PALADYN Journal of Behavioral Robotics

VERSITA

Research Article · DOI: 10.2478/s13230-013-0102-z JBR · 3(3) · 2012 · 113-127

Incremental Development of Multiple Tool Models for Robotic Reaching Through Autonomous Exploration

Lorenzo Jamone[1*], **Bruno Damas**[2,3†], **Nobotsuna Endo**[1‡], **José Santos-Victor**[2§], **Atsuo Takanishi**[1,4¶]

1 Faculty of Science and Engineering, Waseda University, Tokyo, Japan

2 Instituto de Sistemas e Robótica, Instituto Superior Técnico, Lisboa, Portugal

3 Escola Superior de Tecnologia de Setúbal, Setúbal, Portugal

4 Humanoid Robotics Institute, Waseda University, Tokyo, Japan

Received 13-12-2012

Accepted 27-03-2013

Abstract

Autonomy and flexibility are two major requirements for modern robots. In particular, humanoid robots should learn new skills incrementally through autonomous exploration, and adapt to different contexts. In this paper we consider the problem of learning forward models for task space control under dynamically varying kinematic contexts: the robot learns incrementally and autonomously its forward kinematics under different contexts, represented by the inclusion of different tools, and exploits the learned model to realize reaching with those tools. We model the forward kinematics as a multi-valued function, in which different outputs for the same input query are related to different tools (i.e. contexts). The model is estimated using IMLE, a recent online learning algorithm for multi-valued regression, and used for control. No information is given about the tool changes, nor any assumption is made about the tool kinematics. Results are provided both in simulation and with a full-body humanoid. In the latter case we show how the robot successfully performs reaching using a flexible tool, a clear example of complex kinematics.

Keywords

motor learning and adaptation· humanoid robots · reaching with tools · developmental robotics · continuous online learning

1. Introduction

The future of humanoid robots is to become efficient helpers for humans, both in the execution of everyday tasks and in the accomplishment of tedious and dangerous works. Driven by this vision, researchers have been challenged to design more and more complex robots, that show an increasing number of degrees of freedom and sensors [1, 2]; these robots should be able to cope with the unstructured environment in which humans daily live and act. In particular, it would be desirable that robot behaviors become autonomous (not requiring the supervision of a human expert) and flexible (applicable to different situations and contexts).

However, as robots become more complex, building the analytical models needed for robot control is turning more and more difficult and time-consuming. Moreover, the lack of knowledge of certain hard to measure physical parameters (*e.g.* friction) and the existence of highly non-linear physical interactions, such as actuator nonlinearities, soft or deformable parts and unmodeled mass distributions, makes it infeasible to obtain adequate and accurate models for such kind of systems [3]; as a consequence, resorting to modern machine learning techniques is becoming a more and more popular way to provide these complex robots with the necessary representation capability (see [4] for a recent survey).

Among these techniques, there are a few that have been very successful at learning the robot kinematics or dynamics: Gaussian Processes Regression achieves a state of the art performance with respect to estimation generalization error [5], while the Locally Weighted Projection Regression (LWPR) algorithm [6], an online non-linear function approximation algorithm, has been widely used for robotic learning problems due to its excellent memory requirements and low computational complexity.

Many robotic tasks, however, involve handling and manipulation of different objects, which makes the environment and the mappings to be learned non-stationary. The kinematics mapping from robot joint angles to end-effector position, for instance, changes whenever different tools are used; another classical example is the change in the robot dynamics due to the variation of the load of the end-effector.

This is known as learning and control under a varying context, where an unobserved context variable changes the map that has to be learned and used for the control. Such context can generally be a discrete variable, corresponding to the case where only a finite, albeit unknown, number of different contexts exist, or continuous, indicating a smooth change on the mapping to learn. The most straightforward answer to this problem is to introduce some form of adaptation in the learning algorithms, making them forget past experience through the use of

*E-mail: lorejam@liralab.it
†E-mail: bdamas@isr.ist.utl.pt
‡E-mail: n-endo@takanishi.mech.waseda.ac.jp
§E-mail: jasv@isr.ist.utl.pt
¶E-mail: contact@takanishi.mech.waseda.ac.jp

some kind of forgetting factors mechanism. Of course, it is terribly inefficient to relearn the complete mapping every time the context changes, especially when there is an effective chance that a previously learned context may be presented again to the robot. Another approach to this problem, for the discrete case, is to keep a set of models that describe the robot model for each different context. Three critical issues arise when learning multiple models for robot control. The first issue is i) how to identify the correct number of models to use, without any problem specific a-priori information. The other two issues are ii) how to estimate the current context, given that the correct number of models to use is known, and iii) how to use such estimation for either controlling the robot or further training the models. In Section 2 we describe how previous works tackled these issues.
In this paper we propose a different approach, by directly modeling the map to be learned as an unknown multi-valued function, a multimap that can assign different solutions for the same query input point: in such scheme, each branch of the multimap represents the relation from an input vector to an output vector, for a specific unknown context. This multi-valued function is learned from sensory data using the Infinite Mixture of Linear Experts (IMLE) algorithm [7], a recent incremental learning algorithm that is particularly suited for these kind of multi-valued functions. This algorithm describes the map to be learned as a collection of local linear models that can coexist in similar input locations, thus potentially producing multi-valued estimates for the output corresponding to a particular input query point: the most important mechanisms of this algorithm are detailed in Section 3. Using a single IMLE multi-valued model for the discrete context estimation problem has some tremendous advantages over the previous approaches to discrete varying context and control. On one hand, there is no need to maintain a bank of single-valued function approximation models, since IMLE produces a discrete set of solutions for each input query point; the number and values for this set of solutions depend on the specific input query location and the information gathered so far by the algorithm. This also avoids the need to define or estimate in advance the number of single-valued models to use. Secondly, the IMLE training process, based on the EM algorithm, automatically and transparently assigns responsibilities to each of the local models for each training point, with no need to explicitly maintain an estimate for the hidden context variable. This even allows for the existence of a different number of contexts in different locations of the input space. Choosing an appropriate control action is also very simple using IMLE: assuming some form of continuity and smoothness, a particular solution, for a given query point, can be picked by simply choosing the predicted solution closest to the previous output point: using IMLE for robot control is described in Section 4.
To evaluate the performance of the IMLE algorithm under the discrete varying context situation we used it to learn the kinematics of two different humanoid robots, iCub [2] and Kobian [1], while holding different tools. The two robotic platforms are described in Section 5. The learned kinematic models were employed to control the robots reaching movements switching dynamically between the different tools. No information whatsoever was conveyed to the algorithm whenever the tool was changed; moreover, no assumptions were made about the kinematic properties of the tools being used. The results are shown in Section 6 and Section 7: to our knowledge, this is the most general and efficient approach to learning and controlling under discrete varying contexts.

2. Related work

Humans develop the ability to use tools in a meaningful way following a long and complex process that begins at birth. Guerin et al. [8] provide a comprehensive discussion about the mechanisms undelying the development of tool use abilities in human newborns, from the initial sensorimotor learning to the emergence of planning capabilities.
In the context of this paper, we focus on how humans (and robots) learn internal models for motor control, and how they adapt these models to different contexts: an example of different kinematic contexts can be the use of different tools for reaching.
Indeed, adult humans can skillfully perform operations with different tools, exploiting internal models that are i) learned autonomously from motor experience and ii) recalled automatically during tool usage; in other words, humans can dynamically switch between different learned models based on the actual context. According to the literature, subjects involved in motor learning and motor adaptation tasks may take many movements to achieve a good performance in a novel context characterized by a kinematic [9] or dynamic [10] perturbation. However, when the perturbation is removed, re-adaptation to the original context (i.e. a context that has been experienced, and therefore learned, for a lifetime) is often very rapid. This suggests that learning to operate in a new context (e.g. with a new tool) may represent the creation of a new model, whereas re-adaptation represents the switching back to a previously learned model. Further evidence is provided by specific work on re-adaptation, where on repeated presentation of a kinematic [11] or dynamic [12] perturbation subjects adapt increasingly rapidly, suggesting that a model of the perturbated context is incrementally build through motor experience, and it is recalled when the context appears.
Moreover, recent neuroscience studies [13] suggest that an internal representation of the body is stored in the human brain (i.e. a ***body schema***); this schema can be extended to include noncorporeal objects that have a systematic relation to the body itself, such as manipulated tools (i.e. an ***extended body schema***). Indeed, while holding a tool, humans tend to remap parts of the space allocated as "far" to "near" [14]. These inclusions are generally regarded as temporary and related to the actual context.
Studies on monkeys have analyzed the neural mechanisms underlying this behavior. Neurons in the monkey's caudal postcentral gyrus respond to somatosensory and visual stimuli arising from the hands. If the monkey retrieves food with the hand the visual receptive fields of these neurons are limited to the hand, but when they use a tool the visual receptive fields expand to include both the hand and the tool, and this modification is strictly limited to the time of tool usage [15]. It is reasonable to assume that similar phenomena take place in the human brain as well.
As discussed in the introduction, a possible way to achieve similar adaptation capabilities in robots is to resort to modular control approaches based on learned models. A number of solutions has been proposed in the recent years. The early work on adaptive control of Narendra [16] considers that an appropriate number of models is given a-priori, already trained and exhibiting good performance within each context. The MOSAIC architecture [17, 18], on the other hand, assumes that some perceptual cues are available that can guide a correct context estimation in the early learning process, that can in turn successfully assign the perceived data points to a predefined number of models. This, however, can be a quite optimistic assumption, as not only the number of models must be known beforehand but also a very domain specific information must be gathered to build the functions relating perceptual cues to specific contexts — precisely what is to be avoided when using a learning architecture. While the early results using the MOSAIC model are limited to a very simple system consisting of an object moving along a single direction axis, recently an extension has been proposed to control a real humanoid robot [19]. However, even the extended MOSAIC suffers from the main limitation of requiring an a-priori definition of the number of models, which in the work described

in [19] is obtained in simulation before performing the real robot experiments. Finally, the approach presented in [20] claims the ability to deal with continuous varying contexts, although the method only holds under changes in the mass of the object being manipulated — it could not be applied, for instance, when varying a robot link length while trying to learn its forward kinematics. Their assumption of an explicit latent context variable also brings some problems when the current context needs to be inferred for training purposes: in the continuous case they need to resort to two models previously trained using context labeled data before they are able to generalize to unseen contexts, while in the discrete case a bootstrap, based on a EM procedure over a batch of unlabeled data points, is required when no trained models exist yet — this however, goes against the online, incremental philosophy of LWPR [6], the function approximation algorithm used in the corresponding simulations.

