Incremental learning of context-dependent dynamic internal models for robot control

Lorenzo Jamone[1], Bruno Damas[1,2], José Santos-Victor[1]

***Abstract*— Accurate dynamic models can be very difficult to compute analytically for complex robots; moreover, using a pre-computed fixed model does not allow to cope with unexpected changes in the system. An interesting alternative solution is to learn such models from data, and keep them up-to-date through online adaptation. In this paper we consider the problem of learning the robot inverse dynamic model under dynamically varying contexts: the robot learns incrementally and autonomously the model under different conditions, represented by the manipulation of objects of different weights, that change the dynamics of the system. The inverse dynamic mapping is modeled as a multi-valued function, in which different outputs for the same input query are related to different dynamic contexts (i.e. different manipulated objects). The mapping is estimated using IMLE, a recent online learning algorithm for multi-valued regression, and used for Computed Torque control. No information is given about the context switch during either learning or control, nor any assumption is made about the kind of variation in the dynamics imposed by a new contexts. Experimental results with the iCub humanoid robot are provided.**

I. Introduction and Related Work

Adaptation and flexibility are two major requirements for modern robots, both for their application in industry and in more unstructured environment (e.g. homes). From a control theory point of view, exploiting a dynamic model for control leads to better performance (e.g. in terms of accuracy, compliance, energy efficiency) with respect to model-free approaches. However, as robots become more and more complex, computing accurate analytical models is turning more and more difficult, due to the presence of hard-to-model phenomena (e.g. actuator nonlinearities, deformation of soft or elastic components, complex mass distributions); an alternative solution, which is becoming increasingly popular during the recent years, is to estimate these models from data using machine learning methods (see [1], [2] for two recent surveys). In addition, many robotic tasks involve handling and manipulation of different objects, which makes the environment and the mappings to be learned non-stationary. For instance, the dynamics of a robot arm changes during the manipulation of different objects due to the variation of the load of the end-effector. This problem is known as learning and control under varying contexts, where an unobserved context variable changes the mapping that has to be learned and used for the control.

*This work was partially funded by the EU Projects POETICON++ [FP7-ICT-288382] and LIMOMAN [PIEF-GA-2013-628315].

[1]L. Jamone, B. Damas and J. Santos-Victor are with the Instituto de Sistemas e Robótica, Instituto Superior Técnico, Universidade de Lisboa, Portugal. ljamone,bdamas,jasv at isr.ist.utl.pt

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

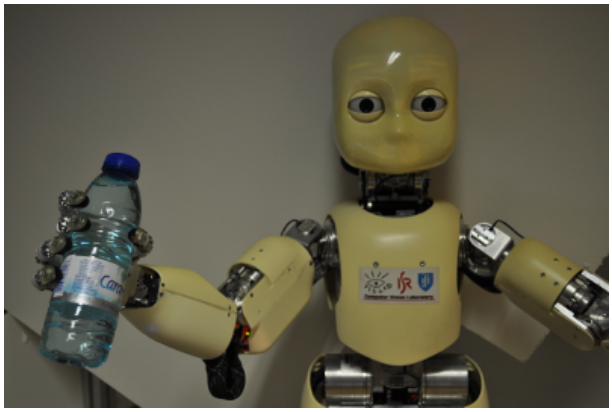

Fig. 1. The iCub robot holding a $33cl$ plastic bottle filled with water.

Looking at the literature of human studies, it can be noticed how 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 [3] or dynamic [4] perturbation. However, once 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 much faster. This might suggest that learning to operate in a new context requires the creation of a new model, whereas re-adaptation can be interpreted as a switching back to a previously learned model. Further evidence is provided by specific work on re-adaptation: during repeated presentation of a kinematic [5] or dynamic [6] perturbation the subjects adapt increasingly rapidly, suggesting that a model of the perturbated context is incrementally learned through motor experience, and it is recalled as soon as the context appears.

