Online Learning of Humanoid Robot Kinematics Under Switching Tools Contexts

Lorenzo Jamone[1], Bruno Damas[2,3], José Santos-Victor[2] and Atsuo Takanishi[1,4]

***Abstract*— In this paper a novel approach to kinematics learning and task space control, under switching contexts, is presented. Such non-stationary contexts may appear in many robotic tasks: in particular, the changing of the context due to the use of tools with different lengths and shapes is herein studied. We model the robot forward kinematics as a multi-valued function, in which different outputs for the same input query are related to actual different hidden contexts. To do that, we employ IMLE, a recent online learning algorithm that fits an infinite mixture of linear experts to the online stream of training data. This algorithm can directly provide multi-valued regression in a online fashion, while having, for classic single-valued regression, a performance comparable to state-of-the-art online learning algorithms. The context varying forward kinematics is learned online through exploration, not relying on any kind of prior knowledge. Using the proposed approach, the robot can dynamically learn how to use different tools, without forgetting the kinematic mappings concerning previously manipulated tools. No information is given about such tool changes to the learning algorithm, nor any assumption is made about the tool kinematics. To our knowledge this is the most general and efficient approach to learning and control under discrete varying contexts. Some experimental results obtained on a high-dimensional simulated humanoid robot provide a strong support to our approach.**

I. Introduction and Related Work

In the last decades we have seen a remarkable growth of the field of robotics. On one side, there has been a huge increase in the complexity of available robots, with current humanoid robots bearing tens of degrees of freedom and a multitude of different sensors. On the other hand, more sophisticated methods were developed to control these robots, in order to make them execute the desired tasks. However, as robots became more complex, building the analytical models needed for robot control turned 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 made infeasible to obtain adequate and accurate models for such kind of systems [1]; as a consequence, an increasing number of researchers has started to employ modern machine learning techniques to provide these complex robots with the necessary representation capability (see [2] for a recent survey). Among these techniques, there are a few that were very successful at learning the forward kinematics or inverse dynamics of a robot: Gaussian Processes Regression achieves a state of the art performance with respect to estimation generalization error [3], while the Locally Weighted Projection Regression (LWPR) algorithm [4], a 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, making 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 under a varying context, where an unobserved context variable changes the map to be learned. 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 some kind of forgetting factors mechanism. Of course, it is terribly inefficient to relearn the complete mapping every time the context changes, specially 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: some of the earliest work on this subject is given by the works of [5] on adaptive control and the MOSAIC architecture [6], while [7] constitutes a more recent vision on this matter. This latter work identifies three critical issues when learning multiple models for use in robot control. The first issue is how to identify the correct number of models to use without any problem specific information. The other two issues are: i) how to estimate the current context, given that the correct number of models to use is known, and ii) how to use such estimation for either controlling the robot or further training the models. The adaptive control of [5], for instance, considers that an appropriate number of models is given *a priori*, already trained and exhibiting good performance

[1]L. Jamone and A. Takanishi are with the Faculty of Science and Engineering, Waseda University, Tokyo, Japan lorejam@liralab.it, contact@takanishi.mech.waseda.ac.jp

[2]B. Damas and J. Santos-Victor are with the Instituto de Sistemas e Robótica, Instituto Superior Técnico, Lisboa, Portugal. {bdamas,jasv}@isr.ist.utl.pt

[3]B. Damas is with Escola Superior de Tecnologia de Setúbal, Portugal.

[4]A. Takanishi is with the Humanoid Robotics Institute, Waseda University, Tokyo, Japan

This work was partially supported by Handle, First-MM, and Poeticon++ projects from the European FP7 program (grant agreements no 231640, 248258 and 288382 respectively).

within each context; the MOSAIC architecture, 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 is a very optimistic assumption, in this paper authors opinion. Additionally, the results presented for the MOSAIC model are somewhat limited to a very simple system consisting of an object moving along a single direction axis. Finally, the approach presented in [7] ambitiously claims the ability to deal with varying contexts, although, as the authors admit, their 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 labelled 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 unlabelled data points, is required when no trained models exist yet — this however, goes against the online, incremental philosophy of LWPR [4], the function approximation algorithm used in the corresponding simulations.

