SENSOR-ASSISTED ADAPTIVE MOTOR CONTROL UNDER CONTINUOUSLY VARYING CONTEXT

Heiko Hoffmann, Georgios Petkos, Sebastian Bitzer, and Sethu Vijayakumar
Institute of Perception, Action and Behavior, School of Informatics, University of Edinburgh, Edinburgh, UK
heiko@clmc.usc.edu, g.petkos@sms.ed.ac.uk, s.bitzer@ed.ac.uk, sethu.vijayakumar@ed.ac.uk

Keywords: Adaptive control, context switching, Kalman filter, force sensor, robot simulation

Abstract: Adaptive motor control under continuously varying context, like the inertia parameters of a manipulated object, is an active research area that lacks a satisfactory solution. Here, we present and compare three novel strategies for learning control under varying context and show how adding tactile sensors may ease this task. The first strategy uses only dynamics information to infer the unknown inertia parameters. It is based on a probabilistic generative model of the control torques, which are linear in the inertia parameters. We demonstrate this inference in the special case of a single continuous context variable – the mass of the manipulated object. In the second strategy, instead of torques, we use tactile forces to infer the mass in a similar way. Finally, the third strategy omits this inference – which may be infeasible if the latent space is multi-dimensional – and directly maps the state, state transitions, and tactile forces onto the control torques. The additional tactile input implicitly contains all control-torque relevant properties of the manipulated object. In simulation, we demonstrate that this direct mapping can provide accurate control torques under multiple varying context variables.

1 INTRODUCTION

In feed-forward control of a robot, an internal inverse-model of the robot dynamics is used to generate the joint torques to produce a desired movement. Such a model always depends on the context in which the robot is embedded, its environment and the objects it interacts with. Some of this context may be hidden to the robot, e.g., properties of a manipulated object, or external forces applied by other agents or humans.

An internal model that does not incorporate all relevant context variables needs to be relearned to adapt to a changing context. This adaptation may be too slow since sufficiently many data points need to be collected to update the learning parameters. An alternative is to learn different models for different contexts and to switch between them (Narendra and Balakrishnan, 1997; Narendra and Xiang, 2000; Petkos et al., 2006) or combine their outputs according to a predicted error measure (Haruno et al., 2001; Wolpert and Kawato, 1998). However, the former can handle only previously-experienced discrete contexts, and the latter have been tested only with linear models.

The above studies learn robot control based purely on the dynamics of the robot; here, we demonstrate the benefit of including tactile forces as additional input for learning *non-linear* inverse models under *continuously-varying* context. Haruno et al. (Haruno et al., 2001) already use sensory information, but only for mapping a visual input onto discrete context states.

Using a physics-based robot-arm simulation (Fig. 1), we present and compare three strategies for learning inverse models under varying context. The first two infer the unknown property (hidden context) of an object during its manipulation (moving along a given trajectory) and immediately use this estimated property for computing control torques.

In the first strategy, only dynamic data are used, namely robot state, joint acceleration, and joint torques. The unknown inertia parameters of an object are inferred using a probabilistic generative model of the joint torques.

In the second strategy, we use instead of the torques the tactile forces exerted by the manipulated object. The same inference as above can be carried out given that the tactile forces are linear in the object's mass. We demonstrate both of these steps with mass as the varying context. If more context variables are changing, estimat-

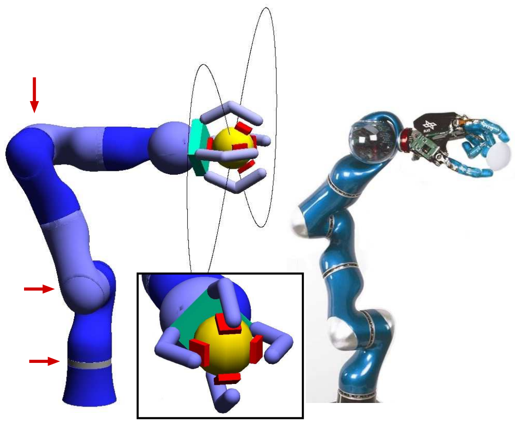
Figure 1: Simulated robot arm with gripper and force sensors and its real counterpart, the DLR light-weight arm III. Arrows indicate the three active joints used for the experiments. The curve illustrates the desired trajectory of the ball.

ing only the mass as hidden context is insufficient. Particularly, if also the mass distribution changes, both the center of mass and inertia tensor vary, leading to a high-dimensional latent variable space, in which inference may be infeasible given a limited number of data points.