Therefore, a promising approach to obtain similar adaptation capabilities in robots is to keep a set of context-specific 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.

A number of solutions has been proposed in the recent years. The early work on adaptive control of Narendra [7] considers that a proper number of models is given a-priori, already trained and showing good performance for each context.

The MOSAIC architecture [8]–[10], 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.

Finally, the approach presented in [11] claims the ability to deal with continuous varying contexts. However, their assumption of an explicit latent context variable 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. Moreover, they make specific assumptions about the kind of variation they expect to occur in the dynamic model — the method only holds under changes in the mass of the object being manipulated.

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 [12], 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 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 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.

In a recent work [13] we exploited the IMLE algorithm to learn the kinematic model of two different humanoid robots for eye-hand coordination and goal-directed reaching in different kinematic contexts (represented by the use of different tools).

In this work we aim to extend those results to the dynamic case, as we want to learn the inverse dynamics of the arm of the iCub humanoid robot [14] in different dynamic contexts (i.e. while the robot is holding different objects, as in Figure 1); details of the robotic platform are provided in Section III. A few approaches have been proposed in the literature for learning inverse dynamics for control, both in simulation [15], [16] and with real manipulators [17]. The main peculiarity of our approach with respect to all previous works is that the learned model is multi-valued, hence allowing control in different dynamic contexts.

The learned model is employed to control the robot movements through the Computed Torque method, as described in Section IV, switching dynamically between the different contexts as soon as they appear. No information is conveyed to the algorithm whenever the context changes; moreover, no assumptions are made about the kind of dynamic variation that the new context is introducing. The results are shown in Section V: although preliminary, these results suggest that we can apply this very general approach to efficiently learn the robot dynamic model under varying dynamic contexts, and exploit such learned model to control the system with powerful methods such as Computed Torque.

II. The IMLE Algorithm

The IMLE algorithm [12] 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 $(\mathbf{z}_i, \mathbf{x}_i)$, where $\mathbf{z}_i \in \mathbb{R}^d$ denotes an input point and $\mathbf{x}_i \in \mathbb{R}^D$ 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: in this way, multi-valued functions can be learned, as the different branches of the multimap can be approximated by different linear models sharing the same input region. This even makes possible the generation of inverse predictions, as reported for instance in [18], [19].

Each expert has an input region where the linear approximation from input to output holds, here denoted as activation region, defined by a Gaussian distribution in the input space. For each expert, another Gaussian distribution is used to model the output noise that affects the linear model. These distributions, together with the parameters that define the linear relation and some other parameters that govern some prior distributions, needed for regularization, must be learned online resorting only to training data. Also, resource allocation is automatically done in this training phase, activating more experts to the mixture as they are needed.

Given the current mixture state, a set of multi-valued predictions is obtained by the IMLE algorithm by clustering the individual experts predictions for a given input query, taking into account the uncertainties on these predictions and their responsibility for the input query, as given by the corresponding activation regions.

More details on the algorithm are available in [12].

III. The robotic platform

In this work we apply our learning and control approach to the iCub robot [14]. The robot is equipped with a 6-axis force/torque sensor placed at the bottom of each arm kinematic chain (i.e. just below the shoulder), that can be used to estimate the torques experienced on the arm joints, by aligning its measures to an analytical model of the system dynamics, as described in [20]. Therefore, since torque sensors on the individual joints are not available, we use this sensor to measure the torque at the joints. Joints positions are measured from incremental encoders mounted on the motors; joints velocities and accelerations are derived from the measured positions using a least-squares algorithm based on an adaptive window, that varies according to the smoothness of the position signal [21].
For the experiments described in this paper we actuate 4 DOFs of the iCub right arm, namely:

. $\mathbf{q} = [\theta_{sp}\ \theta_{sy}\ \theta_{sr}\ \theta_e]^T \in \mathbb{R}^4$

where 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. We will denote the measured joints torques as $\tau_{FT} \in \mathbb{R}^4$. The robot joints limits are defined in Table I.