Some recent works focus more specifically on the problem of adapting the robot kinematics under different tools operation. In [8] 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. Another approach is proposed in [9], 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 joints space dimension being used. Another big limitation of these works is that they can account only for rigid transformations (e.g. they cannot cope with flexible or deformable tools).

This paper takes a different approach, by directly modelling 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 a 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 [10], a recent online, 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 II. 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 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 III. To evaluate the performance of the IMLE algorithm under the discrete varying context situation we used it to learn the kinematics of a simulated iCub humanoid robot, actuating 7 of its joints (4 DOFs for the right arm, 3 DOFs for the waist) in order to control the end-effector position of the robot in the 3D Cartesian space, for different tool geometries and lengths. No information whatsoever was conveyed to the algorithm whenever the tool was changed; moreover, no assumption was made about the size or geometry of the tools being used. The results are shown in Section IV: to our knowledge, this is the most general and efficient approach to learning and control under discrete varying contexts.

II. The IMLE Algorithm

The IMLE algorithm [10] is a probabilistic algorithm that uses a generalized expectation-maximization 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 experts sharing the same input region. It starts with the following probabilistic generative model,

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

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

where the mean ν_j and covariance Σ_j define each expert j active input region, while μ_j and Λ_j define the linear relation from input to output for a particular expert; Ψ_j represents the output noise. The unobserved, latent variable

w_{ij} assigns sample points to experts, while the parameter vector Θ gathers all the parameters to be learned; this model is in essence similar to the one presented in [11]. Some additional priors, however, are additionally defined over the mixture parameters 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 [10].

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

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

where $\hat{\Theta}$ is the most recent estimate for the mixture parameters being learned. Maximization 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 commonly 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, as different solutions in the output space become blended together. The IMLE algorithm tries to find, for a given input query z_q, a set of estimated predictions $\hat{\bar{x}}_k$ 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. After the EM 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 than 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 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 [12], 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, IMLE features a very low computational complexity: for each training point 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 pointLike 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 [10] 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. Since both algorithms have at least a handful of tuning parameters, comparing their performance may of course be debatable, but a yet unpublished comparison of both algorithms under an exhaustive variation of parameters seems to confirm a better performance of IMLE. For prediction, IMLE computational time grows linearly with N_{pred}, the number of multi-valued solutions found.

III. Task Space Control

To control the end-effector position in task space we follow the approach originally proposed in [13], where due to the redundancy of the system two tasks can be simultaneously executed: a main task in the Cartesian space (i.e. positioning of the end-effector) and a secondary task in joints space (i.e. keeping the joints as far as possible from the physical limits), projected in the null space of the main task. Motor velocities $\dot{\mathbf{q}}$ are thus computed as follows:

$$\dot{\mathbf{q}} = K_m J^{\dagger}(\mathbf{q}) \dot{\mathbf{x}}_m^d + K_s (I - J^{\dagger}(\mathbf{q}) J(\mathbf{q})) \ \dot{\mathbf{q}}_s^d \ , \qquad \text{(III.1)}$$

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, $\dot{\mathbf{x}}_m^d$ is the desired task space velocity for the main task and $\dot{\mathbf{q}}_s^d$ is the desired joints space velocity for the secondary task. At every control step these desired velocities are chosen as

$$\dot{\mathbf{x}}_m^d = \mathbf{x}_d - \mathbf{x} \qquad \text{and} \qquad \dot{\mathbf{q}}_s^d = -\nabla M(\mathbf{q}) \ , \qquad \text{(III.2)}$$