In our third strategy, we use a direct mapping from robot state, joint accelerations, and tactile forces onto control torques. This mapping allows accurate control even with more than one changing context variable and without the need to extract these variables explicitly.

The remainder of this article is organized as follows. Section 2 briefly introduces the learning of inverse models under varying context. Section 3 describes the inference of inertia parameters based only on dynamic data. Section 4 describes the inference of mass using tactile forces. Section 5 motivates the direct mapping. Section 6 describes the methods for the simulation experiments. Section 7 shows the results of these experiments; and Section 8 concludes the article.

2 LEARNING DYNAMICS

Complex robot structure and non-rigid body dynamics (e.g, due to hydraulic actuation) can make analytic solutions to the dynamics inaccurate and cumbersome to derive. Thus, in our approach, we learn the dynamics for control from movement data. Particularly, we learn an inverse model, which maps the robot's state (here, joint angles θ and their velocities $\dot{\theta}$) and its desired change (joint accelerations $\ddot{\theta}$) onto the motor commands (joint torques τ) that are required to produce this change,

$$\tau = \mu(\theta, \dot{\theta}, \ddot{\theta}) \quad . \tag{1}$$

Under varying context, learning (1) is insufficient. Here, the inverse model depends on a context variable π,

$$\tau = \mu(\theta, \dot{\theta}, \ddot{\theta}, \pi) \quad , \tag{2}$$

In Sections 3 and 4, we first infer the hidden context variable and then plug this variable into function (2) to compute the control torques. In Section 5, the context variable π is replaced by sensory input that implicitly contains the hidden context.

3 INFERRING CONTEXT FROM DYNAMICS

During robot control, hidden inertia parameters can be inferred by observing control torques and corresponding accelerations (Petkos and Vijayakumar, 2007). This inference can be carried out efficiently because of a linear relationship in the dynamics, as shown in the following.

3.1 Linearity in robot dynamics

The control torques τ of a manipulator are linear in the inertia parameters of its links (Sciavicco and Siciliano, 2000). Thus, μ can be decomposed into

$$\tau = \Phi(\theta, \dot{\theta}, \ddot{\theta})\,\pi \quad . \tag{3}$$

Here, π contains the inertia parameters, $\pi = [m_1, m_1 l_{1x}, m_1 l_{1y}, m_1 l_{1z}, J_{1xx}, J_{1xy}, ..., m_n, m_n l_{nx}, m_n l_{ny}, m_n l_{nz}, J_{nxx}, ..., J_{nzz}]$, where m_i is the mass of link i, l_i its center-of-mass, and J_i its inertia tensor. The dynamics of a robot holding different objects only differs in the π parameters of the combination of object and end-effector link (the robot's link in contact with the object). To obtain Φ, we need to know for a set of contexts c the inertia parameters π_c and the corresponding torques τ_c. Given a sufficient number of τ_c and π_c values, we can compute Φ using ordinary least squares. After computing Φ, the robot's dynamics can be adjusted to different contexts by multiplying Φ with the inertia parameters π. The following two sections show how to estimate π once Φ has been found.

3.2 Inference of inertia parameters

Given the linear equation (3) and the knowledge of Φ, the inertia parameters π can be inferred. Assuming Gaussian noise in the torques τ, the probability density $p(\tau|\theta, \dot{\theta}, \ddot{\theta}, \pi)$ equals

$$p(\tau|\theta, \dot{\theta}, \ddot{\theta}, \pi) = \mathcal{N}(\Phi\pi, \Sigma) \quad , \tag{4}$$

where $\mathcal{N}$ is a Gaussian function with mean $\Phi\pi$ and covariance Σ. Given $p(\tau|\theta, \dot{\theta}, \ddot{\theta}, \pi)$, the most probable inertia parameters π can be inferred using Bayes' rule, e.g., by assuming a constant prior probability $p(\pi)$:

$$p(\pi|\tau, \theta, \dot{\theta}, \ddot{\theta}) \propto p(\tau|\theta, \dot{\theta}, \ddot{\theta}, \pi) \quad . \tag{5}$$