	$right arm$			
$\mathbf{q}^{min}$	$-80°$	$0°$	$0°$	$20°$
$\mathbf{q}^{max}$	$0°$	$80°$	$80°$	$80°$

TABLE I
JOINTS LIMITS OF THE RIGHT ARM OF THE ICUB ROBOT.

IV. Model Based Control

A. Computed Torque control

The idea of Computed Torque control is that a dynamics model of the system can be exploited to compute the required joints torques to realize desired joints trajectories [22], [23]. An analytical model can be obtained analyzing the physical properties (e.g. CAD data or direct measurements) of the system and then deriving its equation of motion, on the basis of some assumptions. For instance, if we make the rigid bodies assumption, we can model the robot dynamics as:

$$\tau = M(\mathbf{q})\ddot{\mathbf{q}} + F(\mathbf{q}, \dot{\mathbf{q}}), \tag{IV.1}$$

where $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ are the vectors of joint positions, velocities and accelerations of the robot, τ is the vector of joint torques, $M(\mathbf{q})$ is the inertia matrix of the robot, and $F(\mathbf{q}, \dot{\mathbf{q}})$ are all the forces acting on the system (e.g. Coriolis forces, centripetal force, gravity, friction).
In this paper we propose to learn such model from online gathered data, without making any previous assumption about the system. We therefore learn a general mapping of this form:

$$\hat{\tau} = \hat{f}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}). \tag{IV.2}$$

The control input $\mathbf{u}$ is in our case the vector of joint torques given as motor command; this can be computed in different ways depending on the control method. Typically, $\mathbf{u}$ should be computed so to allow the robot to track a desired trajectory. As in this case we are considering a control problem in joint space, the desired trajectory will be provided as desired joint angles, velocities and accelerations. One possibility to compute $\mathbf{u}$ is the so called *Feedforward Nonlinear Control* approach [23], in which the control input is chosen as:

$$\mathbf{u} = M(\mathbf{q}_d)\ddot{\mathbf{q}}_d + F(\mathbf{q}_d, \dot{\mathbf{q}}_d) + K_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + K_p(\mathbf{q}_d - \mathbf{q}), \tag{IV.3}$$

where $\mathbf{q}_d$, $\dot{\mathbf{q}}_d$, $\ddot{\mathbf{q}}_d$ denote desired joint angles, velocities and accelerations. A proper choice of K_v and K_p is proven to bring the tracking error to zero [23].
Using the mapping in Eq. IV.2, the control input becomes:

$$\mathbf{u} = \hat{f}(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d) + K_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + K_p(\mathbf{q}_d - \mathbf{q}). \tag{IV.4}$$

B. Trajectory tracking using learned multi-valued models

Given a set of predictions (i.e. a multi-valued prediction) for an input point, how to choose the one that corresponds to the current (but hidden!) context? One possibility is to represent the hidden context as a latent variable and to use the sequence of observations to estimate it online (as in [11]). This, however, requires the number of hidden contexts to be known, and also requires the existence of a set of distinct sensorimotor maps, one for each context. Instead, what we do here is to i) sample the input space between the last observed point $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \tau)_{t-1}$ and the point for which a prediction is desired $(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d, \hat{\tau}_d)_t$, ii) make multi-valued predictions for each point, that include the local derivatives, iii) find a smooth trajectory of the predictions from the last observed to the desired point (see Figure 2). This procedure produces a prediction that shares the same context with the most recently observed data point.

