fixation is achieved we encode the target position in space using the head joints values; since we actuate only 3DOF of the head the mapping from head joints to the target position is unique. If more DOFs of the head are used the redundancy should be solved by the gaze controller, as we did for instance in Ref. 13.

5.2. *Open-loop controller*

The control scheme for the open-loop controller is depicted in Fig. 2. The desired arm configuration which brings the hand in the fixation point is obtained by balancing two contributions with the coefficient α (which will be

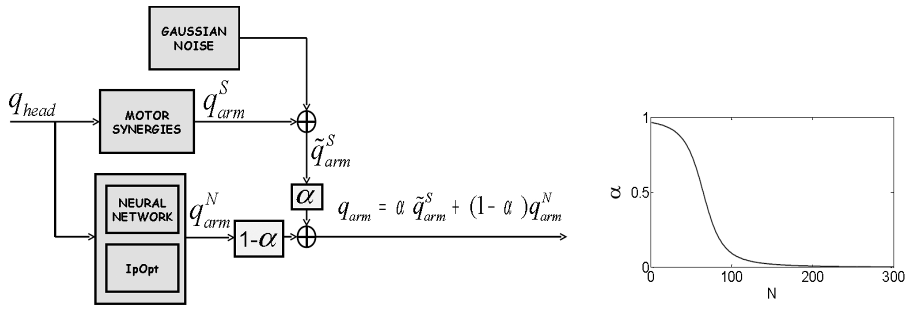


Fig. 2. On the left, the control scheme of the open-loop controller. On the right, the trend of the coefficient α in comparison with the growth of the arm-head forward kinematic map (namely, N, the number of points that have been learned).

discussed later):

$$\mathbf{q}_{arm} = \alpha \tilde{\mathbf{q}}^{S}_{arm} + (1-\alpha)\mathbf{q}^{N}_{arm}. \tag{3}$$

The first contribution $\tilde{\mathbf{q}}^{S}_{arm}$ is the output of the *Motor Synergies* block ($\mathbf{q}^{S}_{arm}$), plus an additional white Gaussian noise (which improves the exploration of the state space, as suggested by observations on infants development). The *Motor Synergies* block chooses one among four pre-defined arm configurations according to the current head configuration $\mathbf{q}_{head}$ in order to bring the hand roughly in the field of view (this is implemented as a nearest neighbor look-up table). These four arm configurations constitute the *a priori* knowledge provided to the system.

The second contribution $\mathbf{q}^{N}_{arm}$ is the output of the part of the controller that relies on learning. An RFWR neural network is trained online to approximate the arm-head forward kinematics, defined as $\mathbf{q}_{head} = fwdKin(\mathbf{q}_{arm})$: we call this model the *arm-head forward kinematic map*. This model maps an arbitrary arm configuration $\mathbf{q}_{arm}$ to the head configuration $\mathbf{q}_{head}$ that allows the fixation of the hand. The model is inverted in order to retrieve the appropriate arm configuration which brings the hand to the fixation point. Inversion is realized using IpOpt (Interior Point Optimizer[56]), a minimization algorithm which has been proven to be fast and reliable: a precise evalutation of its performance is provided in Ref. 44 for solving an inversion problem in a system which is very similar to the one presented here. To solve the redundancy of the arm ($\mathbf{q}_{arm} \in \mathbb{R}^4$ while $\mathbf{q}_{head} \in \mathbb{R}^3$), we choose the solution which minimizes the distance from the current arm configuaration $\mathbf{q}^{C}_{arm}$ to the target one. This choice prevents the controller from generating "useless" motion in joint space.

More formally, the target arm configuration $\mathbf{q}^{N}_{arm}$ is obtained as follows:

$$\mathbf{q}^{N}_{arm} = \mathbf{argmin}_{\mathbf{q}_{arm}\in\Omega} \|\mathbf{q}_{arm} - \mathbf{q}^{C}_{arm}\|^2, \tag{4}$$

$$s.t.\quad \mathbf{0} \leq \|\mathbf{q}_{head} - fwdKin(\mathbf{q}_{arm})\| \leq \boldsymbol{\epsilon}, \tag{5}$$

where $\Omega \equiv [\mathbf{q}^{L}_{arm}, \mathbf{q}^{U}_{arm}]$, being $\mathbf{q}^{L}_{arm}$ and $\mathbf{q}^{U}_{arm}$ the lower and upper bounds on $\mathbf{q}_{arm}$ (joints limits), and ϵ is an arbitrary low error threshold (we set $\epsilon = 0.0001$). In case condition 5 is not satisfied (this could happen if the target is not reachable) the error threshold is increased until a solution is found; if no solution is found the controller uses $\tilde{\mathbf{q}}^{S}_{arm}$ alone.

The coefficient α is a function of the number of points that have been used to train the arm-head forward kinematic map: this means that in the initial stages of development the robot mainly moves using the pre-coded motor synergies, and it gradually switches to the use of the learned model with the growth of the learned data set. The sigmoid function which relates the increase of N to the decrease of α, depicted in Fig. 2 on the right, is described by the following equation:

$$\alpha(N) = a - \frac{a\left(\frac{N}{b} - b\right)}{\sqrt{b + \left(\frac{N}{b} - b\right)^2}} \tag{6}$$

in which we chose $a = 0.5$ and $b = 8$ for our implementation. These parameters have been tuned manually (i.e. defining the shape of the sigmoid function) in order to have the neural network trained with a minimum number of samples before using it for the control (e.g. for our system, about 100 training samples are enough to achieve a decent estimation of the forward model). Furthermore, thanks to a number of simulations (not reported in this paper), we determined that the proposed method is quite robust with respect to the choice of these parameters.

5.3. *Closed-loop controller*

The closed-loop controller exploits a learned inverse model of the visuo-arm Jacobian to cancel the visual error (i.e. the distance of the hand from the object): we will refer to this model as the *visuo-arm inverse map*. This controller is activated after the open-loop controller, when the hand has arrived close to the target object (i.e. inside the central part of the visual field). The visuo-arm inverse map is implemented as an RFWR neural network which is trained online with the data collected by the robot during the movements. Differently from previous work, data gathering and training of the neural network are performed during the robot movements, whenever the hand moves inside the visual field (also during the ballistic movement). The input for the RFWR is the robot configuration $\mathbf{q} = [\mathbf{q}_{arm}\ \mathbf{q}_{head}]$ and the measured hand displacement in the visual field ($\Delta\mathbf{x}$). The output is the arm joints displacement ($\Delta\mathbf{q}_{arm}$). The RFWR learns the following mapping:

$$\Delta\mathbf{q}_{arm} = f^{J_{\text{inv}}}(\mathbf{q}, \Delta\mathbf{x}) \tag{7}$$