Several works focus more specifically on learning a model of the robot body schema (see [21] for a recent survey); among them, a few also deal with the inclusion of tools into the learned model. Taking direct inspiration from the studies of Iriki about neural adaptation in monkeys [15], the work in [22] proposes a visual attention module that can extend the robot body representation through Hebbian learning when a tool is being used; simulation results are provided. In [23] a simple 2-joints planar manipulator is controlled using an analytical model of the Jacobian, and when a tool is added to the kinematic chain the corresponding Jacobian is obtained through multiplication of the analytical Jacobian by a linear constant matrix, which is learned exploiting the temporal integration of visual and tactile information during motor exploration. A more comprehensive system is presented in [24], where recurrent neural networks are used to identify the tool currently being used by the robot, based on previous trained data, and allow to generalize to unseen tools. This work, however, is based on a human generated set of training movements of the robot arm and can only produce movements that are replications of the original ones used for training, thus not addressing the more general problem of full kinematic learning with different tools. Another approach is proposed in [25], where a recurrent neural network parametrized with the length of the tool is used to estimate the inverse kinematics of a humanoid robot. However, the length of the tool must be known in advance to train and query the neural network. Additionally, training is done using circular trajectories in a fixed plane: this procedure learns a subspace of much lower dimensionality than the joint space dimension being used. Another big limitation of these works is that assumptions are made about the kinematics of the tool, and they can account only for rigid transformations (e.g. they cannot cope with flexible or deformable tools).

3. The IMLE Algorithm

The IMLE algorithm [7] is a probabilistic algorithm that uses a generalized expectation-maximization (EM) procedure to update its parameters, fitting an infinite mixture of linear experts to an online stream of training data (z_i, x_i), where $z_i \in \mathbb{R}^d$ denotes an input point and $x_i \in \mathbb{R}^D$ denotes the corresponding output. Its only assumptions about the training data nature is that it can be approximated by a mixture of local linear models: this naturally allows for multi-valued function learning, as the different branches of the multimap can be approximated by different linear models sharing the same input region. It starts with the following probabilistic generative model,

$$p(x_i|z_i, w_{ij}; \Theta) \sim \mathcal{N}\left(\mu_j + \Lambda_j(z_i - \nu_j), \Psi_j\right) , \quad (1)$$

$$p(z_i|w_{ij}; \Theta) \sim \mathcal{N}\left(\nu_j, \Sigma_j\right) , \quad (2)$$

where the mean ν_j and covariance Σ_j define each expert j active input region. Parameters μ_j and Λ_j define the linear relation from input to output for a particular expert: μ_j is the mean for the output marginal distribution of expert j, while Λ_j is a matrix containing the regression coefficients; Ψ_j is a diagonal matrix that represents the uncorrelated noise at each of the output dimensions. The unobserved, latent variable w_{ij} assigns sample points to experts, while the parameter vector Θ gathers all the parameters to be learned. The model starting point is in essence similar to the one presented in [26]; however, some priors over the mixture parameters are additionally defined to perform some regularization and to enforce the principle of localized learning, thus avoiding the interference of experts across different regions of the input space. More details on the full probabilistic model are available in [7].

Training of the model is done using an online EM algorithm: in the expectation step (E-Step) responsibilities are assigned to experts for a new point (z_i, x_i), according to:

$$h_{ij} \equiv E[w_{ij}|x_i, z_i; \hat{\Theta}] = p(w_{ij}|x_i, z_i; \hat{\Theta}) , \quad (3)$$

where $\hat{\Theta}$ is the most recent estimate for the mixture parameters being learned. Maximization step (M-Step) then updates the parameters in $\hat{\Theta}$ according to the responsibilities h_{ij} previously obtained. Based on a model for outlier points, the mixture can grow by automatically adding new experts whenever the perceived data points are not well explained by the mixture. Once again, please refer to the original paper for details.

The E-Step above avoids the typical interference between experts for the multi-valued function estimation case, since it assigns responsibilities based on both the input and output part of the training point. LWPR, for instance, although sharing the mixture of localized linear models concept with IMLE, adapts the distance metrics of each receptive field using a procedure based on the minimization of the (single-valued) prediction error; this, together with an attribution of responsibilities based on the input part of a data point only, makes the coexistence of linear models in the same input region infeasible, as required for multi-valued learning.

Given a current set of mixture parameters, a single-valued prediction algorithm will typically mixture the individual linear models predictions according to some weighted average scheme, using weights $w_j^x(z_q)$ that depend on how strong each model is activated given only the input query point z_q. This is of course unacceptable for multi-valued prediction. The IMLE algorithm tries to find, for a given input query z_q, a set of estimated predictions $\hat{x}_k$ by grouping and clustering the linear models point estimates into a minimal set of predictions. It uses a probabilistic model that relates linear models point predictions $\hat{x}_j$ to the unknown set of true multi-valued predictions $\bar{x}_k$, also taking into account the weights $w_j^x(z_q)$ and the estimation variances R_j provided by each linear model. This process then has to tackle two major questions, namely how to group linear models point estimates into a set of N_{pred} coherent predictions and how to choose N_{pred}, the appropriate number of such predictions. The clustering problem is solved using another EM procedure, by assuming some latent variables s_{jk} exist that assign models point estimates to unknown predictions — we will call it EM_{pred} to distinguish it from the EM procedure described before for mixture training. After the EM_{pred} procedure is carried through, a statistical hypothesis test is performed, to assess the fit of the resulting set of multi-valued predictions: if the test rejects the goodness of fit hypothesis then it is assumed that the number of predictions N_{pred} is insufficient. For a query point z_q the IMLE algorithm starts with the single-valued prediction: if the test finds evidence to reject the hypothesis that the models point estimates are distributed according to a single-valued prediction, the value of N_{pred} is increased to 2 and the EM_{pred} clustering procedure is carried on; if the goodness of fit hypothesis is again rejected the

number of predictions N_{pred} is again increased, until the test fails to reject the hypothesis and a final set of N_{pred} multi-valued predictions is obtained.
This clustering procedure is a distinctive feature of IMLE when compared, for instance, to the work described in [27], that can also deal with multi-valued prediction: while IMLE, for each query, can provide a minimal set of coherent predictions, the latter algorithm is only able to stochastically sample a prediction from the set of models. As noted by their authors, this can lead to unwanted rapid changes of predicted context, that can have disastrous consequences in the control phase.
As a final remark, note that IMLE features a very low computational complexity: for every new training point presented the learning algorithm is $\mathcal{O}(Md(d+D))$, *i.e.*, linear in the number of active experts M and output dimensions D and quadratic in the number of input dimensions d, thus making it directly comparable to the state-of-the-art LWPR in terms of computational complexity per training point[1]. Like LWPR, this complexity can be made linear in d if the input distance metrics Σ_j are constrained to be diagonal. It also has been shown in [7] that IMLE can outperform LWPR in terms of prediction error for single-valued problems, while keeping in general a lower number of allocated linear models. For prediction, IMLE computational time grows linearly with N_{pred}, the number of multi-valued solutions found.

4. Task Space Control

Reaching is a particular case of the more general problem of task space (or operational space) control. Task space is defined as the space in which the robot tasks are described (e.g. the 3D Cartesian space), as opposed to the joints space (or motor space). In the case of reaching, we want to control the position of the robot end-effector in the task space: this end-effector can be represented by either the robot hand or the tip of a manipulated tool. Since the robot needs to be controlled in joints space, an inverse model is required which maps the desired task space behavior into the appropriate joints space trajectories (i.e. motor commands). When the joints space dimension is larger than the task space dimension an infinity of solutions exists for the model inversion: in this case, we say that the robot is redundant with respect to the specified task. To control the end-effector position in task space we follow the approach originally proposed in [28], where due to the redundancy of the system two goals can be simultaneously achieved through null-space projection: a main goal in the task space (i.e. positioning of the end-effector) and a secondary goal in joints space (i.e. keeping the joints as far as possible from the physical limits). The secondary goal is projected in the null-space of the main goal; this results in motor velocities $\dot{\mathbf{q}}$ computed as follows:

$$\dot{\mathbf{q}} = K_m J^{\dagger}(\mathbf{q})\dot{\mathbf{x}}^d + K_s(I - J^{\dagger}(\mathbf{q})J(\mathbf{q}))\,\dot{\mathbf{q}}^d\ , \tag{4}$$

where the Jacobian matrix $J(\mathbf{q})$ maps from motor velocities to task velocities, $J^{\dagger}(\mathbf{q})$ is its Moore-Penrose generalized inverse, $(I - J^{\dagger}(\mathbf{q})J(\mathbf{q}))$ is a null-space projector, $\dot{\mathbf{x}}^d$ is the desired task space velocity for the main goal and $\dot{\mathbf{q}}^d$ is the desired joint space velocity for the secondary goal. At every control step these desired velocities are chosen as

$$\dot{\mathbf{x}}^d = \Delta\mathbf{x} = \mathbf{x}_d - \mathbf{x} \qquad \text{and} \qquad \dot{\mathbf{q}}^d = -\nabla M(\mathbf{q})\ , \tag{5}$$

where $\mathbf{x}_d$ and $\mathbf{x}$ are the desired and actual task space positions and $\nabla M(\mathbf{q})$ is the gradient of $M(\mathbf{q})$, the function we want to minimize as a secondary goal. K_m and K_s are positive definite diagonal gain matrices respectively for the main and secondary goals. Since our secondary goal is to keep the joints as far as possible from their limits we chose $M(\mathbf{q})$ as in [28]:

$$M(\mathbf{q}) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\mathbf{q}_i - \mathbf{a}_i}{\mathbf{a}_i - \mathbf{q}_i^{max}}\right)^2\ , \tag{6}$$

$$\mathbf{a}_i = \frac{\mathbf{q}_i^{max} + \mathbf{q}_i^{min}}{2}\ , \tag{7}$$

where N is the overall number of joints and $\mathbf{q}^{min}$ and $\mathbf{q}^{max}$ are lower and upper joints limits. At every control step we check whether the system is close to singularities by computing the smaller singular value of the Jacobian through singular value decomposition (SVD). If the smaller singular value δ_m is lower than a predefined threshold δ_T we rely on the damped least squares solution [29]. In this case the pseudoinverse is computed as follows:

$$J^{\dagger}(\mathbf{q}) = J^T(\mathbf{q})(J(\mathbf{q})J^T(\mathbf{q}) + \lambda^2 I)^{-1}\ , \tag{8}$$

with

$$\lambda^2 = \left[1 - \left(\frac{\delta_m}{\delta_T}\right)^2\right]\cdot\lambda^2_{MAX}\ . \tag{9}$$

The Jacobian $J(\mathbf{q})$ is obtained estimating the local slope, for an input query point $\mathbf{q}$, of the current learned map from joint to task space, $\mathbf{x} = \hat{f}(\mathbf{q})$. This solution has been proposed in [30], using LWPR for the map estimation. Of course, when using IMLE to provide the Jacobian estimate, we must first choose one of the possible multiple solutions provided by the algorithm for the same query point $\mathbf{q}$. A fairly simple and efficient solution is to pick the prediction closest to the output point $\mathbf{x}$ acquired on the previous control step. This procedure is implicitly assuming that the context does not change very frequently (i.e. at each control step), which seems a reasonable assumption.