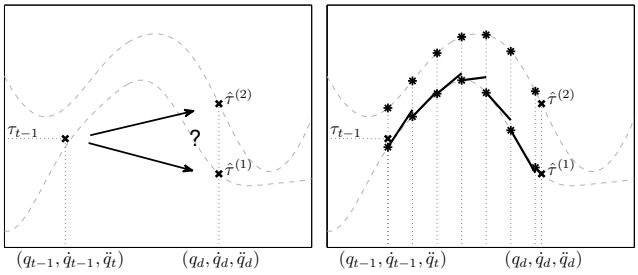

Fig. 2. Enforcing coherence of prediction contexts. Left: given only the last observed data point it is not easy to understand which multi-valued prediction to use. Right: by sampling predictions between the most recent observation and the new input query, a prediction sharing the same context as the previous observation can be found.

V. Experimental Results

We report here results obtained with the iCub humanoid robot. In Section V-A we show how the robot is able to learn a multi-valued mapping that encodes the dynamics of the right arm in different contexts, represented by the presence of different loads on the arm. Then, in Section V-B we

demonstrate that the learned mapping can be used to control the arm with the Computed Torque method; in particular, the robot performs a gravity compensation task while holding objects of different weights.

A. Estimation of multi-valued dynamics

The robot performs first a motor babbling phase in which it moves the right arm to random reference configurations in the joint space using a low-level joint position control, spanning the whole joints space within the limits defined in Table I. Training points, consisting of joint positions, velocities and accelerations $[\mathbf{q}\ \dot{\mathbf{q}}\ \ddot{\mathbf{q}}]$ and respective joint torques τ_{FT}, are gathered at a sampling rate of 5Hz and presented to the online learning algorithm. In the beginning the robot moves without any object in the hand, exploring 1200 random arm configurations and training the network with about 20000 points (first context, c_0). Then, it moves again to the same 1200 arm configurations while holding a small water bottle (see Figure 1), of the approximate weight of $350gr$ (second context, c_1). Then, it moves to other 1200 random configurations, training the network with about 20000 more points. The online estimation of the model is tested with respect to different test sets of 4000 samples that have been acquired before the motor babbling, for each different context. We compute the Root Mean Square Error (RMSE) of the estimation, where the error is defined as the difference between the estimated torque, $\hat{\tau} = \hat{f}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$, and the measured one.

Figure 3 shows the estimation performance of IMLE. The x-axis of each plot (i.e. training samples) is divided into three parts of equal length: 20000 training samples of context c_0, then 20000 of c_1, then other 20000 of c_0. In the top plot it can be seen how the RMSE with respect to test points of c_0 is reduced during the first part; then, during the second part also the estimation error with respect to test points of c_1 is reduced (i.e. context c_1 is learned as well). The estimation error with respect to c_0 reaches a steady state, as it remains constant during further training in the third part. The plot in the middle reveals that the number of linear experts increases during learning of contexts c_0 and c_1, and then remains constant during further training. Finally, the bottom plot displays the average number of solutions found during testing: while before experiencing context c_1 IMLE always finds only one solution (i.e. the solution that explains context c_0), after some training points of c_1 are acquired the solutions become always two, and this holds even when further training samples of c_0 are provided.

To prove that learning of one context does not affect the estimation of another, in Figure 4 we display the RMSE during the all learning sequence (divided in three equal parts, like in Figure 3: c_0, then c_1, then c_0 again) with respect to test sets of 4000 samples each that belong to a specific context: c_0 in the top plot and c_1 in the bottom plot. It is clear from the top plot how training with points of context c_1 does not affect the estimation of c_0, as the RMSE remains reasonably low during the second and third part of the plot; the dual effect can be seen in the bottom plot.