that approximates the arm inverse kinematics. The inclusion of the robot configuration in the input vector and the use of a spatially localized learning algorithm (RFWR) allow to learn a valid inverse solution, as explained before in Sec. 3. In particular, when the training data present multiple $\Delta\mathbf{q}^{i}_{arm}$ in the vicinity of $\mathbf{q}^*$ mapping the same $\Delta\mathbf{x}$, the learned solution is the average of all the solutions, $\langle\Delta\mathbf{q}^{i}_{arm}\rangle_i$. This is still a valid solution, as proven in Ref. 45, because the following relation holds:

$$\Delta\mathbf{x} = \langle\Delta\mathbf{x}\rangle_i = \left\langle J(\mathbf{q}^*)\Delta\mathbf{q}^{i}_{arm}\right\rangle_i = J(\mathbf{q}^*)\left\langle\Delta\mathbf{q}^{i}_{arm}\right\rangle_i \tag{8}$$

due to the local linearity of the Jacobian $J(\mathbf{q})$.
The closed-loop controller generates motor velocities $\dot{\mathbf{q}}_{arm}$ at 200 Hz rate by using the following equation:

$$\dot{\mathbf{q}}_{arm}(t) = Kf^{J_{\text{inv}}}(\mathbf{q}(t), \mathbf{x}(t)), \tag{9}$$

where $K \in \mathbb{R}^{4\times4}$ is a positive definite gain matrix, and $f^{J_{\text{inv}}}(\mathbf{q}(t), \mathbf{x}(t))$ is the visuo-arm inverse map. The goal of the controller is to reduce $\mathbf{x}$ to zero by moving the arm (i.e. to bring the hand to the fixation point), therefore the visuo-arm inverse map is queried with $\mathbf{x}$ instead of $\Delta\mathbf{x}$ ($\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}^d = \mathbf{x} - 0 = \mathbf{x}$).

The output of the RFWR (in this case, the visuo-arm inverse map) is null if no local model is associated with the current input. This always happens in the beginning, when still no samples have been collected and used for training, but can occur also later, for instance if the system works in regions of the motor space that have not been explored yet. If the estimated inverse model is null, the motor commands generated by the controller 9 are equal to zero. Anyway the robot must move in order to collect data for training the neural network. Therefore, when the RFWR outputs zero we modify Eq. (9) as follows:

$$\dot{\mathbf{q}}_{arm}(t) = KJ_{\text{inv}}^{\text{INIT}}(\mathbf{q}_{arm})\mathbf{x}(t), \tag{10}$$

where $J_{\text{inv}}^{\text{INIT}}(\mathbf{q}_{arm}) \in \mathbb{R}^{4\times 3}$ is a matrix representing a coarse estimation of the inverse Jacobian (and therefore $J_{\text{inv}}^{\text{INIT}}(\mathbf{q}_{arm})\mathbf{x}(t)$ can replace the $f^{J_{\text{inv}}}(\mathbf{q}(t), \mathbf{x}(t))$ term in Eq. 9). This matrix is the outcome of what we call the *initialization map*: an RFWR neural network which is queried with the current $\mathbf{q}_{arm}$ as input, and gives a local $J_{\text{inv}}^{\text{INIT}}$ as output.
The neural network is updated during the movement as follows:

(1) **collect new sample** $(\Delta\mathbf{x}, \Delta\mathbf{q}_{arm})_i$ at time i
(2) **retrieve** $J_{\text{inv}}^{\text{INIT}} i = RFWR(\mathbf{q}_{arm}^{i})$
(3) **update** $J_{\text{inv}}^{\text{INIT}} i$ with $(\Delta\mathbf{x}, \Delta\mathbf{q}_{arm})_i \rightarrow J_{\text{inv}}^{\text{INIT}} i+1$
(4) **train** $RFWR(\mathbf{q}_{arm}^{i}, J_{\text{inv}}^{\text{INIT}} i+1)$

where the update of $J_{\text{inv}}^{\text{INIT}} i$ from $(\Delta\mathbf{x}, \Delta\mathbf{q}_{arm})_i$ at step 3 is done with incremental least squares, as in Ref. 2. When this RFWR outputs zero (no local model is associated with the input) an arbitrary constant matrix is used as output. This ensures that the closed-loop controller is always able to generate motion. In our implementation we chose a matrix with all coefficients equal to one $(J_{\text{inv}}^{\text{INIT}}(i,j) = 1\ \forall i,j)$. The motion generated using such a matrix will not necessarily bring the hand closer to the target but will produce exploratory movements. As for the open-loop controller, the choice to rely on noisy initial motor commands has been suggested by observations on infants development. In Fig. 3, we show a closed-loop movement using the matrix with all coefficient equal to one, without any update (left image), and the improvements provided by the online learning (right image). For visualization purposes, we plot only two components of $\mathbf{x}$, namely x_1 and x_2, since these are the dimensions on which more motion is visible. However, the trend of the convergence to zero of x_0 is similar to the other two components. This way of depicting visual trajectories will be kept in the remainder of the paper. The light gray cross indicates the starting point, while the dark gray cross is the goal (center of the image, i.e. fixation point). While in the left image the hand is driven in a direction opposite to the target (and it moves out of the visual field), in the right image the updated matrix eventually brings the hand in the fixation point, with a straight trajectory. Here we are considering $J_{\text{inv}}^{\text{INIT}}$ as a function of the arm configuration only; theoretically, $J_{\text{inv}}^{\text{INIT}}$ is a function of both the arm configuration and the head configuration. Anyway, if we consider movements of the hand near the fixation point, we can reasonably approximate $J_{\text{inv}}^{\text{INIT}}(\mathbf{q}_{arm}, \mathbf{q}_{head})$ with

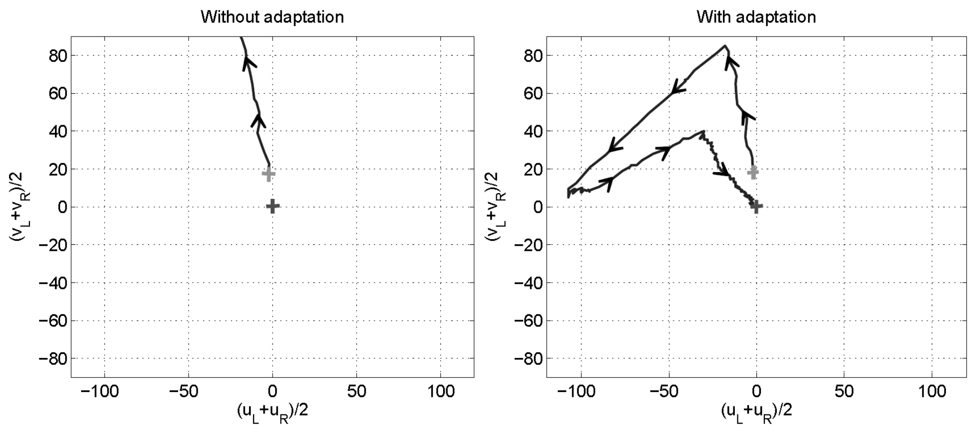


Fig. 3. Online adaptation of the initialization map. On the left, the closed-loop visual trajectory using the initialization matrix with no update. On the right, the map is updated online. The light gray cross is the starting point, the dark gray cross is the goal (center of the image, i.e. fixation point).