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 task. K_m and K_s are positive definite diagonal gain matrices respectively for the main and secondary task. Since our secondary task is to keep the joints as far as possible from their limits we chose $M(\mathbf{q})$ as in [13]:

$$M(\mathbf{q}) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\mathbf{q}_i - \mathbf{a}_i}{\mathbf{a}_i - \mathbf{q}_i^{max}}\right)^2 \quad , \tag{III.3}$$

$$\mathbf{a}_i = \frac{\mathbf{q}_i^{max} + \mathbf{q}_i^{min}}{2} \, , \tag{III.4}$$

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 [14]. In this case the pseudoinverse is computed as follows:

$$J^{\dagger}(\mathbf{q}) = J^T(\mathbf{q})(J(\mathbf{q})J^T(\mathbf{q}) + \lambda^2 I)^{-1} \, , \tag{III.5}$$

with

$$\lambda^2 = \left[1 - \left(\frac{\delta_m}{\delta_T}\right)^2\right] \cdot {\lambda^2}_{MAX} \; ; \tag{III.6}$$

our implementation sets these values to $\delta_T = 0.0001$ and ${\lambda^2}_{MAX} = 0.00005$.

The Jacobian $J(\mathbf{q})$ is obtained estimating the local slope, for 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 [15], 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 then to pick the prediction closest to the previous acquired output point $\mathbf{x}$. This procedure is implicitly assuming that the context does not change very frequently, which seems a reasonable assumption.

IV. Experimental Results

The experiments are carried out using the iCub Dynamic Simulator [16]: snapshots are displayed in Figure 1. 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 III. 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 48x30 cm L shaped tool) are displayed in Figure 1. All the software has been realized using YARP [17]. For the sake of clarity, we recall the definition of joints space vector $\mathbf{q}$ and task space vector $\mathbf{x}$, as used hereinafter:

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

where θ_{sy}, θ_{sp}, θ_{sr} are the shoulder yaw, pitch 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 or 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 I.

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

TABLE I

JOINTS LIMITS OF THE ICUB ROBOT SIMULATOR.

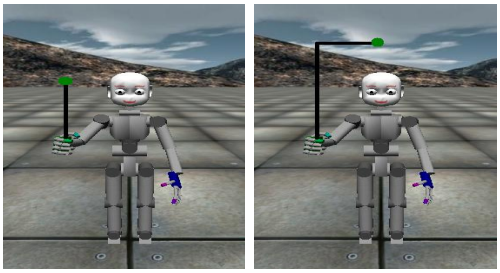

Fig. 1. Snapshots of the iCub Simulator grabbing the two different tools used in the experiments: on the left, the 28 cm stick tool, on the right, the 48x30 cm L shaped tool.

Section IV-A evaluates the online estimation perfomance of IMLE during a motor babbling phase, while in Section IV-B 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.

A. Model estimation during motor babbling

During the motor babbling phase the robot moves to random reference configurations in the joints space using a low-level joint position control, spanning the whole joints space within the robot limits defined in Table I. 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 motor babbling without any tool; after 100,000 training points, the 28 cm stick tool (see left image in Figure 1) 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 continues the motor babbling for more 100,000 points (without the tool). During this procedure, the root mean square error (RMSE) over two independent test sets of 3,000 samples each, S_1 and S_2, is calculated: the joint values $\mathbf{q}$ are the same in both test sets, while the $\mathbf{x}$ values