5. Robotic platforms

The proposed learning strategy is very general and our goal is to apply it to different humanoid robots, namely iCub [2] and Kobian [1] (depicted in Figure 1). The experiments reported in Section 6 and Section 7 have been realized using the dynamic simulator of the iCub [31] and the real robot Kobian, respectively. In the following subsections we describe the two platforms in detail, describing the different joints and task spaces and the different tools used for reaching. All the software has been realized using YARP [32] using a modular approach that allowed us to control both iCub and Kobian robots with minor modifications in the code.

5.1. The iCub

The iCub is a humanoid robot for research in embodied cognition, developed in the context of the EU project RobotCub and subsequently adopted by more than 20 laboratories worldwide. It has 53 motors that move the head, arms and hands, waist, and legs. It can see and

[1] This is a consequence of using the Sherman-Morrison formula to perform the matrix inversions involved in the training procedure.

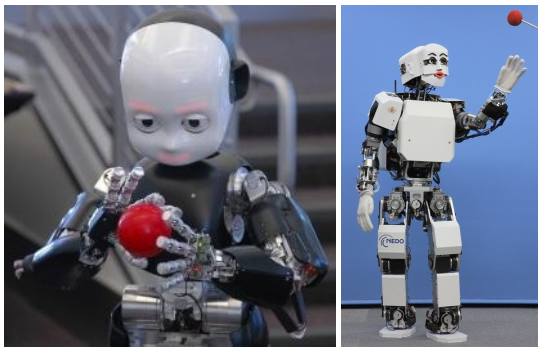

Figure 1. The humanoid robots iCub (on the left) and Kobian (on the right).

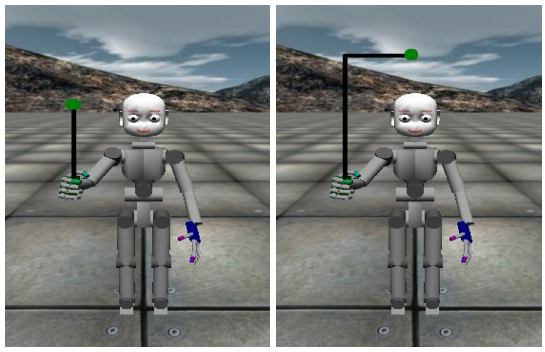

Figure 2. Snapshots of the iCub Simulator grabbing the two different tools used in the experiments: on the left, the 28 cm stick tool, on the right, the 48x30 cm L-shaped tool.

hear, it has the sense of proprioception (body configuration) and movement (using accelerometers and gyroscopes). In the following experiments we use the iCub simulator [31], a realistic software that uses ODE (Open Dynamic Engine) for simulating rigid bodies and collision detection algorithms to compute the physical interaction with objects. Snapshots of the simulated iCub holding the different tools are displayed in Figure 2. The right arm and the waist of the robot are actuated to control the end-effector position in the 3D Cartesian space, using the task space controller described in Section 4. The end-effector can be either the robot hand or the tip of a tool: the two tools used in the experiments (a 28 cm long stick tool and a 48×30 cm L-shaped tool) are displayed in Figure 2. The joint space vector $\mathbf{q}$ and task space vector $\mathbf{x}$ are defined as follows:

- $\mathbf{q} = [\theta_{sp}\ \theta_{sy}\ \theta_{sr}\ \theta_e\ \theta_{wy}\ \theta_{wr}\ \theta_{wp}]^T \in \mathbb{R}^7$
- $\mathbf{x} = [x_p\ y_p\ z_p]^T \in \mathbb{R}^3$

where θ_{sp}, θ_{sy}, θ_{sr} are the shoulder pitch, yaw and roll rotations (elevation/depression, adduction/abduction and rotation of the arm), θ_e is the elbow flexion/extension, θ_{wy}, θ_{wr}, θ_{wp} are the waist yaw, roll and pitch rotations (rotation, adduction/abduction, elevation/depression of the trunk), and x_p, y_p, z_p are the three cartesian coordinates describing the position of the end-effector (it can be either the hand of the tip of a tool) with respect to a fixed reference frame placed on the ground, in the middle between the robot feet. The robot joints limits are defined in Table 1.

Table 1. Joints limits of the iCub robot simulator.

	arm				waist		
$\mathbf{q}^{min}$	$-80°$	$0°$	$0°$	$20°$	$-30°$	$-30°$	$-10°$
$\mathbf{q}^{max}$	$0°$	$80°$	$80°$	$80°$	$30°$	$30°$	$30°$

5.2. Kobian

Kobian is a 48-DOFs full humanoid robot, that has been designed to integrate the bipedal walking skill of Wabian [33] to the emotion expression capabilities of the human-like head robot WE-4 [34], as described in [35]. Kobian can express different emotions (e.g. happiness, sadness, fear, anger) with face and whole-body movements [1], also during locomotion [36]. The robot size is similar to that of an average

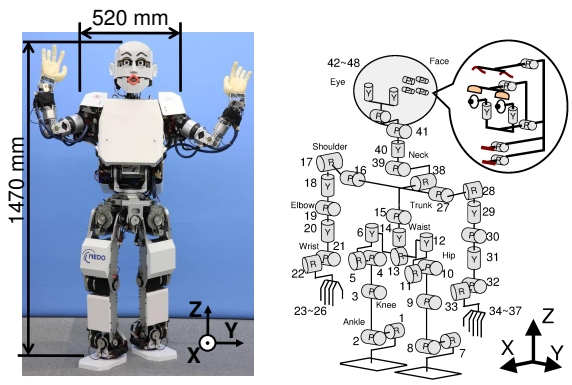

Figure 3. Kobian humanoid robot. On the left, Kobian expressing surprise. On the right, description of the robot 48 DOFs.

Japanese woman (see left image in Figure 3) and the overall weight is 62 kg. The degress of freedom of the robot are distributed as follows: 12 in the two legs, 3 in the waist, 14 in the two arms, 8 in the two hands, 4 in the neck and 7 in the head (see right image in Figure 3).
All the joints are driven by DC motors with encoders activated by electric motor drivers (Tokushu Denso Co., Ltd.); counter readings of the encoders (i.e. measure of joint angular positions) and output of the velocity references to the motor drivers (i.e. motor commands) are done by a PC (Pentium M 1.8 GHz, QNX Neutrino 6.3.0 operative system) embedded in the robot back through the I/O boards (HRP interface boards of ZUCO, Co., Ltd.). This PC is interfaced through Ethernet (or wireless) connection to an external laptop (SONY) on which the higher level programs run (i.e. learning, coordinated control). The laptop is a node of a local network of several PCs, that can be exploited to realize distributed computation when multiple behaviors and reasoning processes are executed together. This modular distributed computation is supported by the use of YARP [32]. Two CMOS color cameras (ARTRAY, ARTCAM-022MINI) are embedded in the robot eyeballs, and directly connected to the laptop through an USB connection; the cameras provide images of 640×480 pixels at a frame rate of 30 Hz.
As we want to realize visually guided reaching, we aim at controlling the Kobian arm (in this case, the right arm) and head, in order to operate in the part of the workspace which is both visible and reachable. The

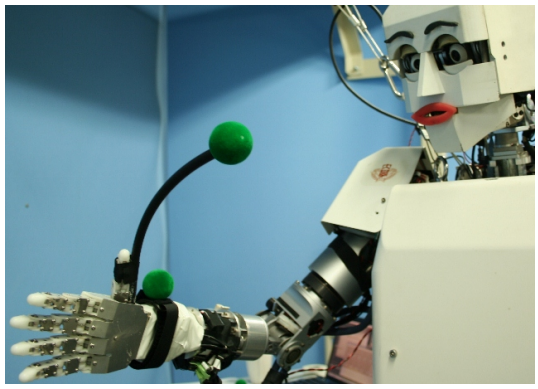

Figure 4. Kobian humanoid robot with the flexible rubber tool. Green balls (i.e. visual markers) are attached to both the wrist and the tip of the tool.

robot end-effector is represented by a visual marker (a green ball in this case) that can be attached either to the wrist (as a marker for the hand) or to the tip of a 35 cm long flexible rubber tube (i.e. the tool); Figure 4 shows the two possible positions of the green ball on the robot. The position of the ball center in the cameras images is computed through visual processing (i.e. color based segmentation) with a precision of about ± 2 pixels (this error is due mainly to slight changes in the illumination, as we perform the experiments in natural environment).
The joints involved in head and arm control are the following:

- $\mathbf{q}_{head} = [\theta_{ny}\ \theta_{np}]^T \in \mathbb{R}^2$
- $\mathbf{q}_{arm} = [\theta_{sp}\ \theta_{sy}\ \theta_{sr}\ \theta_e]^T \in \mathbb{R}^4$

where θ_{ny} and θ_{np} are the neck yaw and pitch rotations (head rotation and elevation/depression, joints 40 and 39 in Figure 3), θ_{sp}, θ_{sy}, θ_{sr} are the shoulder pitch, yaw and roll rotations (elevation/depression, adduction/abduction and rotation of the arm) and θ_e is the elbow flexion/extension (joints 16, 17, 18 and 19 in Figure 3). The corresponding limits are defined in Table 2.

Table 2. Joints limits of the Kobian robot.

	head		arm			
$\mathbf{q}^{min}$	$-40°$	$-10°$	$-70°$	$-30°$	$-15°$	$-90°$
$\mathbf{q}^{max}$	$40°$	$20°$	$-5°$	$-5°$	$15°$	$-5°$

The visual position of the end-effector in camera coordinates $\mathbf{x}_v$ can be defined as follows:

$$\mathbf{x}_v = \begin{bmatrix} u_R \\ v_R \\ u_L - u_R \end{bmatrix} \tag{10}$$

being u_R and v_R the coordinates on the right image plane and u_L and v_L the coordinates on the left image plane. Since $v_L = v_R$ (a perceived target has always the same vertical position on both images), we can univocally describe the 3D visual position with the 2D position in the right image (u_R, v_R) plus the depth information (given by $u_L - u_R$).