$J_{\mathrm{inv}}^{\mathrm{INIT}}(\mathbf{q}_{arm})$, being $\mathbf{q}_{head}$ a function of $\mathbf{q}_{arm}$ ($\mathbf{q}_{head} = fwdKin(\mathbf{q}_{arm})$). This approximation has been used in Ref. 13 and is explained with more details there. The more the hand drifts away from the image center (i.e. the fixation point), the larger the approximation error. As for the initialization map we do not require high accuracy (this map is then replaced by the visuo-arm inverse map) but we do need fast adaptation, we decided to define it as a function of the arm configuration only: learning in such a reduced state space can be performed faster.

Summarizing, during the closed-loop control the two maps are used as follows:

- at every control step both the initialization map and the visuo-arm inverse map are queried;
  - — if the visuo-arm inverse map gives a non-null response it is used for control, exploiting Eq. (9);
  - — if the visuo-arm inverse map gives a null response, the initialization map is used instead, as described in Eq. (10).

5.4. *Goal-directed exploration*

We describe here the goal-directed behavior of the robot, and how data is gathered and used to train the RFWR neural networks. Exploration, learning and execution collapse in the same process, which is fully autonomous and continuous. Practically speaking, the robot is switched on and operates without external intervention from the experimenter (except from showing some objects periodically).

The behavior of the robot comprises the steps listed hereinafter.

- A simple attention system based on color segmentation (whose details are not relevant for the understanding of the paper) provides the robot with a salient point

in the image planes, which corresponds to a target object in space. The robot gazes at the object.
- Once the robot has fixated the object, the open-loop controller is activated.
- When the hand falls inside the central part of visual field (or when the ballistic movement is finished) the closed-loop controller is activated.
- If the hand does not fall inside the visual field (the closed-loop controller cannot be activated) the robot gazes at the hand (this condition rarely happens).
- During the closed-loop control the robot continuously checks the visual error:
  - if the error increases too much (the hand is falling outside the visual field) or if it has not been zeroed after a certain amount of time (the target lies outside the robot reachable space), the robot stops moving the arm and gazes at the hand;
  - otherwise the closed-loop movement continues until the visual error is reduced to zero.
- During both open-loop and closed-loop movements data are collected and used to update the visuo-arm inverse map.
- As soon as the reaching action is finished (either successfully or not) the robot gazes at a new target (provided by the attention system).

Whenever the robot is fixating the hand, the arm-head forward kinematic map is trained with the arm configuration $\mathbf{q}_{arm}$ as input and the head configuration $\mathbf{q}_{head}$ as output. Fixation of the hand occurs either when the reaching controller brings the hand to fixation or when the robot gazes at the hand (i.e. in case of both successful and unsuccessful reaching actions). That being so, it becomes more clear how the addition of noise to the motor synergies could improve the initial exploration: if the closed-loop controller is not activated (the hand does not fall inside the visual field after the open-loop movement) this noise extends the variability of the data gathered to train the arm-head forward kinematic map. Furthermore, whenever the hand is moving inside the visual field the visuo-arm inverse map is trained with the robot configuration $\mathbf{q} = [\mathbf{q}_{arm}\ \mathbf{q}_{head}]$ and the measured hand displacement ($\Delta\mathbf{x}$) as input, and with the arm joints displacement ($\Delta\mathbf{q}_{arm}$) as output.

5.5. *Representation of the reachable space*

The results of reaching actions are used to learn incrementally a representation of the robot reachable space: a *reachable space map*. This map allows the robot to estimate whether a fixated point in space (or a fixated object) is reachable or not. The map is implemented using an RFWR neural network which is trained online during the sequence of reaching trials: the input is the head configuration $\mathbf{q}_{head}$ and the output is a value $S \in \{0, 1\}$ which indicates the failure/success of the reaching action.

Every time the arm-head forward kinematic map is trained with a new sample $\langle \mathbf{q}_{arm}, \mathbf{q}_{head} \rangle$, the reachable space map is trained with $\langle \mathbf{q}_{head}, S = 1 \rangle$, because a feasible arm configuration which brings the hand in the fixation point defined by

$\mathbf{q}_{head}$ exists. Conversely, if a reaching task has not been accomplished the map is trained with $\langle \mathbf{q}_{head}, S = 0 \rangle$.

Some reaching trials could fail even if the target lies inside the workspace: for example because in the beginning of learning both the arm-head forwad kinematic map and the visuo-arm inverse map are not sufficiently precise. This generates incorrect training data for the reachable space map. Nevertheless, this can be interpreted as noise in the input data, decreasing as the whole reaching system develops. This noise should not affect learning considerably, since the learning network is characterized by a forgetting factor that decreases as more samples are learned: the network forgets more at the beginning and less as learning occurs.

When queried with the head configuration (i.e. the fixation point) as input, the map gives as output a real number estimating the probability of the fixated point to be reachable.

6. Experimental Results

We conducted an experiment to determine the performances of the learning. The robot executed 1200 reaching trials toward randomly distributed 3D target positions in space. All these positions were inside the reachable workspace. This space has been defined in the head configuration space by limiting the head joint angles as follow: $\mathbf{q}_{head}^{\min} = [-2.0^\circ\ 20^\circ\ -20^\circ]$ and $\mathbf{q}_{head}^{\max} = [2.0^\circ\ 60^\circ\ 20^\circ]$. The special 3D positions of the targets do not influence the learning performances as long as they cover the robot reachable workspace more or less uniformly (i.e. without leaving big unexplored areas); this has been verified in simulation using the dynamic simulator of the iCub robot,[57] which has been modified to match the kinematics of our robot James. Therefore, the experiment we conducted on the real robot can be considered general enough.

Apart from some initial reaching trials (among the first 25), the robot was always able to reach for the desired targets. The sequence of images in Fig. 4 shows the improvements of the trajectories of the hand in the robot visual field; only some representative trajectories are shown (among the first 500 reaching trials). The closed-loop control is activated as soon as the hand enters in the central part of the visual field (the highlighted central square).

After 500 reaching trials, the position of the hand marker (i.e. the green ball attached to the robot wrist, as described in Sec. 4) was moved of 6 cm. This change was artificially induced to simulate different situations, like a modification of the robot kinematics (e.g. links length) or the use of a tool for reaching.

Then, several tests have been performed to analyze the performance of the system at different learning stages. In particular, we present results concerning the open-loop controller (i.e. ballistic reaching) and the closed-loop controller separately, in Sec. 6.1 and in Sec. 6.2, respectively.