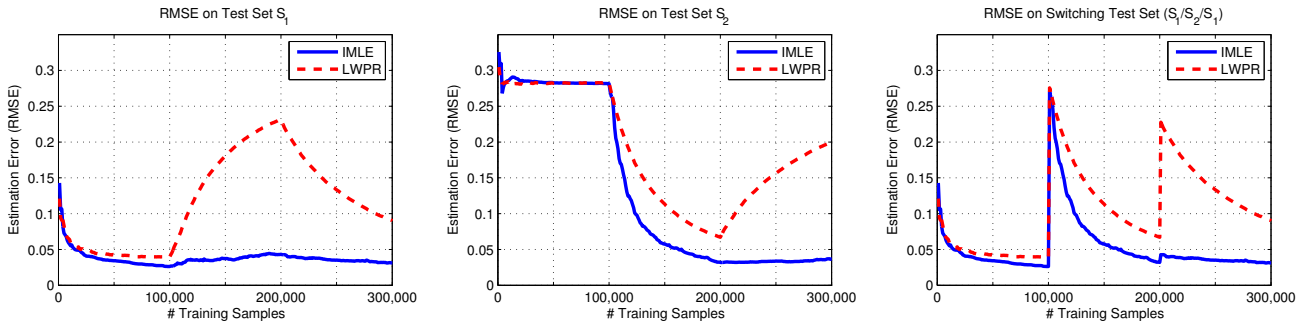

Fig. 2. RMSE of the forward kinematic estimation during motor babbling, using different test sets. IMLE and LWPR are trained first without the tool (until sample 100,000), then with the tool (until sample 200,000) and then again without the tool (until sample 300,000). Left: test set S_1 (no tool). Center: 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.

are the positions of the end-effector, that is either the hand (S_1, no tool) or the tip of the 28 cm stick tool (S_2, tool). The obtained results are shown in Figure 2. 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, and 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 Figure 2, 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, updates the model to the tool condition, forgetting the previous kinematics (the one without the tool); IMLE, on the other hand, creates a new branch of the model for the new tool, without forgetting the previously learned kinematics. After removing the tool (at 200,000 training points) LWPR starts to update the model again to cope with the initial context (the RMSE suddenly raises again, and then slowly decreases with training), while IMLE, having kept the knowledge of the kinematics with no tool, suffers almost no change in its RMSE. In general, after learning a model, the error in the corresponding test set will remain low irrespectively of further training under different kinematics.

B. 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 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 3 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 4. The same movement is then realized by the robot using the 28cm stick tool, after additional motor babbling with the tool (100,000 training samples): results are displayed in Figure 5. Then, the motion is executed again without the tool, with results shown in Figure 6. 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 7). 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 8.

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 48x30cm L shaped tool depicted in the right image in Figure 1; the target positions used in the previous test are shifted in space, as the robot reaches a different workspace when using this tool. Additional motor babbling with the tool is performed (100,000 training points), then the test movement is executed (see Figure 9). Then, more motor babbling without the tool is realized (100,000 training points), and the test movement is executed again (see Figure 10). Overall, this set of experiments leads to the following results:

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 [15] using LWPR.

Trajectories resulting from the control using IMLE (left images in Figures 3, 5, 6, 9 and 10, both images in Figures 7 and 8) 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 good both when controlling the hand and when controlling the tip of a tool. Figure 4 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 0.5 before starting the shorter edge of the cube-shaped trajectory, while it goes to 2.5 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.

b) New tools can be dynamically included in the learned model, not only relying on motor babbling, but also directly during the task space control.

Results in Figure 8 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).

c) Considering single-valued regression the performance of IMLE in the control is in line with the one of LWPR.

As expected from the results in Section IV-A, without including any tool the control performance of IMLE is in line with LWPR (see Figure 3).

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

The superiority of the multi-valued approach can be clearly seen from the control performance in Figures 5 and 6. This is especially evident when dealing with the bigger L shaped tool (see Figures 9 and 10): 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 enough to make the system controllable (see Figure 9). Moreover, the additional motor babbling without the tool does not affect IMLE performance to a big extent, as it does for LWPR (see Figure 10).

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.

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 3, LWPR was creating about 400 local experts only for the robot kinematics without any tool attached.