5.2.1. Head control and definition of the forward kinematics

When implementing visual based reaching on a humanoid robot, we are interested in locating 3D points in space by moving the head. Here we describe how head control is used to localize the robot end-effector and how we define the forward kinematics in Kobian.
The head joints are controlled to bring the end-effector to the center of the right image (i.e. making u_R and v_R equal to zero); in the rest of paper we will refer to this behavior as "fixation". If the end-effector is visible (i.e. inside the image plane) head joints velocities are generated as follows:

$$\begin{bmatrix} \dot{\mathbf{q}}^0_{head} \\ \dot{\mathbf{q}}^1_{head} \end{bmatrix} = \begin{bmatrix} K_0 & 0 \\ 0 & K_1 \end{bmatrix} \begin{bmatrix} \Delta u_R \\ \Delta v_R \end{bmatrix} \tag{11}$$

where K_0 and K_1 are positive gain constants, and $\Delta u_R = u_R^d - u_R$ is the difference between desired and actual u_R (in this case $u_R^d = 0$); the same applies for Δv_R. If the end-effector 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 the controller given by Equation 11 is activated.
When fixation is achieved (u_R and v_R are equal to zero) the 3D position of the end-effector with respect to the arm base can be encoded using a representation that includes motor variables (similar to the one originally proposed in [37]), as follows:

$$\mathbf{x}_m = \begin{bmatrix} \mathbf{q}^0_{head} \\ \mathbf{q}^1_{head} \\ u_L - u_R \end{bmatrix} \tag{12}$$

According to these definitions, in the experimental results presented in Section 7 we estimate a forward kinematic model in the form $\mathbf{x}_m = f(\mathbf{q}_{arm})$. The Jacobian $J(q)$ derived from this model can be used to control the arm in task space, using the controller described in Section 4. Interestingly, due to the linear relation between $\Delta\mathbf{x}_m$ and $\Delta\mathbf{x}_v$ (given by the K_0 and K_1 constants in Equation 11), the desired task space velocities for the control (i.e. $\Delta\mathbf{x}$ in Equation 5) can be expressed using either $\Delta\mathbf{x}_m$ or $\Delta\mathbf{x}_v$, depending if the head is moving or is fixed. These two possible uses of the learned kinematic model are shown in Section 7.2 and Section 7.3 respectively.

6. Results with iCub simulator

We show here the simulation results obtained using the dynamic simulator of the iCub humanoid robot. Section 6.1 evaluates the online estimation perfomance of IMLE during a motor babbling phase, while in Section 6.2 we use the map learned with IMLE for task space control. Some of the results present a comparison with LWPR, in order to show that:

a) the performance of IMLE and LWPR (a state-of-the-art online algorithm for non-linear regression) are directly comparable for single-valued regression;

b) the multi-valued approach (which is supported by IMLE) allows to efficiently deal with the dynamical inclusion of different tools in the kinematic model, a problem in which the classical single-valued approach fails.

6.1. Model estimation during motor babbling

During the motor babbling phase the robot moves to random reference configurations in the joint space using a low-level joint position control, spanning the whole joints space within the robot limits defined in Table 1. Training points, consisting of joint values $\mathbf{q}$ and respective 3D positions of the end-effector $\mathbf{x}$, are acquired and presented to the learning algorithms. We start the simulation without any tool; after 100,000 training points, the 28 cm stick tool (see left image in Figure 2) is attached to the robot hand, without informing the algorithm of such change in the forward kinematics. After more 100,000 training points the tool is removed and the robot keeps learning its end-effector kinematics for more 100,000 points. During this motor babbling phase, the root mean square error (RMSE) over two independent test sets S_1 and S_2 is calculated: test set S_1, with 3,000 samples, corresponds to the forward kinematics of the robot without using any tool, while S_2 corresponds to the task space positions of the end-effector using the 28 cm stick tool, for the same joint angles of S_1. The obtained results are shown in Figure 5: the average of the RMSE over the three components of the output vector $\mathbf{x}$ is plot as a function of the number of samples used for training. These results show that:

a) for single-valued regression the RMSE performance of IMLE is comparable to LWPR (in these particular experiments it is slightly better, even if the number of allocated linear models is significantly lower for IMLE than LWPR — not shown in the figure);

b) modeling the kinematics as a multi-valued function allows to learn different tool kinematics within a single model;

c) IMLE is an effective algorithm for multi-valued regression and prediction.

Indeed, the estimation performance of IMLE in the case of single-valued regression (during the first part of the motor babbling, before the inclusion of the tool) is in line with the one of LWPR; this can be noticed in the left image of Figure 5, looking at the evolution of the estimation error during the first 100,000 training samples. Then, after motor babbling with the tool is performed, the advantages of multi-valued regression speak for themselves: LWPR, as well as any other single-valued regression algorithm, has to forget the previous kinematics model if it wants to achieve a better performance on the current training/test set; as a consequence, whenever the model switches (tool removal/append), LWPR has to start over the learning, and the RMSE on one of the test sets will increase to allow for a decrease on the one corresponding to the current training data. IMLE, on the other hand, does not need to restart learning after the two different situations are presented to it: as can be seen in the right of Figure 5, after removing the tool (200,000 training points) the RMSE suffers almost no change. In general, after learning a model, the error in the corresponding test set will remain low irrespectively of further training under different kinematic models.

6.2. Task space control experiment

To evaluate the performance of the task space control using the learned kinematics a test movement is executed. A sequence of 16 target end-effector positions is provided to the robot: the end-effector trajectory resulting from the control should draw a cube in the task space, including two diagonals. The end-effector can be either the hand or the tip of a tool.
Figure 6 shows the execution of the test movement with the hand, after motor babbling without any tool (100,000 training samples); the overall position error of the end-effector during the movement is displayed in Figure 7. The same movement is then realized by the robot using the 28 cm stick tool, after additional motor babbling with the tool (100,000 training samples): results are displayed in Figure 8. Then, the motion is executed again without the tool, with results shown in Figure 9. Then, motor babbling is performed again without the tool (100,000 training samples), and the test trajectory is executed controlling either the hand or the tool position (see Figure 10).
To test the online learning performance of IMLE during the control, the test movement is also executed using the stick tool after motor babbling was performed only without the tool: the resulting trajectories are shown in Figure 11.
Lastly, as we expect the advantages of the multi-valued approach to be even more evident when dealing with bigger tools, the test movement is executed with the 48x30 cm L-shaped tool depicted in the right image in Figure 2; the target positions used in the previous test are shifted in space, as the robot reaches in a different workspace by using this tool. Additional motor babbling with the tool is performed (100,000 training points), then the test movement is executed (see Figure 12). Then, more motor babbling without the tool is realized (100,000 training points), and the test movement is executed again (see Figure 13).
Note that, during the experiments, IMLE was allocating about 40 experts to describe the robot forward kinematics, raising to about 130 and 280 with the inclusion of the stick tool and the L-shaped tool, respectively. The number of allocated linear models has a strong influence on the computational burden of the algorithm, and it may also signal some sort of learning overfitting. These are reasonably low numbers, especially considering the dimension of the explored space. As a reference, with the learning parameters that led to the control performance shown in Figure 6, LWPR was creating about 400 local experts only for the robot kinematics (without tools). Trajectories resulting from the control using IMLE (left images in Figures 6, 8, 9, 12 and 13, both images in Figures 10 and 11) look straight and regular, suggesting that the estimation of the Jacobian $J(\mathbf{q})$ (which is obtained computing the local slope of the learned multi-valued kinematic model, $\mathbf{x} = f(\mathbf{q})$) is satisfactory both when controlling the hand and when controlling the tip of a tool. Figure 7 shows how the overall task space position error is canceled during the control of the hand using IMLE. The error is computed as the difference between current and target hand position; the sum of the absolute values of the three components (X, Y and Z) of the error is displayed. Every 20 seconds the target position is changed and the position error raises accordingly: for instance, the error goes to 0.05 before starting the shorter edge of the cube-shaped trajectory, while it goes to 0.25 before starting the diagonal. The particular trajectory of the error, which decreases fast at first and then progressively more slowly, is peculiar of Jacobian based control, and is a further proof of the good quality of the Jacobian estimation.
Results in Figure 11 show that the robot can learn to control a new tool on a specific trajectory without relying on any motor babbling with the tool: the left image displays the trajectory of the tip of the tool during the first iteration of the test movement (after motor babbling was performed without the tool), while in the right image the trajectory on the third iteration is depicted. During one iteration the robot collects about 15,000 training points, even if a big part of them have the same values, as the robot holds static positions for large portions of the movement (i.e. when the task space position error is zero).
As expected from the results in Section 6.1, without including any tool the control performance of IMLE is in line with LWPR (see Figure 6). Then, the superiority of the multi-valued approach is clear when the robot uses different tools (see Figures 8 and 9). This is especially evident when dealing with the bigger L-shaped tool (see Figures 12 and 13): while for IMLE the 100,000 training points collected during the motor babbling were sufficient to obtain a good model of the new tool, allowing to control the system in task space, for LWPR they were not

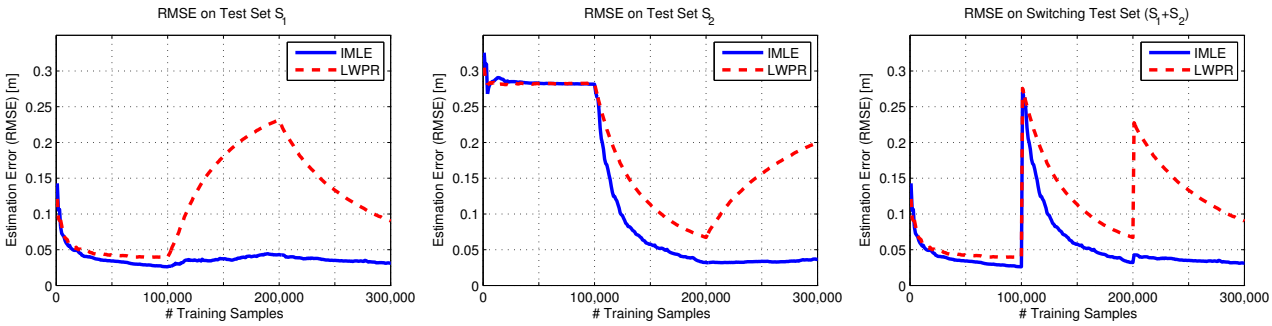

Figure 5. RMSE of the forward kinematic estimation (average of the three components of **x**) during motor babbling with the iCub Simulator, using different test sets. Motor babbling is realized without the tool (until sample 100,000), then with the tool (until sample 200,000) and then again without the tool (until sample 300,000). Left: test set S_1 (no tool). Center: test set S_2 (28 cm stick tool). Right: test set is changed according to the tool used during the training, *i.e.*, first S_1 is used, then S_2 and finally S_1 again.

enough to make the system controllable (see Figure 12). Moreover, the additional motor babbling without the tool does not affect IMLE performances to a big extent, as it does for LWPR (see Figure 13).
Overall, this set of experiments shows that:

a) IMLE provides a good Jacobian estimation, and can therefore be used for task space control, relying, for instance, on the approach that was proposed in [30] using LWPR;

b) new tools can be dynamically included in the learned model either relying on motor babbling or directly during the task space control;

c) considering single-valued regression (Figure 6, control of the hand in task space after motor babbling without any tool) the performance of IMLE in the control is in line with the one of LWPR;

d) if multiple tools are used, the improvements of the multi-valued approach with respect to the single-valued one are dramatic;

e) the computational requirements for IMLE are reasonably low (i.e. a small number of local experts is allocated), allowing the use of IMLE for real-time online learning, estimation and control.