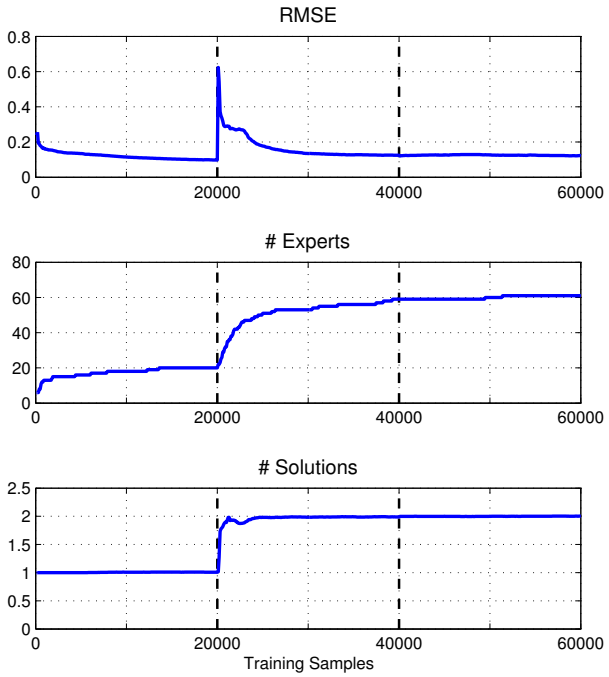

Fig. 3. From top to bottom: estimation error with respect to the test set of the relative context, number of linear experts created and average number of solutions, during online learning of the robot dynamics experiencing different contexts, c_0 from 0 to 20000, c_1 from 20000 to 40000, c_0 from 40000 to 60000.

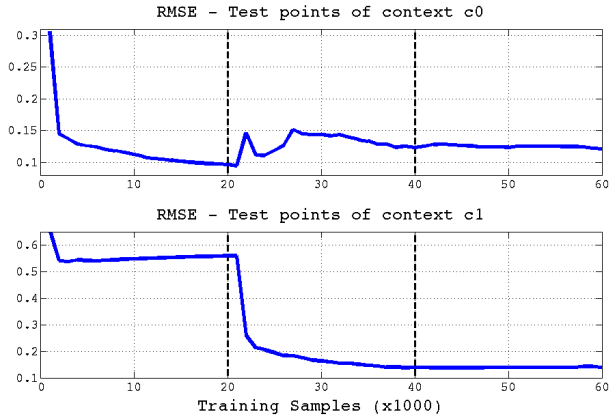

Fig. 4. Estimation error with respect to the test set of a specific context: top image, test set of context c_0, bottom image, test set of context c_1.

B. Gravity compensation control

We present here control results related to the gravity compensation task. The task is to keep the arm in position, hence generating the appropriate joint torques to counteract gravity. With respect to the Computed Torque control framework introduced in Section IV, and in particular to the *Feedforward Nonlinear Control* approach that we described, this means that the task is to track a desired joints trajectory that is defined as: $\mathbf{q}_d = \mathbf{q}$ (i.e. the current joints position), $\dot{\mathbf{q}}_d = 0$, $\ddot{\mathbf{q}}_d = 0$ (i.e. no velocity and no acceleration). We tested the controller to keep the arm in three different test configurations, namely $\mathbf{q}^{t1} = [-60\ 50\ 30\ 55]$, $\mathbf{q}^{t2} =$

$[-50\ \ 60\ \ 40\ \ 65]$, $\mathbf{q}^{t3} = [-40\ \ 40\ \ 20\ \ 45]$, for about ten seconds for each configuration; the hand was either empty (i.e. context c_0) or grasping the bottle (i.e. context c_1).
Figure 5 and Figure 6 show the results for context c_0. As it can be see in Figure 5, the arm is kept in the desired configurations by the controller, as the position of each of the four arm joints (solid blue line) tracks the desired reference (dashed red line). Figure 6 displays how the torque of each joint (solid blue line) follows the desired torque estimated by the learned mapping, $\tau_d = \hat{f}(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)$ (dashed red line), due to the application of the control input, $\mathbf{u} = \hat{f}(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d) + K_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + K_p(\mathbf{q}_d - \mathbf{q})$ (thin solid green line).