Finally, the learning process has been executed again from the beginning with 1400 reaching trials directed to both reachable and nonreachable 3D target positions,

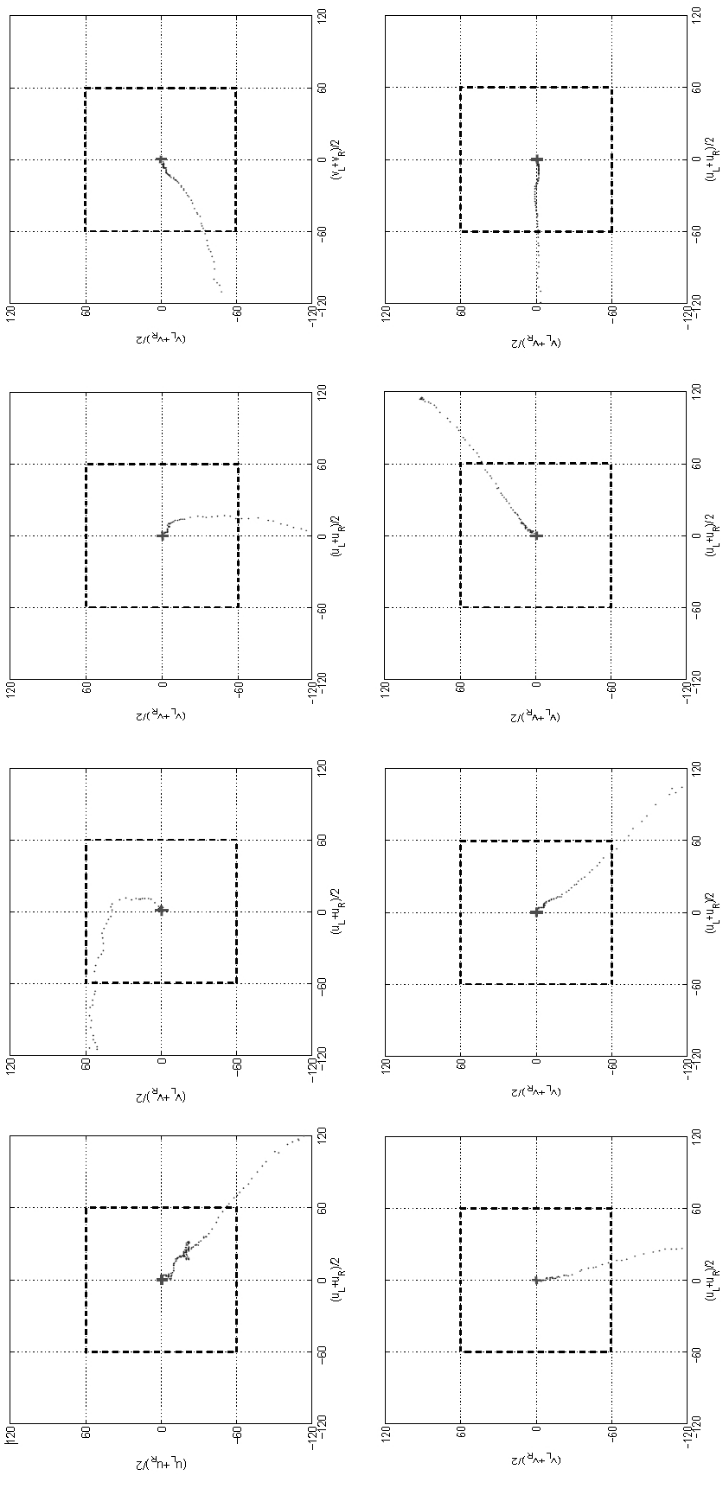


Fig. 4. Visual trajectories of reaching movements composed by open-loop and closed-loop phase. The closed-loop control is activated as soon as the hand enters in the central part of the visual field (the highlighted central square). Only some representative trajectories are shown, among the first 500 reaching trials.

in order to build a reachable space map. Here the head joint angles limits were: $\mathbf{q}_{head}^{\min} = [-5.0° \; 10° \; -30°]$ and $\mathbf{q}_{head}^{\max} = [5.0° \; 90° \; 30°]$. Results concerning the estimation of the reachable space map are provided in Sec. 6.3.

6.1. *Ballistic reaching adaptation*

In order to precisely assess the accuracy of ballistic reaching, we tested the robot while reaching for eight different positions within its workspace by using only the open-loop controller. These positions have been chosen to cover the robot workspace nearly uniformly. The attention system has been replaced here by a "simulated" attention system which was just driving the robot fixation toward precise 3D points in space (i.e. moving the head toward eight different pre-selected configurations). After the robot has fixated each point, ballistic reaching is performed (without any correction based on visual feedback) until the arm reaches its target configuration; at this point the visual error $\|\mathbf{x}\|$ is computed.

The sequence of eight movements has been repeated several times, using the arm-head forward kinematic map at different stages of the learning (i.e. increasing the number of learned samples). In Table 1, the eight different head configurations (corresponding to eight different target points in space) used during the test are reported. Figure 5 plots the RMSE (Root Mean Square Error) on the eight target

Table 1. The eight head configurations (corresponding to eight different 3D target points in space) used in the test experiment for the assessment of ballistic reaching performances.

#	1	2	3	4	5	6	7	8
Vergence	−1.0°	−1.0°	−0.5°	0.0°	0.5°	1.0°	1.5°	1.5°
Yaw	25°	30°	40°	35°	40°	50°	45°	55°
Pitch	5°	15°	0°	−10°	10°	0°	−5°	−15°

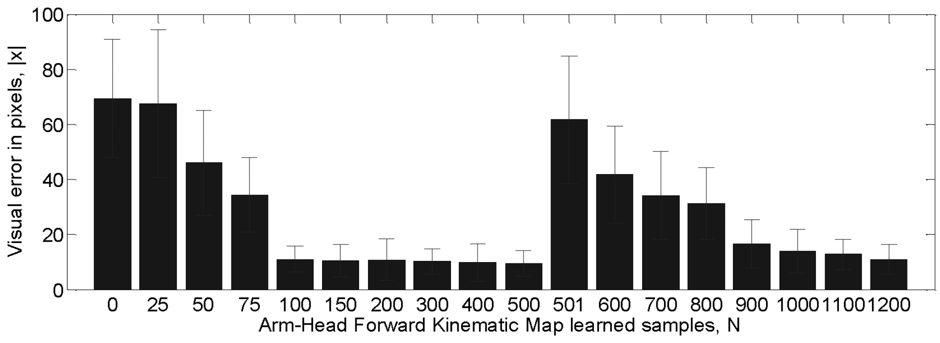


Fig. 5. Position error of the end-effector after ballistic reaching. On the x-axis the number of training samples of the arm-head forward kinematic map is reported (which is also the number of reaching trials during the goal-directed exploration). Ballistic motions toward eight target locations within the robot workspace have been performed at different (discrete) learning stages. RMSE of the position errors at each learning stage is reported on y-axis (mean and standard deviation of the eight different target locations).

points (the vertical bars indicate the standard deviation). As expected, the RMSE decreases as more samples are learned by the robot. Since the robot collects one training sample for each reaching trial during the goal-directed exploration, the number of samples is equal to the number of trials. The RMSE after 500 trials is about 10 pixels, with 5 pixels of standard deviation. Considering the average absolute position of the hand with respect to the eye cameras, 1 pixel corresponds to about 1.0 cm.