3.3 Temporal correlation of inertia parameters

The above inference ignores the temporal correlation of the inertia parameters. Usually, however, context changes infrequently or only slightly. Thus, we may describe the context π at time step $t + 1$ as a small random variation of the context at the previous time step:

$$\pi_{t+1} = \pi_t + \varepsilon_{t+1} \quad , \tag{6}$$

where ε is a Gaussian-distributed random variable with constant covariance Ω and zero mean. Thus, the transition probability $p(\pi_{t+1}|\pi_t)$ is given as

$$p(\pi_{t+1}|\pi_t) = \mathcal{N}(\pi_t, \Omega) \quad . \tag{7}$$

Given the two conditional probabilities (4) and (7), the hidden variable π can be described with a Markov process (Fig. 2), and the current estimate π_t can be updated using a Kalman filter (Kalman, 1960). Written with probability distributions, the filter update is

$$p(\pi_{t+1}|\tau^{t+1}, x^{t+1}) = \eta\, p(\tau_{t+1}|x_{t+1}, \pi_{t+1}) \cdot \int p(\pi_{t+1}|\pi_t) p(\pi_t|\tau^t, x^t) d\pi_t \quad , \tag{8}$$

where η is a normalization constant, and x stands for $\{\theta, \dot{\theta}, \ddot{\theta}\}$ to keep the equation compact. A variable with superscript t stands for all observations (of that variable) up to time step t.

3.4 Special case: Inference of mass

We will demonstrate the above inference in the case of object mass m as the hidden context. This

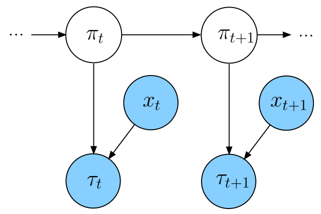

Figure 2: Hidden Markov model for dependence of torques τ on context π. Here, the state and state transitions are combined to a vector $x = \{\theta, \dot{\theta}, \ddot{\theta}\}$.

restriction essentially assumes that the shape and center of mass of the manipulated object are invariant. If m is the only variable context[1], the dynamic equation is linear in m,

$$\tau = g(\theta, \dot{\theta}, \ddot{\theta}) + m\, h(\theta, \dot{\theta}, \ddot{\theta}) \quad . \tag{9}$$

In our experiments, we first learn two mappings $\tau_1(\theta, \dot{\theta}, \ddot{\theta})$ and $\tau_2(\theta, \dot{\theta}, \ddot{\theta})$ for two given contexts m_1 and m_2. Given these mappings, g and h can be computed.

To estimate m, we plug (4) and (7) into the filter equation (8) and use (9) instead of (3). Furthermore, we assume that the probability distribution of the mass at time t is a Gaussian with mean m_t and variance Q_t. The resulting update equations for m_t and Q_t are

$$m_{t+1} = \frac{h^T \Sigma^{-1} (\tau - g) + \frac{m_t}{Q_t + \Omega}}{\frac{1}{Q_t+\Omega} + h^T \Sigma^{-1} h} \quad , \tag{10}$$

$$Q_{t+1} = \left(\frac{1}{Q_t + \Omega} + h^T \Sigma^{-1} h \right)^{-1} \quad . \tag{11}$$

Section 7 demonstrates the result for this inference of m during motor control. For feed-forward control, we plug the inferred m into (9) to compute the joint torques τ.

4 INFERRING CONTEXT FROM TACTILE SENSORS

For inferring context, tactile forces measured at the interface between hand and object may

[1]This assumption is not exactly true in our case. A changing object mass also changes the center of mass and inertia tensor of the combination end-effector link plus object. Here, to keep the demonstration simple, we make a linear approximation and ignore terms of higher order in m – the maximum value of m was about one third of the mass of the end-effector link.

serve as a substitute for the control torques. We demonstrate this inference for the special case of object mass as context.

The sensory values s_i (the tactile forces) are linear in the mass m held in the robot's hand, as shown in the following. In the reference frame of the hand, the acceleration a of an object leads to a small displacement dx (Fig. 3).

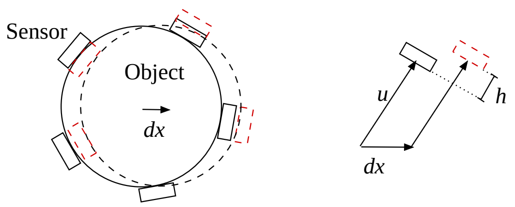

Figure 3: A force on the object held in the robot's hand leads to a displacement dx. This displacement shifts each sensor at position u (relative to the object's center) by h.