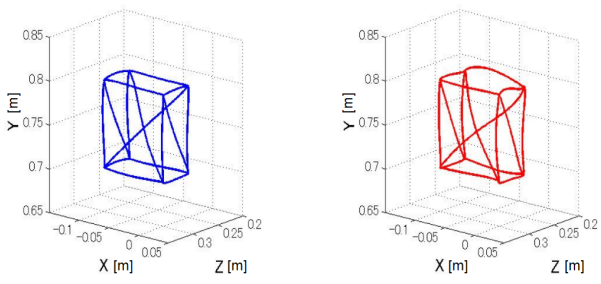

Figure 6. Task space trajectory of the hand during the test movement, after motor babbling without tool was performed. On the left: using IMLE. On the right: using LWPR.

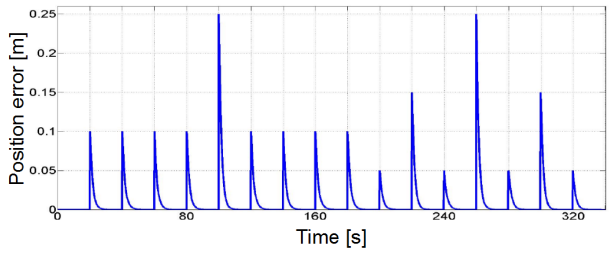

Figure 7. Overall task space position error (sum of the absolute values of the X, Y and Z components) of the hand during the test movement using IMLE. Every 20 seconds the target position is changed and the position error raises accordingly; the particular trajectory of the error reduction during the movements is a further proof of the good estimation of the Jacobian.

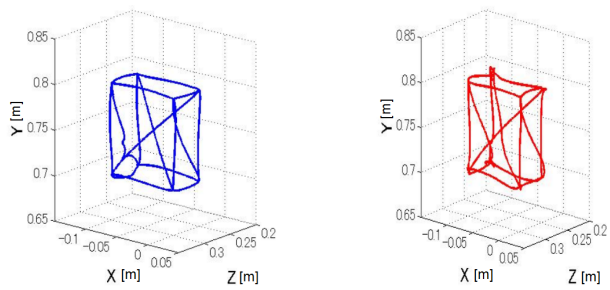

Figure 8. Task space trajectory of the tip of the stick tool during the test movement, after motor babbling was performed first without and then with the tool. On the left: using IMLE. On the right: using LWPR.

7. Results with Kobian robot

We report here the experimental results obtained with Kobian. First we describe how the robot performs motor babbling actuating both the robot head and arm, in order to incrementally estimate the forward kinematics (both with a flexible tool and without) and we present the results

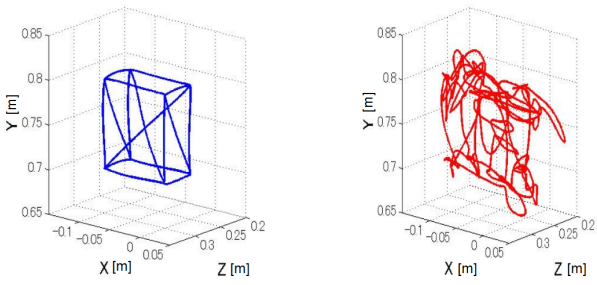

Figure 9. Task space trajectory of the hand during the test movement, after motor babbling was performed first without and then with the tool. On the left: using IMLE. On the right: using LWPR.

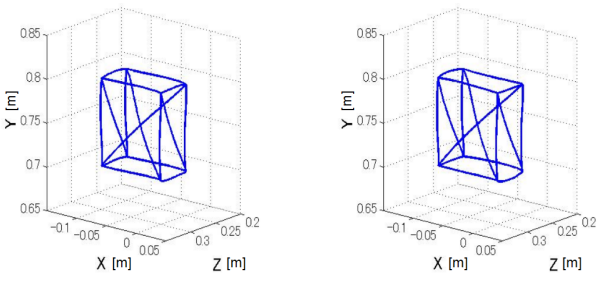

Figure 10. Task space trajectory during the test movement using IMLE, after motor babbling was performed first without tool, then with tool, and then again without tool. On the left: controlling the hand. On the right: controlling the tip of the tool.

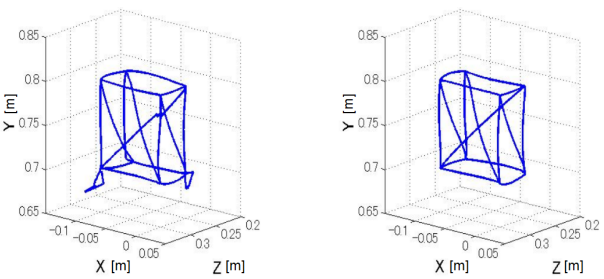

Figure 11. Task space trajectory of the tip of the stick tool during the test movement using IMLE, without previous motor babbling with the tool. On the left: first iteration of the movement. On the right: after 3 iterations of the movement.

of this estimation with respect to given test sets (Section 7.1). Then we show how the learned kinematic model can be used for task space control, realizing two different behaviors: arm-head coordination (Section 7.2) and visually guided reaching (Section 7.3). All the described behaviors are fully autonomous, and learning is realized in an online fashion. The inclusion of the flexible tool (i.e. the context change) is never signalled in any way, neither during motor babbling or task space control. Moreover, even if we present the results in separate Sections, the robot may alternate between these behaviors dynamically depending on the current situation (e.g. performing more motor babbling if the

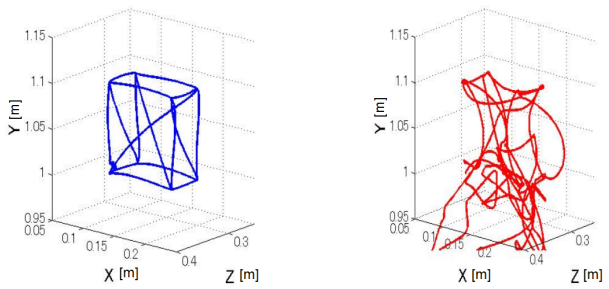

Figure 12. Task space trajectory of the tip of the L-shaped tool during the test movement, after motor babbling was performed first without tool, then with the stick tool, then without tool, and then with the L-shaped tool. On the left: using IMLE. On the right: using LWPR.

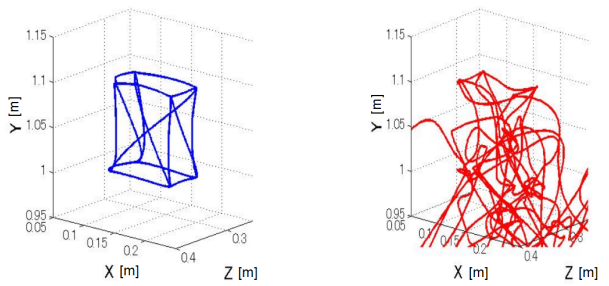

Figure 13. Task space trajectory of the tip of the L-shaped tool during the test movement, after motor babbling was performed first without tool, then with the stick tool, then without tool, then with the L-shaped tool, and then again without tool. On the left: using IMLE. On the right: using LWPR.

outcome of visually guided reaching is not satisfactory). The use of a flexible tool (made of a soft rubber) increases the complexity of the model that has to be learned, with respect to the use of a rigid tool: in fact, the shape of the tool changes depending on the arm configuration, due to the influence of gravity.

7.1. Arm-head motor babbling and model estimation

During the motor babbling Kobian moves the arm to random reference configurations in the joint space using a low-level joint position control, spanning the whole arm joints space within the limits defined in Table 2. After each movement of the arm, the head controller described in Section 5.2.1 is activated, in order to bring the end-effector to the center of the right image plane (i.e. fixation). Training points, consisting of arm joint values $\mathbf{q}_{arm}$ and respective 3D positions of the end-effector $\mathbf{x}_m$, are acquired at the instant of fixation and presented to both IMLE and LWPR learning algorithms in an online, incremental fashion. Due to the head joints limits, not all the possible arm configurations allows fixation. In particular, we further limited the range of $\mathbf{q}^0_{arm}$ to $[-70° \ -60°]$ and the range of $\mathbf{q}^3_{arm}$ to $[-90° \ -40°]$ in order to achieve fixation for all the arm configurations explored during motor babbling, both with and without the tool. This has been done to provide a more fair and precise evaluation of the estimation capabilities of IMLE and LWPR: the robot collects the same number of training points for both contexts (with and without tool), in the same area of the input space (i.e. arm joints space).

During normal operations this modification of the arm joints limits is not required, and indeed the motor babbling behavior also allows to automatically identify the area of the workspace in which 3D points can be both fixated and reached for.
We start the motor babbling without the tool; after 500 training samples have been gathered, we proceed doing motor babbling with the tool, without informing the algorithm of such change in the forward kinematics. After more 500 training points have been collected the tool is removed and the robot keeps learning the forward kinematics without the tool for more 300 points. During this motor babbling phase, the root mean square error (RMSE) over two independent test sets of 100 samples (S_1 and S_2) is calculated: test set S_1 corresponds to the forward kinematics of the robot without the tool, while S_2 corresponds to the forward kinematics with the flexible tool. Figure 14 and Figure 15 show the distribution of arm joints values and head joints values respectively, for both training and testing, with and without the tool. From these two plots is evident that while the input space ($\mathbf{q}_{arm}$) has a similar distribution both with and without the tool, the output space ($\mathbf{x}_m$; only the first two components, $\mathbf{q}^0_{head}$ and $\mathbf{q}^1_{head}$, are displayed in the figure) has very different distributions, requiring the map to be learned ($\mathbf{x}_m = f(\mathbf{q}_{arm})$) to be modeled as a multi-valued function.

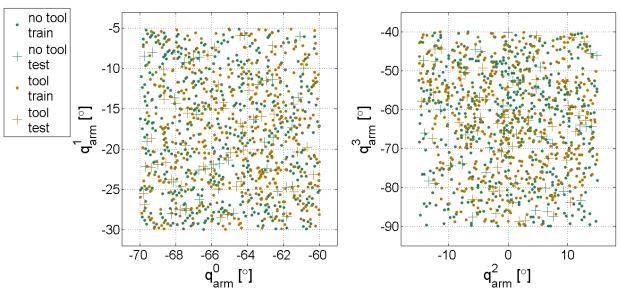

Figure 14. Distribution of arm joints values (i.e. model input) during motor babbling. Green color indicates without tool, brown color with tool. Dots indicates training samples, crosses test samples.