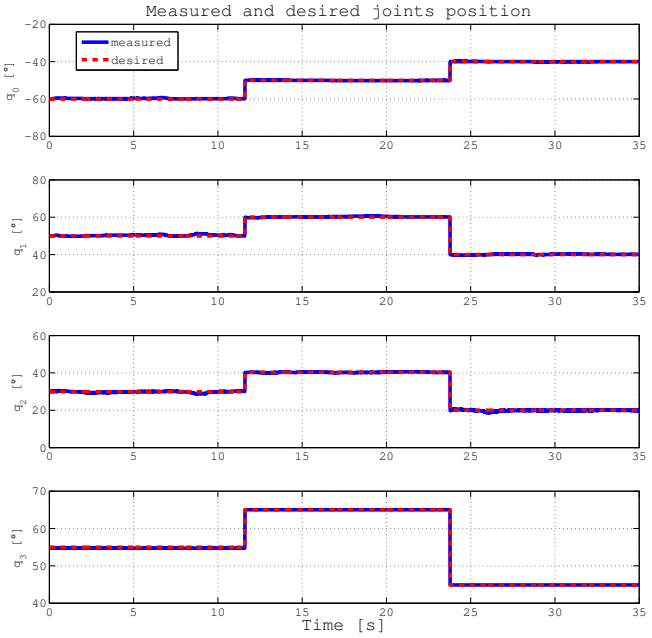

Fig. 5. Measured (solid blue line) and desired (dashed red line) joint position during gravity compensation control with context c_0, for each joint of the right arm.

Figure 7 and Figure 8 display the results for context c_1. As shown in Figure 5, the arm is kept close to the desired configurations; however, the performance of the controller when the robot is grasping the bottle (i.e. context c_1) is worst than in the previous case (i.e. context c_0). It can be noticed from Figure 8 how the controller is generating much bigger torques in order to compensate for the position and velocity errors. As a result, the arm is able to counteract the force of gravity (i.e. it does not fall down), but with not negligible oscillations that were not present in the previous case. Nevertheless, this problem is not related to the model estimation performed by IMLE, as the accuracy of the estimation is the same for both contexts, as proven by the results shown in Section V-A. On the contrary, the oscillations are caused by the current implementation of the low-level torque control on the iCub. Indeed, our investigations show that when a relatively high load is added on the hand (e.g. context c_1, robot grasping a bottle) the controller is not able to follow the

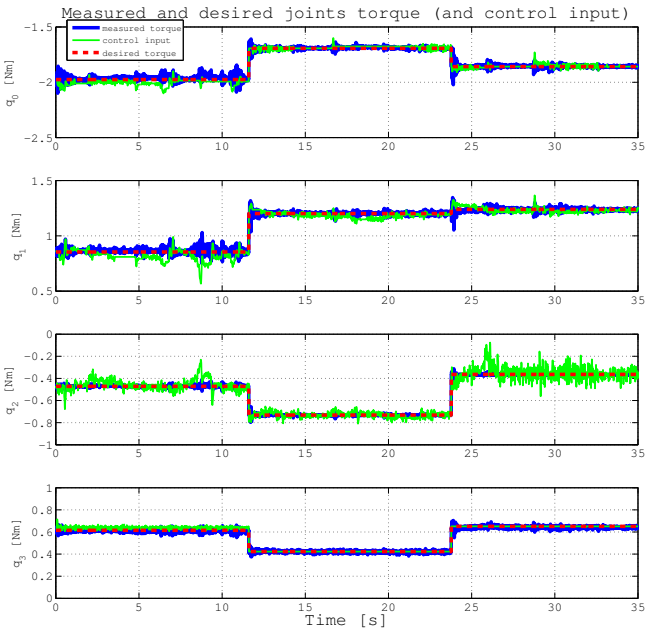

Fig. 6. Measured (solid blue line) and desired (dashed red line) joint torque during gravity compensation control with context c_0, for each joint of the right arm. The controller output (thin solid green line) is also plotted.

torque reference accurately, especially in the three coupled joints of the shoulder. A new version of the controller, that includes friction compensation and a better computation of the shoulder joints decoupling, will be released soon by the iCub software developers, and it will allow accurate control also with high loads.