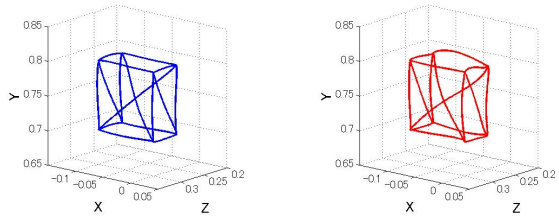

Fig. 3. Task space trajectory of the hand during the test movement, after motor babbling without tool was performed. On the left: using IMLE. On the right: using LWPR.

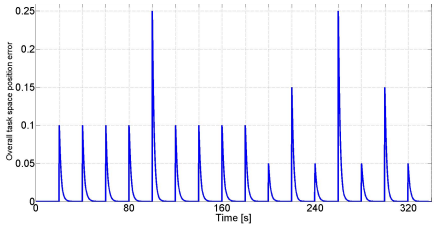

Fig. 4. Task space position error during test movement.

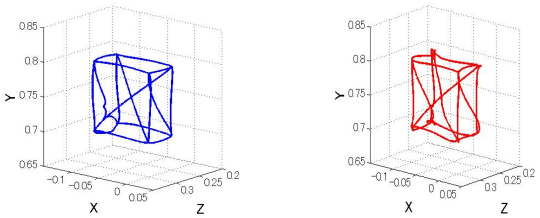

Fig. 5. Task space trajectory of the tip of the stick tool during the test movement, after motor babbling was performed first without and then with the tool. On the left: using IMLE. On the right: using LWPR.

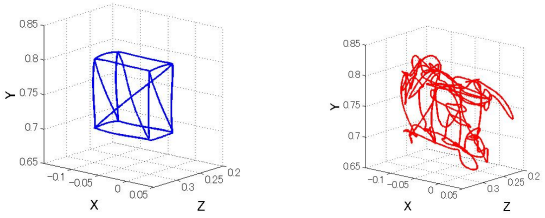

Fig. 6. Task space trajectory of the hand during the test movement, after motor babbling was performed first without and then with the tool. On the left: using IMLE. On the right: using LWPR.

V. Conclusions

We presented a novel approach to learn the kinematic model of a redundant robot for task space control that can cope with the use of tools of different lengths and shapes. Modeling the forward kinematics as a multi-valued function and using IMLE (an online multi-valued function approximation algorithm) to learn this model allows to control the

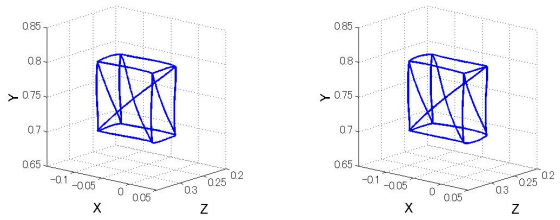

Fig. 7. Task space trajectory during the test movement using IMLE, after motor babbling was performed first without tool, then with tool, and then again without tool. On the left: controlling the hand. On the right: controlling the tip of the tool.

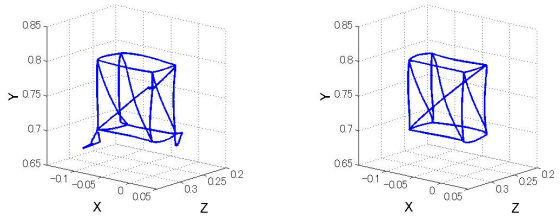

Fig. 8. Task space trajectory of the tip of the stick tool during the test movement using IMLE, without previous motor babbling with the tool. On the left: first iteration of the movement. On the right: after 3 iterations of the movement.

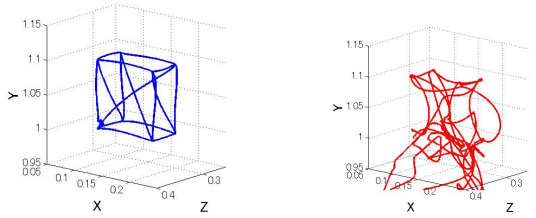

Fig. 9. Task space trajectory of the tip of the L shaped tool during the test movement, after motor babbling was performed first without tool, then with the stick tool, then without tool, and then with the L shaped tool. On the left: using IMLE. On the right: using LWPR.