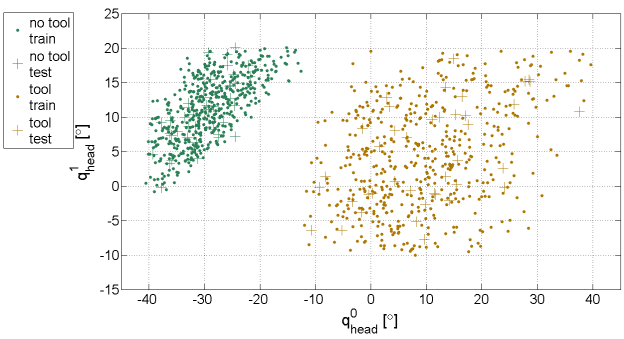

Figure 15. Distribution of head joints values (i.e. model output) during motor babbling. Green color indicates without tool, brown color with tool. Dots indicates training samples, crosses test samples.

The results of the estimation are shown in Figure 16. These results are in line with the simulation results obtained with the iCub Simulator

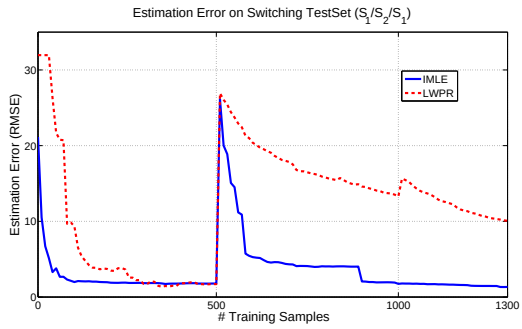

Figure 16. RMSE of the forward kinematic estimation during motor babbling with Kobian, both without and with the tool. Motor babbling is realized without the tool (until sample 500), then with the tool (until sample 1000) and then again without the tool (until sample 1300). The test set is changed according to the tool used during the training, *i.e.*, first S_1 is used, then S_2 and finally S_1 again.

(compare this plot to the right plot in Figure 5). Again, the estimation performance of IMLE in the case of single-valued regression (in the first part of the plot, before the inclusion of the tool at training sample 500) is in line with the one of LWPR. The advantages of multi-valued regression are however evident after motor babbling is performed with the tool as well:

a) faster convergence of the estimation error when a new tool is introduced (from sample 500 to sample 1000);

b) no degradation in the estimation of the original kinematic model (without the tool) when the tool is removed (after sample 1000);

c) further improvement on the original kinematic model during further motor babbling without the tool (from sample 1000 to sample 1300).

The residual RMSE (average of the three components of the output vector) at sample 1000 (after 500 training samples without the tool and 500 with the tool) is 2.79 for test set S_1 (without tool) and 2.74 for test set S_2 (with tool); these values are averages over quantities with different ranges and units (i.e. degrees [°] for joints angles, pixels [px] for the visual depth measured as $u_L - u_R$), that we show for visualization purposes. More precisely, the RMSE is $[0.44°\ \ 0.53°\ \ 7.39px]$ for S_1 and $[0.97°\ \ 0.91°\ \ 6.33px]$ for S_2; normalized to the range of those variables, the RMSE becomes $[0.5\%\ \ 1.8\%\ \ 2.9\%]$ for S_1 and $[1.2\%\ 3\%\ 2.5\%]$ for S_2.

7.2. Arm-head coordination

We describe here a task space control experiment in which Kobian moves both the head and the arm simultaneously exploiting the learned forward kinematics model. A sequence of 16 target visual positions $\mathbf{x}^d = \mathbf{x}^d_v = [u^d_R\ v^d_R\ (u_L - u_R)^d]$ is provided to the robot: u^d_R and v^d_R can be either -50, 50 or 0, while $(u_L - u_R)^d$ is always 0. The target visual position is updated every 2 seconds. When a new target is presented, the robot moves the head to fixate the target (i.e. the head controller described in 5.2.1 is activated). As the head starts to move to fixate the new target (i.e. a point in the image plane), the arm is controlled to follow the head motion. The resulting behavior is a coordinated movement of the head and the arm, during which the end-effector remains in fixation (i.e. u_R and v_R remain equal to zero) with a constant value

of depth (i.e. $u_L - u_R$ remains constant; equal to zero in this specific experiment).
The arm is actuated using the task space controller described in Section 4, where the desired task space velocity (see Equation 5) has been modified as follows:

$$\dot{\mathbf{x}}^d = ff + fb = \dot{\mathbf{x}}_m + k_c \cdot (\mathbf{x}_v^d - \mathbf{x}_v) = \begin{bmatrix} \dot{\mathbf{q}}_{head}^0 \\ \dot{\mathbf{q}}_{head}^1 \\ 0 \end{bmatrix} + k_c \cdot (\mathbf{x}_v^d - \mathbf{x}_v) \quad (13)$$

where $\dot{\mathbf{q}}_{head}^0$ and $\dot{\mathbf{q}}_{head}^0$ are the head joints velocities generated by the head controller (feed-forward term) and $\mathbf{x}_v^d - \mathbf{x}_v$ is the error in the visual position of the end-effector (feed-back term), with $\mathbf{x}_v^d = 0$ in our experiment. The presence of the feed-forward term ensures a quick movement of the arm in response to the head motion, while the feedback term is necessary to cancel the steady-state residual visual error. Figure 17 and Figure 18 show the norm of the end-effector visual position error ($\left\|\mathbf{x}_v^d - \mathbf{x}_v\right\|$) during the execution of the test, without and with the flexible tool respectively. It can be noticed from these plots that the end-effector remains in fixation during the coordinated head-arm movement, as the visual error is typically limited to 1 or 2 pixels. The error raises a bit each time the head starts to move towards a new target, and then it is quickly reduced as soon as the arm motion follows; even during this initial part of the motion, the visual error is still less than $3px$ (when moving without the tool) and $6px$ (when using the tool). In particular, when using the tool, the error raises for a very short time (almost just the duration of one control step, $20ms$) at the beginning of each movement due to small (unmodeled) oscillations of the flexible tool (as the tool is made of a soft rubber).

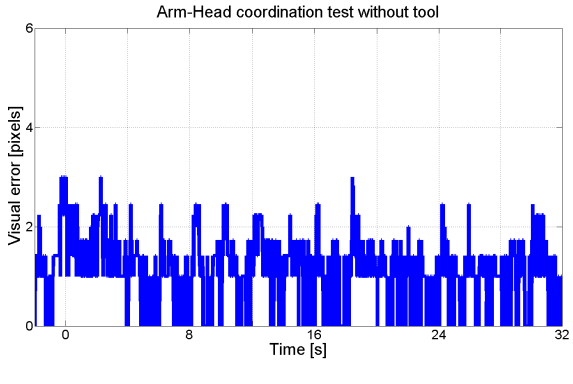

Figure 17. End-effector visual position error with respect to time during the arm-head coordination test performed without the tool. The norm of the visual error ($\left\|\mathbf{x}_v^d - \mathbf{x}_v\right\|$) is shown.

Figure 19 and Figure 20 display the trajectories of the head and arm joints positions ($\mathbf{q}_{head}$ and $\mathbf{q}_{arm}$) respectively, during the execution of the test without the tool. The motion of the head and the arm joints is perfectly synchronized in time, and the joints movements are fast and smooth; similar trajectories (not reported here) have been measured while performing the test with the tool.

7.3. Visually guided reaching

To further test the performance of the task space control using the learned kinematics a visually guided reaching experiment is executed

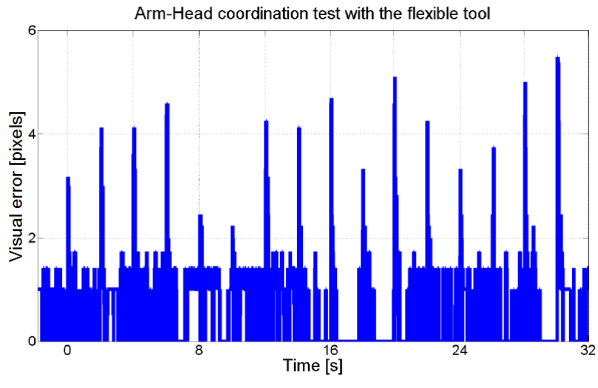

Figure 18. End-effector visual position error with respect to time during the arm-head coordination test performed with the tool. The norm of the visual error ($\left\|\mathbf{x}_v^d - \mathbf{x}_v\right\|$) is shown.

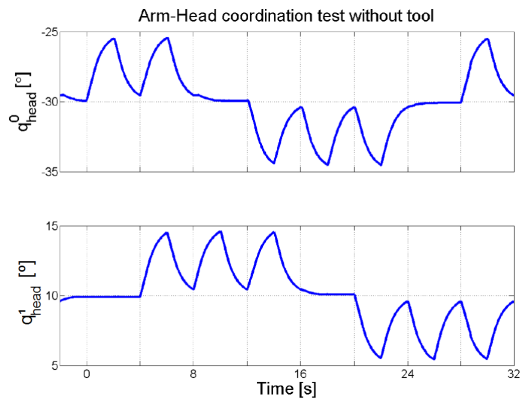

Figure 19. Head joints positions trajectory with respect to time during the arm-head coordination test performed without the tool.

both without and with the flexible tool. The same sequence of visual targets of the arm-head coordination test (see Section 7.2) is presented to the robot, but in this experiment the head is kept in a fixed position, while the arm is controlled to reach for the targets. The end-effector trajectory resulting from the control should draw an asterisk with 8 ends in the right image plane: we refer to this movement as the STAR test. The arm is controlled using the task space controller described in Section 4, with $\dot{\mathbf{x}}^d = \mathbf{x}_v^d - \mathbf{x}_v$, and the head is kept in a fixed position ($\mathbf{q}_{head} = [-30° \ -10°]$ when the test is performed without the tool, $\mathbf{q}_{head} = [15° \ 5°]$ with the tool). The STAR test is executed both without and with the tool, after motor babbling has been performed both without and with the tool, as described in Section 7.1. The visual trajectories of the end-effector in the right image plane are shown in Figure 21 (without the tool) and Figure 22 (with the tool).
Again, these results are in line with the ones obtained in simulation, and presented in Section 6.2. The robot can switch between the two different contexts (without and with the tool) without a significant difference in the performance. Indeed, a perfect estimation of the Jacobian would produce perfectly straight trajectories of the end-effector in the image plane. The experimental trajectories shown in the plots are al-

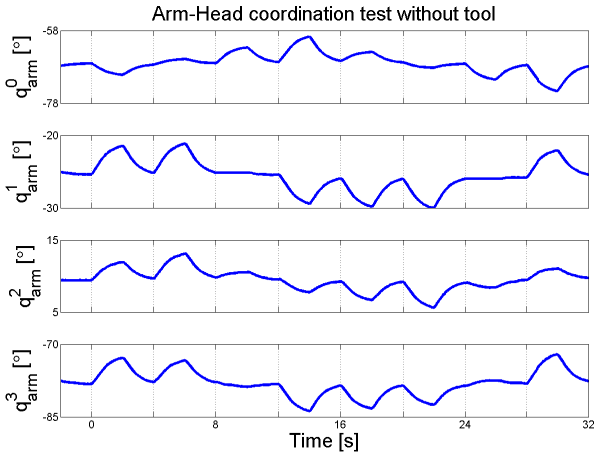

Figure 20. Arm joints positions trajectory with respect to time during the arm-head coordination test performed without the tool.