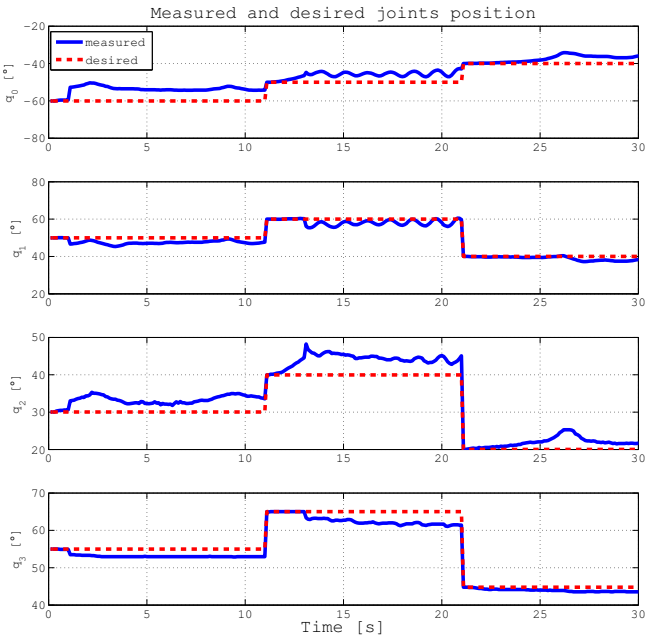

Fig. 7. Measured (solid blue line) and desired (dashed red line) joint position during gravity compensation control with context c_1, for each joint of the right arm.

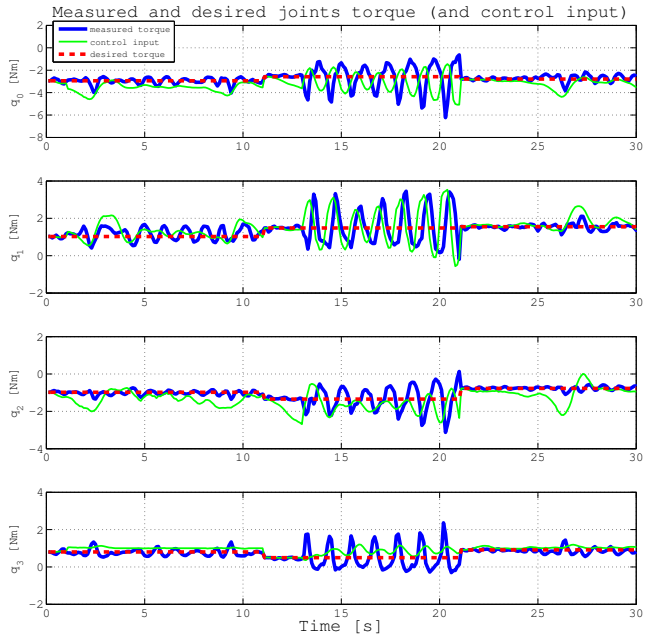

Fig. 8. Measured (solid blue line) and desired (dashed red line) joint torque during gravity compensation control with context c_1, for each joint of the right arm. The controller output (thin solid green line) is also plotted.

VI. Conclusions and Future Work

We presented a novel approach to learn incrementally the dynamic model of a robot in different dynamic contexts. We do not make any assumption about the kinematics and dynamics of the system, about the number of different contexts that will be experienced, and about how these contexts will change the system dynamics. Also, context switch is not signaled to the learning algorithm: the multi-valued mapping is estimated from an online stream of unlabeled data.

We report results obtained on a real robotic platform, the iCub humanoid robot, in which we accurately estimate a multi-valued model of the arm dynamics in two different contexts: the arm without any additional load (empty hand) and while the hand is grasping a $33cl$ bottle filled with water. Then, we provide preliminary results on the use of such learned model for Computed Torque control under the different contexts, in particular for the case of gravity compensation (i.e. keeping the arm in position while counteracting the force of gravity).