The change in the hand marker position introduced at trial 500 causes a sudden increase of the RMSE, which raises to 62 pixels. Nevertheless, the robot continues to update the arm-head forward kinematics map and as a consequence the RMSE decreases again as more and more samples are collected and used for training.

The ballistic controller relies on the knowledge of the arm-head forward kinematic map, which is learned incrementally through experience. Therefore, we expect the error of the ballistic controller to be consistent with the arm-head forward kinematic map estimation error. To verify this, we computed the instantaneous estimation error of the arm-head forward kinematic map during the learning process described in Sec. 5.4: every time a new sample is collected the map is queried with that input sample, the estimation error is computed, and then the sample is used to update the map. Figure 6 depicts the trend of this error as the number of training samples increases. The residual estimation error justifies the position error shown by the ballistic controller in Fig. 5. Again, the error suddenly increases when the marker position is changed, at trial 500, and then is progressively reduced.

Moreover, around trial 800 and 1000 the robot has been switched off and then recalibrated. In our system, calibration of the motor encoders is based on absolute position sensors, which are noisy. This induces small changes in the kinematic model after every recalibration. As a consequence, the estimation error raises a bit and then

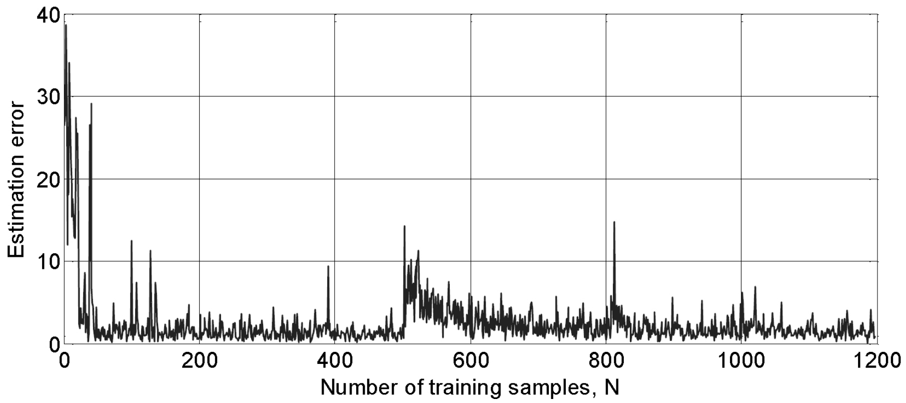


Fig. 6. Estimation error of the arm-head forward kinematic map (average of the three output dimensions). On the x-axis the number of samples used for training (which is equal to the number of reaching trials performed).

quickly decreases, as evident in the plot. This is a further example of the effectiveness of online learning as opposed to the employment of a fixed analytical model.

6.2. *Closed-loop control adaptation*

The performance of the closed-loop controller (which depends on the quality of the visuo-arm inverse map estimation) has been tested by using it to drive the hand toward eight symmetrical targets around the robot fixation point (forward and backward motion), while keeping the gaze still; x_0 (which is not shown in the plots) was controlled to zero, while x_1 and x_2 were controlled to either -50, $+50$ or zero (pixels).

Figure 7 reports the results obtained at different learning stages (i.e. map trained with increasing number of samples). The first plot on the left shows the trajectories after 25 reaching trials. At that point, the visuo-arm inverse map had been trained with about 400 samples; the initialization map was not used anymore, since the visuo-arm inverse map always gave a non-null response if queried. Anyway, the visual trajectories are very irregular, indicating a poor quality of the estimation; in fact, a perfect estimation would produce straight trajectories in the visual space. A good estimation is obtained after 1200 reaching trials (last plot on the right, map trained with about 15,000 samples), as trajectories are almost straight. The change in the position of the hand visual marker at trial 500 causes the trajectories to change suddenly, even if, differently from what happened for the open-loop control, this

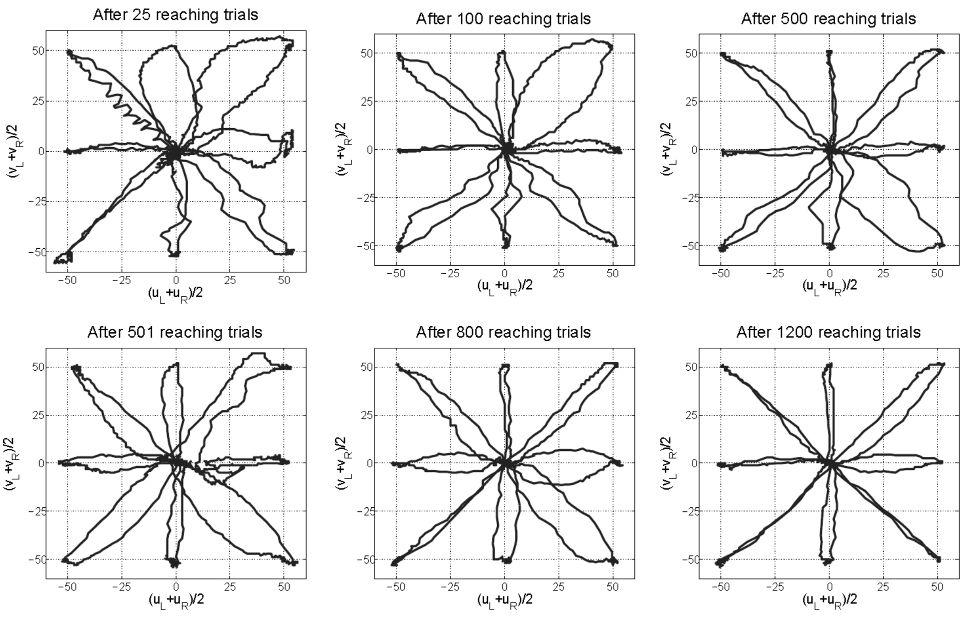


Fig. 7. Evolution of the closed-loop visual trajectories in performing the test movement. As learning progresses trajectories become more and more straight.

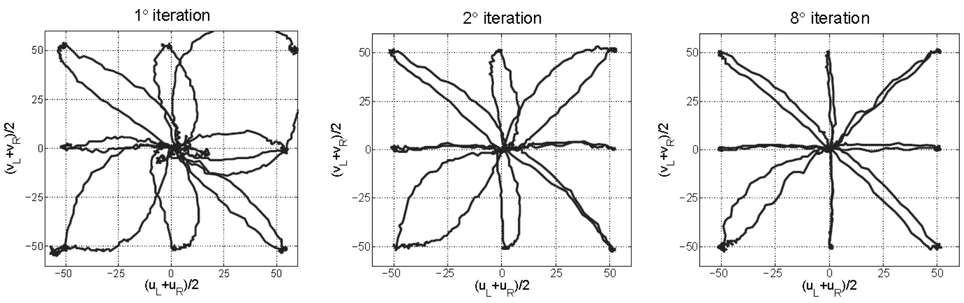


Fig. 8. Evolution of the closed-loop visual trajectories in performing the test movement after a 45° CCW rotation of the perceived visual space.

affected the performances of the closed-loop control to a minor extent; in fact, such a structural modification does not change the visuo-arm Jacobian much.