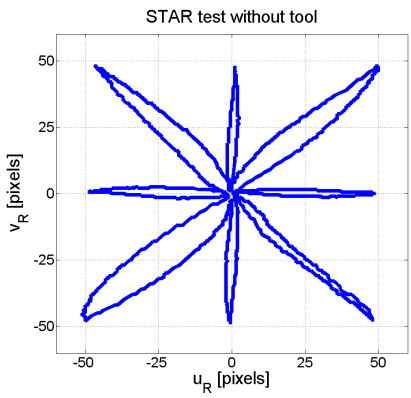

Figure 21. End-effector visual trajectory in the image plane during the STAR test performed without the tool.

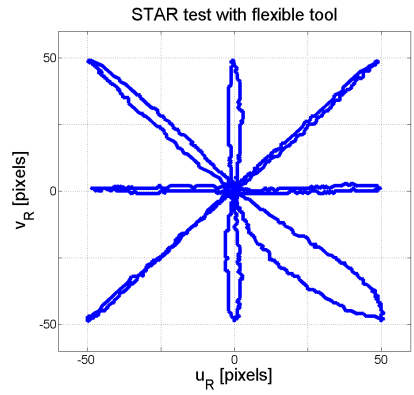

Figure 22. End-effector visual trajectory in the image plane during the STAR test performed with the tool.

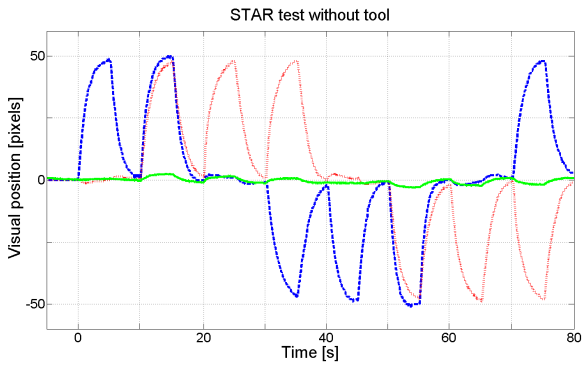

Figure 23. End-effector visual position with respect to time during the STAR test performed without the tool. The three components of the task space vector $\mathbf{x}_v$ are shown: u_R (blue dashed line), v_R (red dotted line) and $u_L - u_R$ (green solid line).

most straight both with and without the tool; slight deviations from the straight trajectory can be seen in a few diagonal directions, with trajectories that are however more than satisfactory. These deviations have to be ascribed to the residual estimation error of the forward kinematics (as reported in Section 7.1, and shown in Figure 16) and to the noise in the visual perception (about ± 2 pixels, as reported in Section 5.2). The trajectories of all the components of $\mathbf{x}_v$ and $\mathbf{q}_{arm}$ with respect to time are depicted in Figure 23 and Figure 24 respectively, regarding the execution of the test without the tool, and in Figure 25 and Figure 26, when using the tool. Figure 23 and Figure 25 show how the u_R (blue dashed line) and v_R (red dotted line) components of the task space vector $\mathbf{x}_v$ reach the desired values (either -50, 50 or 0) and how the $u_L - u_R$ (green solid line) component is effectively controlled to zero (as desired). Figure 24 and Figure 26 show the smooth trajectories of the controlled arm joints $\mathbf{q}_{arm}$; due to the robustness of the Jacobian control, the noise in the visual perception and the residual estimation error of the learned model are not reflected in a noisy motor command generation, neither they cause a jerky motion.

7.3.1. Selection of the best IMLE solution

As described in Section 4, when using IMLE to obtain the Jacobian estimate needed for control, we must choose one of the possible multiple solutions provided by the algorithm for a single query point. The strategy we typically adopt is to pick the prediction closest to the output point acquired on the previous control step. However, this cannot be done in the visually guided reaching experiment described here, as output points ($\mathbf{x}_m = [\mathbf{q}^0_{head}\ \mathbf{q}^1_{head}\ u_L - u_R]$ in this case) must be acquired during fixation of the end-effector (when $\mathbf{x}_v = 0$); such fixation does not occur during the whole trajectory, because the arm moves while the head is kept in a fixed position, but only at the beginning of the test. Therefore, we need to estimate the head motion that the robot would perform to maintain fixation of the end-effector. In particular, we estimate $\hat{\mathbf{x}}_m(i)$ at each control step i using the estimated Jacobian of the previous control step, $J(\mathbf{q}_{arm})(i-1)$, and the measured arm joints displacement from the previous to the current step, $\Delta\mathbf{q}_{arm}(i) = \mathbf{q}_{arm}(i) - \mathbf{q}_{arm}(i-1)$, in the following way:

$$\hat{\mathbf{x}}_m(i) = \hat{\mathbf{x}}_m(i-1) + J(\mathbf{q}_{arm})(i-1) \cdot \Delta\mathbf{q}_{arm}(i), \qquad (14)$$

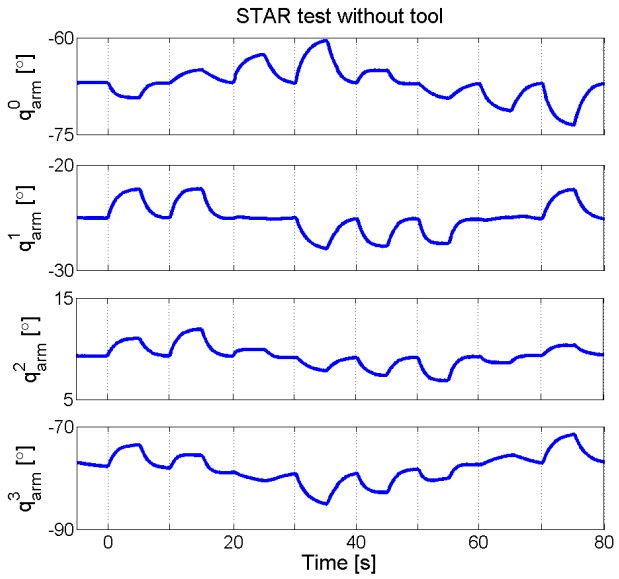

Figure 24. Arm joints positions with respect to time during the STAR test performed without the tool.

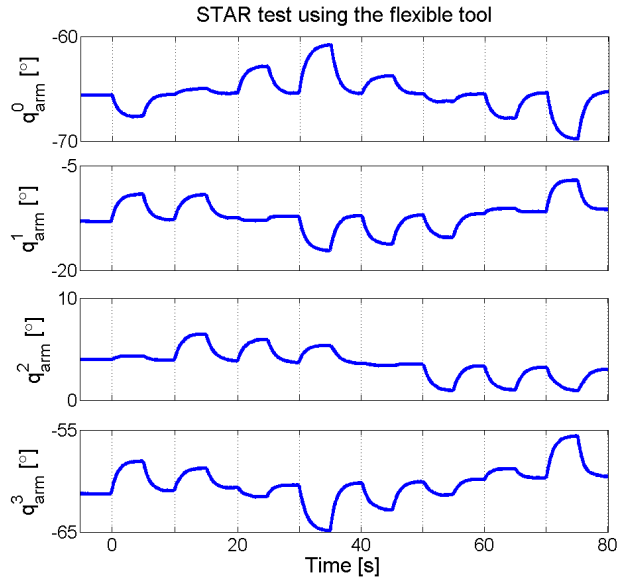

Figure 26. Arm joints positions with respect to time during the STAR test performed with the tool.

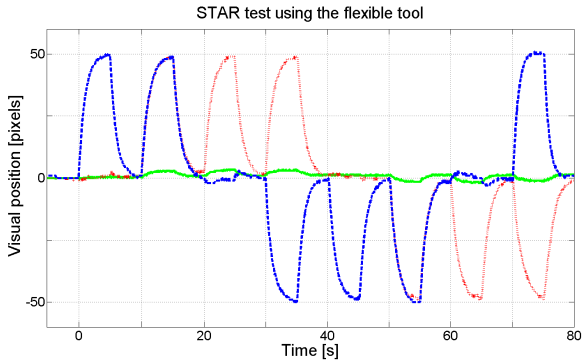

Figure 25. End-effector visual position with respect to time during the STAR test performed with the tool. The three components of the task space vector $\mathbf{x}_v$ are shown: u_R (blue dashed line), v_R (red dotted line) and $u_L - u_R$ (green solid line).

and we initialize the estimation ($\hat{\mathbf{x}}_m(0) = \mathbf{x}_m(0)$) at the beginning of the test. Indeed, the third component of the output vector (i.e. the visual depth $\mathbf{x}^2_m = u_L - u_R$) can also be directly measured at each control step, while the other two components (i.e. the head joints positions, $\mathbf{x}^0_m = \mathbf{q}^0_{head}$ and $\mathbf{x}^1_m = \mathbf{q}^1_{head}$) need to be estimated. Then, to obtain the proper Jacobian at the i control step we pick the IMLE solution which gives the prediction ($\mathbf{x}_m(i) = f(\mathbf{q}_{arm}(i))$) closest to this estimated output $\hat{\mathbf{x}}_m(i)$, and we estimate the Jacobian computing the local slope of this solution.
To evaluate the accuracy of this estimation we compare the estimated head motion to the real head motion during the execution of the arm-head coordination TEST without the tool (described in Section 7.2), in which the head is actually moving. Figure 27 displays both the real head motion, $\mathbf{q}_{head}(t)$, and the estimated one, $\hat{\mathbf{q}}_{head}(t)$. The estimation error is almost zero over the whole trajectory; the maximum instantaneous error measured during the test is $\left\|\mathbf{q}_{head} - \hat{\mathbf{q}}_{head}\right\| = 0.27°$, while the mean and standard deviation are $0.09°$ and $0.05°$ respectively.
This estimation of the head motion allows to select the appropriate IMLE solution for control during the STAR test, as proven by the good results obtained while controlling the arm both without and with the tool (as shown in Figures from 21 to 26).