Although the model estimation is equally accurate for both contexts, the controller performs better with one of the contexts, with respect to the other: this is related to the current implementation of the iCub low-level torque control, that cannot deal very efficiently with high loads.

Therefore, the next step is to test our system with an improved version of such controller, that is going to be released soon, and to use the learned model for tracking more complex trajectories than the simple one we used in the reported experiments.

References

[1] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, 2011.

[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] J. Krakauer, Z. Pine, M. Ghilardi, and C. Ghez, "Learning of visuomotor transformations for vectorial planning of reaching trajectories," *Journal of Neuroscience*, vol. 20, pp. 8916–8924, 2000.

[4] R. Shadmehr and F. Mussa-Ivaldi, "Adaptive representation of dynamics during learning of a motor task," *Journal of Neuroscience*, vol. 14, pp. 3208–3224, 1994.

[5] R. Welch, B. Bridgeman, S. Anand, and K. Browman, "Alternating prism exposure causes dual adaptation and generalization to a novel displacement," *Perception and Psychophysics*, vol. 54, no. 2, pp. 195–204, 1993.

[6] T. Brashers-Krug, R. Shadmehr, and E. Bizzi, "Consolidation in human motor memory," *Nature*, vol. 382, pp. 252–255, 1996.

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

[8] D. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*, vol. 11, no. 7-8, pp. 1317–1329, 1998.

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

[10] N. Sugimoto, J. Morimoto, S. Hyon, and M. Kawato, "The emosaic model for humanoid robot control," *Neural Networks*, vol. 29, no. 30, pp. 8–19, 2012.

[11] 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.

[12] B. Damas and J. Santos-Victor, "Online learning of single- and multivalued functions with an infinite mixture of linear experts." *Neural computation*, vol. 25, no. 11, pp. 3044–91, 2013.

[13] L. Jamone, B. Damas, N. Endo, J. Santos-Victor, and A. Takanishi, "Incremental development of multiple tool models for robotic reaching through autonomous exploration," *Paladyn Journal of Behavioral Robotics*, vol. 3, no. 3, pp. 113–127, 2013.

[14] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, A. Bernardino, and L.Montesano, "The icub humanoid robot: an open-systems platform for research in cognitive development," *Neural Networks*, vol. 23, no. 8-9, pp. 1125–1134, 2010.

[15] J. Sun de la Cruz, D. Kulic, and W. Owen, "Learning inverse dynamics for redundant manipulator control," in *International Conference on Autonomous and Intelligent Systems (AIS)*, 2010, pp. 1–6.

[16] S. Ulbrich, M. Bechtel, T. Asfour, and R. Dillmann, "Learning robot dynamics with kinematic bezier maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 3598–3604.

[17] D. Nguyen-Tuong, J. Peters, and M. Seeger, "Computed torque control with nonparametric regression models," in *American Control Conference*, 2008.

[18] 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, pp. 1499–1506.

[19] B. Damas, L. Jamone, and J. Santos-Victor, "Open and Closed-Loop Task Space Trajectory Control of Redundant Robots Using Learned Models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2013.

[20] M. Fumagalli, S. Ivaldi, M. Randazzo, L. Natale, G. Metta, G. Sandini, and F. Nori, "Force feedback exploiting tactile and proximal force/torque sensing," *Autonomous Robots*, vol. 33, pp. 381–398, 2012.

[21] F. Janabi-Sharifi, V. Hayward, and C.-S. Chen, "Discrete-time adaptive windowing for velocity estimation," *Control Systems Technology, IEEE Transactions on*, vol. 8, no. 6, pp. 1003–1009, 2000.

[22] J. E. Slotine and W. Li, *Applied NonLinear Control*. Prentice-Hall International Eds., 1991.

[23] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Prentice Hall, 2004.