To test the ability of the robot to adapt to changes in the visuo-arm Jacobian, a 45° counterclockwise (CCW) rotation around the x_0 axis of the visual space was introduced. The perceived position of the hand was modified as follows:

$$\begin{aligned} x_0^{\mathrm{MOD}} &= x_0 \\ x_1^{\mathrm{MOD}} &= \cos 45^\circ x_1 - \sin 45^\circ x_2 \\ x_2^{\mathrm{MOD}} &= \cos 45^\circ x_1 + \sin 45^\circ x_2 \end{aligned}$$

The consequences of this kinematic perturbation can be seen in Fig. 8: while in the first iteration of the test movement trajectories were curved and ungracious (first plot on the left), already during the second iteration the robot was showing improvements, recovering a good estimation after eight iterations. A similar effect was observed when the perturbation was removed (the related plots are not shown here).

6.3. *Reachable space estimation*

Figure 9 depicts two different visualizations of the reachable space map after 1400 samples have been gathered and used for training: a 3D plot in the head configuration space, on the left, and a projection of the same map on the pitch/vergence plane (with yaw = 90°), on the right. Figure 10 shows projections of the map on the vergence/yaw plane, considering different values of pitch orientation, precisely $\langle -30^\circ; -15^\circ; 0^\circ; 15^\circ; 30^\circ \rangle$. Black color means high values of the map output (high probability of the point to be reachable). Generally, it can be noticed that the map output increases with increasing values of vergence (i.e. fixated point closer to the robot eyes). Morover, the output is overall higher in the central part of the yaw axis (these are positions in space that the robot can reach more easily, with the arm joints far from the physical limits); on the other hand, for yaw values close to 90° the map output is typically low, especially if vergence is negative.

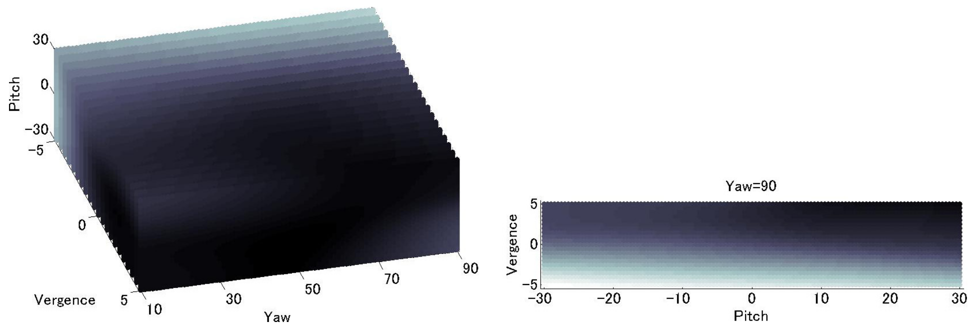


Fig. 9. The reachable space map, after training with 1400 samples. Dark gray (black) means high probability to be reachable, light gray (white) means low probability. On the left, 3D visualization in head configuration space. On the right, projections on the vergence/pitch plane, with yaw = 90°.

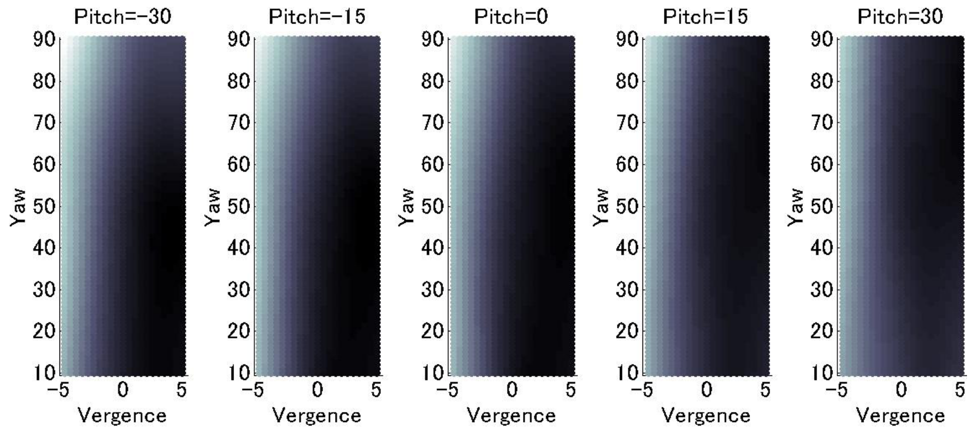


Fig. 10. The reachable space map, after training with 1400 samples. Projections on the vergence/yaw plane, considering different values of pitch orientation, $\langle -30°; -15°; 0°; 15°; 30° \rangle$. Dark gray (black) means high probability to be reachable, light gray (white) means low probability.

If we analyze the robot physical structure (which is indeed similar to the human one), it is clear that if we consider targets placed at a certain distance from the eyes (encoded by vergence) they are more likely to be reachable if the neck is bent forward (positive pitch) rather than backward (negative pitch), being the shoulder positioned below the head (see Fig. 1). Noticeably, increasing the pitch value the map output points become generally higher. This is also evident in the right image in Fig. 9, where head yaw is 90°.

7. Conclusion

In this paper, we propose a novel strategy which allows a humanoid robot to learn autonomously and online how to reach for visually identified objects. Differently from previous works, in our approach there is no separation between the exploration and the exploitation phase; on the contrary, the robot shows a continuous goal-directed behavior, whose performance evolves with time. The reaching controller combines an open-loop controller that brings the hand near to the target and a closed-loop controller that cancels the residual hand position error. The open-loop controller relies on an arm-head forward kinematic model, while the closed-loop controller exploits an inverse model of the visuo-arm Jacobian. Both models are learned autonomously by the robot during the execution of reaching actions towards visual stimuli. We show experimentally how the errors of the controllers are progressively reduced, and the reaching trajectories become rectilinear and regular, as learning is performed. The ability of the robot to continuously adapt to changes in the kinematic structure or sensory system has been demonstrated by simulating a sudden change in one of the arm links and by introducing a perturbation (45° CCW rotation) in the perceived visual space. Furthermore, the robot is able to learn a representation of its reachable space, based on motor information, while performing reaching attempts directed to both reachable and nonreachable points.