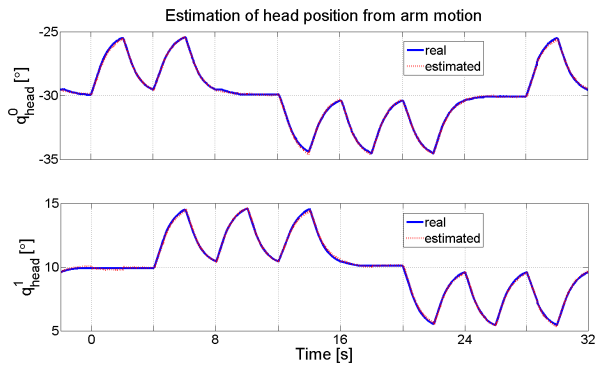

Figure 27. Comparison between real head motion, $\mathbf{q}_{head}(t)$ blue solid line, and estimated one, $\hat{\mathbf{q}}_{head}(t)$ red dashed line, during the execution of the arm-head coordination test without the tool.

8. Conclusions and Future Work

We presented a novel approach to learn the kinematic model of a redundant robot for task space control that can cope with the dynamic inclusion of tools with different kinematic properties. Modeling the forward kinematics as a multi-valued function and using IMLE (an online multi-valued function approximation algorithm) to learn this model allows an efficient control under dynamically switching contexts (in the case study presented in this paper, the different contexts consist in different tools used for reaching). Differently from previous works in the literature, no assumptions are made about the kinematic properties of the tool. Also, no information is given to the robot about the current tool being used, or when a change or removal of the tool is performed. Moreover, the number of different contexts represented by the model doesn't need to be decided *a priori*, but it is automatically determined by the learning algorithm based on the training samples. Experimental results show the effectiveness of the proposed strategy: after acquiring some training data through autonomous exploration, the robot can easily switch from one tool to another without degradation in the control performance, and can cope with new tools being dynamical included during the control phase. Learning can be performed both during motor babbling and during the control; indeed, we show that the motor babbling part can be limited or eliminated, due to the online nature of IMLE, therefore leading to a complete goal-directed exploration. This strategy is very general and can be applied to any robot that needs to be controlled in a specific task space, given that both the joint space and the task space variables can be measured. Indeed, we presented results controlling both a simulated humanoid robot in the 3D Cartesian space and a real humanoid robot in the 3D visual space (uncalibrated camera coordinates). The real robot experiments show how the system can deal with sensor noise, control delays and inaccuracies; moreover, the robot was able to perform reaching movements using a flexible rubber tool (whose shape changes depending on the arm configuration, due to the influence of gravity), a clear example of complex, difficult to model, kinematics.

We believe that this work constitutes an important step towards the achievement of autonomy and flexibility in robot behaviors. Recently, we proposed an autonomous learning strategy for robotic reaching, which is completely goal-directed and online, and does not require any motor babbling [38]; this form of goal-directed exploration has been further analyzed in a separate publication [39]. Moreover, we showed how the robot can build an internal (motor) representation of its workspace while perfoming goal-directed reaching movements, and how this representation can be used to plan and execute complex reaching movements, like bimanual whole-body reaching [40]. On the basis of these studies, a first extension to this work could be to use IMLE to learn a multi-valued representation of the workspace during goal-directed reaching with different tools, in order to provide the robot with the ability to choose the best tool to perform a specific reaching task, based on its past experience.

A further interesting extension of this study would be the application of IMLE to the online learning of the robot dynamics. This would allow to cope with more complex control problems, and to compare quantitatively with recent works that tackled the same issue (as, for instance, the application proposed in [19]).

References

[1] N. Endo, A. Takanishi, Development of Whole-body Emotional Expression Humanoid Robot for ADL-assistive RT services. Journal of Robotics and Mechatronics 23, 6, pp. 969-977 (2011)

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

[3] J. Peters, S. Schaal, Learning Operational Space Control. Robotics: Science and Systems (2006)

[4] O. Sigaud, C. Salan, V. Padois, On-line regression algorithms for learning mechanical models of robots: A survey. Robotics and Autonomous Systems 59, 12, pp. 1115–1129 (2011).

[5] D. Nguyen-Tuong, J. Peters, Local gaussian process regression for real-time model-based robot control. International Conference on Intelligent Robots and Systems (2008)

[6] S. Vijayakumar, A. D'Souza, S. Schaal, Incremental Online Learning in High Dimensions. Neural Computation 17, 12, pp. 2602-2634 (2005)

[7] B. Damas, J. Santos-Victor, An Online Algorithm for Simultaneously Learning Forward and Inverse Kinematics. International Conference on Intelligent Robots and Systems (2012)

[8] F. Guerin, N. Kruger, D. Kraft, A Survey of the Ontogeny of Tool Use: from Sensorimotor Experience to Planning. IEEE Transactions on Autonomous Mental Development, *Online early access* (2012)

[9] J. Krakauer, Z. Pine, M. Ghilardi, C. Ghez, Learning of Visuomotor Transformations for Vectorial Planning of Reaching Trajectories. Journal of Neuroscience 20, pp. 8916-8924 (2000)

[10] R. Shadmehr, F. Mussa-Ivaldi, Adaptive representation of dynamics during learning of a motor task. Journal of Neuroscience 14, pp. 3208-3224 (1994)

[11] R. Welch, B. Bridgeman, S. Anand, K. Browman, Alternating prism exposure causes dual adaptation and generalization to a novel displacement. Perception and Psychophysics 54, 2, pp. 195-204 (1993)

[12] T. Brashers-Krug, R. Shadmehr, E. Bizzi, Consolidation in human motor memory. Nature 382, pp. 252-255 (1996)

[13] G. Berlucchi, S. Aglioti, The body in the brain: neural bases of corporeal awareness. Trends in Neurosciences 20, 12, pp. 560-564 (1997)

[14] A. Berti, F. Frassinetti, When Far Becomes Near: Remapping of Space by Tool Use. Journal of Cognitive Neuroscience 12, 3, pp. 415-420 (2000)

[15] A. Iriki, M. Tanaka, Y. Iwamura, Coding of modified body schema during tool use by macaque postcentral neurones. NeuroReport 7, pp. 2325-2330 (1996)

[16] K. Narendra, J. Balakrishnan, Adaptive control using multiple models. IEEE Transactions on Automatic Control 42, 2, pp. 171-187 (1997)

[17] D. Wolpert, M. Kawato, Multiple paired forward and inverse models for motor control. Neural Networks 11, 7-8, pp. 1317-1329 (1998)

[18] M. Haruno, D. Wolpert, M. Kawato, Mosaic model for sensorimotor learning and control. Neural Computation 13, 10, pp. 2201-2220 (2001)

[19] N. Sugimoto, J. Morimoto, S. Hyon, M. Kawato, The eMOSAIC model for humanoid robot control. Neural Networks 29, 30, pp. 8-19 (2012)

[20] G. Petkos, S. Vijayakumar, Context estimation and learning control through latent variable extraction: From discrete to continuous contexts. International Conference on Robotics and Automation (2007)

[21] M. Hoffmann, H. Marques, A. Arieta, H. Sumioka, M. Lungarella, R. Pfeifer, Body schema in robotics: A review. IEEE Transactions on Autonomous Mental Development 2, 4, pp. 304–324 (2010)

[22] M. Hikita, S. Fuke, M. Ogino, M. Asada, Cross-modal body representation based on visual attention by saliency. International Conference on Intelligent Robots and Systems (2008)
[23] C. Nabeshima, Y. Kuniyoshi, M. Lungarella, Adaptive body schema for robotic tool-use. Advanced Robotics 20, 10, pp. 1105-1126 (2006)
[24] S. Nishide, J. Tani, T. Takahashi, H. Okuno, T. Ogata, Tool–Body Assimilation of Humanoid Robot Using a Neurodynamical System. IEEE Transactions on Autonomous Mental Development 4, 2, pp. 139-149 (2012)
[25] M. Rolf, J. Steil, M. Gienger, Learning flexible full body kinematics for humanoid tool use. International Symposium on Learning and Adaptive Behavior in Robotic Systems (2010)
[26] L. Xu, M. Jordan, G. Hinton, An Alternative Model for Mixtures of Experts. Advances in Neural Information Processing Systems, pp. 633-640 (1995)
[27] D. Grollman, O. Jenkins, Incremental learning of subtasks from unsegmented demonstration. International Conference on Intelligent Robots and Systems (2010)
[28] A. Ligeois, Automatic supervisory control of the configuration and behavior of multibody mechanisms. Transactions on System, Man and Cybernetics 7, pp. 868-871 (1977)
[29] Y. Nakamura, H. Hanafusa, Inverse kinematic solutions with singularity robustness for robot manipulator control. Transactions of the ASME Journal of Dynamic Systems, Measurement and Control 108, pp. 163–171 (1986)
[30] C. Salaun, V. Padois, O. Sigaud, Control of reundant robots using learned models: an operational space control approach. International Conference on Intelligent Robots and Systems (2009)
[31] V. Tikhanoff, P. Fitzpatrick, G. Metta, L. Natale, F. Nori, A. Cangelosi, An open source simulator for cognitive robotics research: The prototype of the icub humanoid robot simulator. Workshop on Performance Metrics for Intelligent Systems (2008)
[32] G. Metta, P. Fitzpatrick, L. Natale, Yarp: yet another robot platform. International Journal on Advanced Robotics Systems 3, 1, pp. 43-48 (2006)
[33] Y. Ogura, H. Aikawa, K. Shimomura, H. Kondo, A. Morishima, H. Lim, A. Takanishi, Development of a new humanoid robot WABIAN-2. International Conference on Robotics and Automation (2006)
[34] H. Miwa, T. Okuchi, H. Takanobu, A. Takanishi, Development of a new human-like head robot WE-4. International Conference on Intelligent Robots and Systems (2002)
[35] M. Zecca, N. Endo, S. Momoki, K. Itoh, A. Takanishi, Design of the humanoid robot KOBIAN-preliminary analysis of facial and whole body emotion expression capabilities. International Conference on Humanoid Robots (2008)
[36] N. Endo, K. Endo, K. Hashimoto, T. Kojima, F. Iida, A. Takanishi, Integration of Emotion Expression and Visual Tracking Locomotion Based on Vestibulo-Ocular Reflex. International Symposium on Robot and Human Interactive Communication (2010)
[37] G. Metta, G. Sandini, J. Konczak, A developmental approach to visually-guided reaching in artificial systems. Neural Networks 12, 10, pp. 1413-1427 (1999)
[38] L. Jamone, L. Natale, G. Metta, F. Nori, G. Sandini, Autonomous online learning of reaching behavior in a humanoid robot. International Journal of Humanoid Robotics 9, 3, pp. 1250017.1-1250017.26 (2012)
[39] L. Jamone, L. Natale, K. Hashimoto, G. Sandini, A. Takanishi, Learning task space control through goal directed exploration. International Conference on Robotics and Biomimetics (2011)
[40] L. Jamone, L. Natale, G. Sandini, A. Takanishi, Interactive online learning of the kinematic workspace of a humanoid robot. International Conference on Intelligent Robots and Systems (2012)