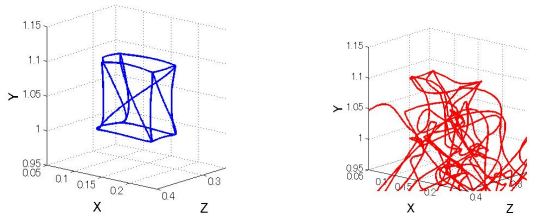

Fig. 10. Task space trajectory of the tip of the L shaped tool during the test movement, after motor babbling was performed first without tool, then with the stick tool, then without tool, then with the L shaped tool, and then again without tool. On the left: using IMLE. On the right: using LWPR.

system under dynamically switching contexts, due to tool changes. 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. Simulation 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. In this work, learning is performed both during motor babbling and control; however, the motor babbling part can be limited or eliminated due to the online nature of IMLE, hence leading to a complete goal-directed exploration, as proposed in [18].

REFERENCES

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

[2] O. Sigaud, C. Salan, and V. Padois, "On-line regression algorithms for learning mechanical models of robots: A survey," *Robotics and Autonomous Systems*, no. 59(12), pp. 1115–1129, 2011.

[3] D. Nguyen-Tuong and J. Peters, "Local gaussian process regression for real-time model-based robot control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 380–385.

[4] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental Online Learning in High Dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[5] K. Narendra and J. Balakrishnan, "Adaptive control using multiple models," *IEEE Transactions on Automatic Control*, vol. 42, no. 2, pp. 171–187, 1997.

[6] M. Haruno, D. Wolpert, and M. Kawato, "Mosaic model for sensorimotor learning and control," *Neural Computation*, vol. 13, no. 10, pp. 2201–2220, 2001.

[7] G. Petkos and S. Vijayakumar, "Context estimation and learning control through latent variable extraction: From discrete to continuous contexts," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 2117–2123.

[8] C. Nabeshima, Y. Kuniyoshi, and M. Lungarella, "Adaptive body schema for robotic tool-use," *Advanced Robotics*, no. 20(10), pp. 1105–1126, 2006.

[9] M. Rolf, J. J. Steil, and M. Gienger, "Learning flexible full body kinematics for humanoid tool use," in *Int. Symp. Learning and Adaptive Behavior in Robotic Systems*, 2010.

[10] B. Damas and J. Santos-Victor, "An Online Algorithm for Simultaneously Learning Forward and Inverse Kinematics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[11] L. Xu, M. Jordan, and G. Hinton, "An Alternative Model for Mixtures of Experts," *Advances in Neural Information Processing Systems*, pp. 633–640, 1995.

[12] D. Grollman and O. Jenkins, "Incremental learning of subtasks from unsegmented demonstration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 261–266.

[13] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *Transactions on System, Man and Cybernetics*, no. 7, pp. 868–871, 1977.

[14] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control*, no. 108, pp. 163–171, 1986.

[15] C. Salaun, V. Padois, and O. Sigaud, "Control of reundant robots using learned models: an operational space control approach," in *International Conference on Intelligent Robots and Systems (IROS)*, St. Luis, USA, October, 11-15 2009, pp. 878–885.

[16] 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*, National Institute of Standards and Technology, Washington DC, August 19-21 2008.

[17] G. Metta, P. Fitzpatrick, and L. Natale, "Yarp: yet another robot platform," *International Journal on Advanced Robotics Systems*, March 2006, special Issue on Software Development and Integration in Robotics.

[18] L. Jamone, L. Natale, K. Hashimoto, G. Sandini, and A. Takanishi, "Learning task space control through goal directed exploration," in *International Conference on Robotics and Biomimetics*. Phuket, Thailand: IEEE-RAS, 2011.