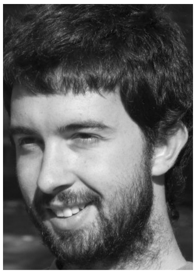
References

1. M. R. Blackburn and H. G. Nguyen, Learning in robot vision directed reaching: A comparison of methods, in *ARPA Image Understanding Workshop* (1994), pp. 781–788.
2. K. Hosoda and M. Asada, Versatile visual servoing without knowledge of true Jacobian, in *IEEE Int. Conf. Intelligent Robots and Systems (IROS)* (IEEE Press, Munich, Germany, 1994), pp. 186–193.
3. M. Marjanovic, B. Scassellati and M. Williamson, Self-taught visually guided pointing for a humanoid robot, in *IEEE Int. Conf. Simulation of Adaptive Behavior* (IEEE Press, Cape Cod, USA, 1996), pp. 35–44.
4. S. Rougeaux and Y. Kuniyoshi, Robust tracking by a humanoid vision system, in *Int. Work. Humanoid and Human Friendly Robotics* (1998).
5. G. Metta, G. Sandini and J. Konczak, A developmental approach to visually-guided reaching in artificial systems, *Neural Netw.* **12**(10) (1999) 1413–1427.
6. G. Metta, F. Panerai, R. Manzotti and G. Sandini, Babybot: An artificial developing robotic agent, in *IEEE Int. Conf. Simulation of Adaptive Behavior* (Paris, France, 2000).
7. S. Vijayakumar, A. D'Souza, T. Shibata, J. Conradt and S. Schaal, Statistical Learning for Humanoid Robots, *Auton. Robots* **12**(1) (2002) 55–69.
8. C. Gaskett and G. Cheng, Online learning of a motor map for humanoid robot Reaching, in *Int. Conf. Computational Intelligence, Robotics and Autonomous Systems (CIRAS)* (Singapore, 2003).
9. J.-T. Lapresté, F. Jurieand, M. Dhome and F. Chaumette, An efficient method to compute the inverse Jacobian matrix in visual servoing, in *IEEE Int. Conf. Robotics and Automation (ICRA)* (IEEE Press, New Orleans, USA, 2004), pp. 727–732.
10. G. Sun and B. Scassellati, A fast and efficient model for learning to reach, *Int. J. Humanoid Robotics* **2**(4) (2005) 391–413.
11. M. Lopes and J. Santos-Victor, Learning sensory-motor maps for redundant robots, in *IEEE Int. Conf. Intelligent Robots and Systems (IROS)* (IEEE Press, Beijin, China, 2006), pp. 2670–2676.
12. N. Mansard, M. Lopes, J. Santos-Victor and F. Chaumette, Jacobian learning methods for tasks sequencing in visual servoing, in *IEEE Int. Conf. Intelligent Robots and Systems (IROS)* (IEEE Press, Beijin, China, 2006), pp. 4284–4290.

13. L. Natale, F. Nori, G. Metta and G. Sandini, Learning precise 3D reaching in a humanoid robot, in *IEEE Int. Conf. Development and Learning (ICDL)* (IEEE Press, London, UK, 2007), pp. 324–329.
14. M. Hersch, E. Sauser and A. Billard, Online learning of the body schema, *International Journal of Humanoid Robotics* **5**(2) (2008) 161–181.
15. J. Sturm, C. Plagemann and W. Burgard, Unsupervised body scheme learning through self-perception, in *IEEE Int. Conf. Robotics and Automation (ICRA)* (IEEE Press, Pasadena, USA, 2008), pp. 3328–3333.
16. J. Peters and S. Schaal, Learning to control in operational space, *Int. J. Robotics Res.* **27**(2) (2008) 197–212.
17. C. Salaun, V. Padois and O. Sigaud, Learning forward models for the operational space control of redundant robots, in *From Motor Learning to Interaction Learning in Robots* (Springer, 2010), pp. 169–192.
18. R. Martinez-Cantin, M. Lopes and L. Montesano, Body schema acquisition through active learning, in *IEEE Int. Conf. Robotics and Automation (ICRA)* (IEEE Press, Anchorage, USA, 2010), pp. 1860–1866.
19. I. Eibl-Eibesfeld, *Ethology: the Biology of Behavior* (Holt, Rinehart and Winston, 1970).
20. J. L. Gould, *Ethology: The Mechanism and Evolution of Behavior* (Norton, 1982).
21. J. Konczak, On the notion of motor primitives in humans and robots, in *Int. Work. Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems* (2005), pp. 47–53.
22. B. L. White, P. Castle and R. Held, Observations on the development of visually guided reaching, *Child dev.* **35** (1964) 349–364.
23. E. W. Bushnell, The Ontogeny of Intermodal Relations: Vision and Touch in Infancy, in *Intersensory Perception and Sensory Integration* (New York: Plenum Press, 1981) pp. 5–37.
24. C. Von Hofsten, Eye-hand coordination in the newborn, *Developmental Psychology* **18** (1982) 450–461.
25. J. F. Soechting and M. Flanders, Sensorimotor representations for pointing to targets in three-dimensional space, *J. Neurophysiol.* **62** (1989) 582–594.
26. J. McIntyre, F. Stratta and F. Lacquaniti, Viewer-centered frame of reference for pointing to memorized targets in three-dimensional space, *J. Neurophysiol.* **62** (1997) 582–594.
27. M. Flanders, L. Daghestani and A. Berthoz, Reaching beyond reach, *Exp. Brain Res.* **126**(1) (1999) 19–30.
28. E. Bushnell, The decline of visually guided reaching during infancy, *Infant Behav. Dev.* **8** (1985) 139–155.
29. C. Von Hofsten, Structuring of early reaching movements: a longitudinal study, *J. Motor Behav.* **23**(4) (1991) 280–292.
30. H. Kinney, B. Brody, A. Kloman and F. Gilles, Sequence of central nervous system myelination in human infancy. ii. patterns of myelination in autopsied infants, *J. Neuropathology Exp. Neurology* **47**(3) (1988) 217–234.
31. A. Mathew and M. Cook, The control of reaching movements by young infants, *Child Dev.* **61**(4) (1990) 1238–1257.
32. D. Ashmead, M. McCarty, L. Lucas and M. Belvedere, Visual guidance in infants' reaching toward suddenly displaced targets, *Child Dev.* **64**(4) (1993) 1111–1127.
33. J. Konczak, M. Borutta, H. Topka and J. Dichgans, Development of goal-directed reaching in infants: Hand trajectory formation and joint force control, *Exp. Brain Res.* **106** (1995) 156–168.
34. A. Maravita and A. Iriki, Tools for the body (schema), *Trends Cogn. Sci.* **8**(2) (2004) 79–86.
35. J. W. Krakauer, M. F. Ghilardi and C. Ghez, Independent learning of internal models for kinematic and dynamic control of reaching, *Nature Neurosci.* **2**(11) (1999) 675–694.

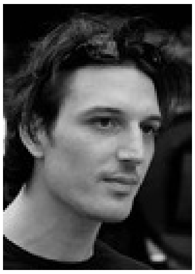
36. J. W. Krakauer, Z. M. Pine, M. F. Ghilardi and C. Ghez, Learning of visuomotor transformations for vectorial planning of reaching trajectories, *J. Neurosci.* **23**(20) (2000) 8916–8924.
37. Y. Coello, A. Bartolo, B. Amiti, H. Devanne, E. Houdayer and P. Derambure, Perceiving what is reachable depends on motor representations: Evidence from a transcranial magnetic stimulation study, *PLoS One* **3**(8) (2008) e2862.
38. P. Cardellicchio, C. Sinigaglia and M. Costantini, The space of affordances: a TMS study, *Neuropsychologia* **49**(5) (2011) 1369–1372.
39. A. Hutchinson, G. D. Hager and P. I. Corke, A tutorial on visual servo control, *Trans. Robotics Autom.* **12**(5) (1996) 651–670.
40. L. Jamone, L. Natale, M. Fumagalli, F. Nori, G. Metta and G. Sandini, Machine-learning based control of a human-like tendon driven neck, in *IEEE Int. Conf. Robotics and Automation (ICRA)* (IEEE Press, Anchorage, USA, 2010), pp. 859–865.
41. J. Baillieul, Kinematic programming alternatives for redundant manipulators, in *IEEE Int. Conf. Robotics and Automation (ICRA)* (IEEE Press, St. Louis, USA, 1985), pp. 722–728.
42. L. Sciavicco and B. Siciliano, A solution algorithm for the inverse kinematic problem for redundant manipulator, *Int. J. Robotics Autom.* **4** (1988) 403–410.
43. A. Ligeois, Automatic supervisory control of the configuration and behavior of multibody mechanisms, *Trans. Syst. Man Cybern.* **7** (1977) 868–871.
44. U. Pattacini, F. Nori, L. Natale, G. Metta and G. Sandini, An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots, in *IEEE Int. Conf. Intelligent Robots and Systems (IROS)* (IEEE Press, Taipei, Taiwan, 2010), pp. 1668–1674.
45. D. Bullock, S. Grossberg and F. H. Guenther, A selforganizing neural model of motor equivalent reaching and tool use by a multijoint arm, *J. Cogn. Neurosci.* **5**(4) (1993) 408–435.
46. L. Jamone, L. Natale, K. Hashimoto, G. Sandini and A. Takanishi, Learning task space control through goal directed exploration, in *IEEE Int. Conf. Robotics Biomimetics (ROBIO)* (IEEE Press, Phuket, Thailand, 2011).
47. Y. Guan, K. Yokoi and X. Zhang, Numerical methods for reachable space generation of humanoid robots, *Int. J. Robotics Res.* **27** (2008) 935–950.
48. F. Zacharias, C. Borst and G. Hirzinger, Capturing robot workspace structure: Representing robot capabilities, in *IEEE Int. Conf. Intelligent Robots and Systems (IROS)* (IEEE Press, San Diego, USA, 2007), pp. 3229–3236.
49. L. Jamone, F. Nori, G. Metta and G. Sandini, James: A humanoid robot acting over an unstructured world, in *IEEE Int. Conf. Humanoid Robots* (IEEE Press, Genoa, Italy, 2006), pp. 143–150.
50. F. Nori, L. Jamone, G. Metta and G. Sandini, Accurate control of a human-like tendon driven neck, in *IEEE Int. Conf. Humanoid Robots* (IEEE Press, Pittsburgh, USA, 2007), pp. 371–378.
51. M. Fumagalli, L. Jamone, A. Parmiggiani, F. Nori, L. Natale, G. Metta, M. Randazzo, and G. Sandini, A force sensor for the control of a human-like tendon driven neck, in *IEEE Int. Conf. Humanoid Robots* (IEEE Press, Paris, France, 2009), pp. 478–485.
52. G. Metta, P. Fitzpatrick and L. Natale, YARP: Yet another robot platform, *Int. J. Adv. Robotics Syst.* **3**(1) (2006) 43–48.
53. S. Schaal and C. G. Atkenson, Constructive incremental learning from only local information, *Neural Comput.* **10**(8) (1998) 2047–2084.
54. S. Vijayakumar and S. Schaal, Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space, in *Int. Conf. Machine Learning (ICML)* (San Francisco, USA, 2000), pp. 1079–1086.

55. D. Nguyen-Tuong, M. Seeger and J. Peters, Model learning with local gaussian process regression, *Adv. Robotics* **23**(15) (2009) 2015–2034.
56. A. Wachter and L. T. Biegler, On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming, *Math. Prog.* **106**(1) (2006) 25–57.
57. V. Tikhanoff, P. Fitzpatrick, G. Metta, L. Natale, F. Nori and A. Cangelosi, An open source simulator for cognitive robotics research: The prototype of the iCub humanoid robot simulator, in *Workshop on Performance Metrics for Intelligent Systems* (2008).



Lorenzo Jamone received his M.S. degree in Computer Engineering from the University of Genoa, and his Ph.D. degree in Humanoid Technologies from the University of Genoa and the IIT (Italian Institute of Technology), in 2006 and 2010, respectively. He was a Research Fellow at the RBCS department of the IIT, in 2010, and he is now Postdoctoral Researcher at the Takanishi Laboratory in Waseda University, Tokyo.

His research interests include humanoid robotics, autonomous online learning, machine learning, motor control, force and tactile sensing.

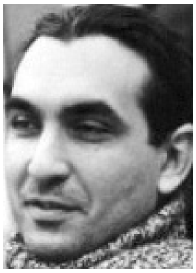


Lorenzo Natale received his degree in Electronic Engineering (with honors) in 2000 and Ph.D. in Robotics in 2004 from the University of Genoa. During his MSc and Ph.D. he worked in LIRA-Lab (Laboratory for Integrated Advanced Robotics), at the University of Genoa. Between 2005 and 2006 he was Postdoctoral researcher at the MIT Computer Science and Artificial Intelligence Laboratory, in the Humanoid Robotics Group.

He is now team leader at the IIT. His research focuses on developmental robotics, sensorimotor learning and perception in artificial and biological systems.

Francesco Nori received his D.Eng. degree (highest honors) from the University of Padova in 2002. During 2002 he was visiting student in the UCLA Vision Lab, University of California Los Angeles. He received his Ph.D. in Control and Dynamical Systems from the University of Padova in 2005. In 2006, he started his PostDoc at the LIRA-Lab, University of Genoa. In 2007 he moved to the IIT as a team leader.

His research interests include motor control, humanoid robotics, computational vision, human motion tracking.



Giorgio Metta is senior scientist at the IIT and assistant professor at the University of Genoa. He holds a MS with honors (in 1994) and PhD (in 2000) in electronic engineering both from the University of Genoa. From 2001 to 2002 he was Postdoctoral associate at the MIT AI-Lab where he worked on various humanoid robotic platforms. He is assistant professor at the University of Genoa since 2005 and with IIT since 2006.

He is author of more than 150 publications. His research activities are in the fields of biologically motivated and humanoid robotics and in particular in developing life-long developing bio-inspired artificial systems.

Giulio Sandini is Director of Research at the IIT (department of Robotics, Brain and Cognitive Sciences) and full professor of bioengineering at the University of Genoa. He graduated in Electronic Engineering at the University of Genoa in 1976 and he was research fellow and assistant professor at the Scuola Normale Superiore in Pisa until 1984. After his return to Genoa in 1984 as associate professor, in 1990 he founded the LIRA-Lab.

He is with IIT since 2006. He is author of more than 300 publications and five international patents. His research activities are in the fields of Biological and Artificial Vision, Computational and Cognitive Neuroscience and Robotics.