This displacement pushes each sensor by the amount h_i depending on the sensor's position u_i. Let e_i be a vector of unit length pointing in the direction of u_i, then $h_i = e_i^T dx$. Our sensors act like Hookean springs; thus, the resulting force equals $f_i = \kappa h_i e_i$, with κ being the spring constant. Since the object is held such that it cannot escape the grip, the sum of sensory forces f_i must equal the inertial force $m\,a$,

$$m\,a = \sum_{i=1}^{n} f_i = \kappa \sum_i (e_i^T dx)\, e_i \quad . \tag{12}$$

This linear equation allows the computation of dx,

$$dx = \frac{m}{\kappa}(E^T E)^{-1} a \quad , \tag{13}$$

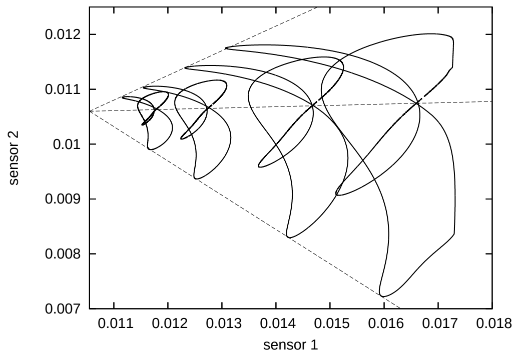

Figure 4: Two-dimensional projection of sensor values during figure-8 movements with four different masses. From left to right, the mass increases as 0.005, 0.01, 0.02, and 0.03.

where E is a matrix with row vectors e_i. Thus, each f_i is proportional to m. The total force measured at a sensor equals f_i plus a constant grip force (whose sum over all sensors equals zero). Therefore, the sensory values s can be written as

$$s = s_0 + m\,\varphi(\theta, \dot{\theta}, \ddot{\theta}) \quad , \tag{14}$$

where φ is a function depending on the state and acceleration of the robot arm. This linearity is illustrated in Fig. 4 using data from our simulated sensors. Based on (14), the same inference as in Section 3 is possible using (10) and (11), and the estimated m can be used for control by using (9).

5 DIRECT MAPPING

The above inference of mass fails if we have additional hidden context variables, e.g., if the robot hand holds a dumb-bell, which can swing. In this general case, we could still use the inference based on dynamics. However, since for the combination of end-effector and object, both the inertia tensor and the center of mass vary, we need to estimate 10 hidden context variables (Sciavicco and Siciliano, 2000). Given the limited amount of training data that we use in our experiments, we expect that inference fails in such a high-dimensional latent space.

As alternative, we suggest using the sensory values as additional input for learning feed-forward torques. Thus, the robot's state and desired acceleration are augmented by the sensory values, and this augmented input is directly mapped on the control torques. This step avoids inferring unknown hidden variables, which are not our primary interest; our main goal is computing control torques under varying context.

The sum of forces measured at the tactile sensors equals the force that the manipulated object exerts on the robotic hand. This force combined with the robot's state and desired acceleration is sufficient information for predicting the control torques. Thus, tactile forces contain the relevant effects of an unknown varying context. This context may be, e.g., a change in mass, a change in mass distribution, or a human exerting a force on the object held by the robot.

Depending on the number of sensors, the sensory input may be high-dimensional. However, since the sensors encode only a force vector, the intrinsic dimensionality is only three. For learning the mapping $\tau(\theta, \dot{\theta}, \ddot{\theta}, s)$, regression techniques exist, like locally-weighted projection regression (Vijayakumar et al., 2005), that can exploit this reduced intrinsic dimensionality efficiently. In the following, we demonstrate the validity of our arguments in a robot-arm simulation.

6 ROBOT SIMULATION

This section describes the methods: the simulated robot, the simulated tactile sensors, the control tasks, the control architecture, and the learning of the feed-forward controller.

6.1 Simulated robot arm

Our simulated robot arm replicates the real Light-Weight Robot III designed by the German Aerospace Center, DLR (Fig. 1). The DLR arm has seven degrees-of-freedom; however, only three of them were controlled in the present study; the remaining joints were stiff. As end-effector, we attached a simple gripper with four stiff fingers; its only purpose was to hold a spherical object tightly with the help of five simulated force sensors. The physics was computed with the Open Dynamics Engine (`http://www.ode.org`).

6.2 Simulated force sensors

Our force sensors are small boxes attached to damped springs (Fig. 1). In the simulation, damped springs were realized using slider joints, whose positions were adjusted by a proportional-derivative controller. The resting position of each spring was set such that it was always under pressure. As sensor reading s, we used the current position of a box (relative to the resting position).

6.3 Control tasks

We used two tasks: moving a ball around an eight figure and swinging a dumb-bell. In the first, one context variable was hidden, the mass of the ball. The maximum mass ($m = 0.03$) of a ball was about one seventh of the total mass of the robot arm. In the second task, two variables were hidden: the dumb-bell mass and its orientation. The two ball masses of the dumb-bell were equal.

In all tasks, the desired trajectory of the end-effector was pre-defined (Figs. 1 and 5) and its inverse in joint angles pre-computed. The eight was traversed only once lasting 5000 time steps, and for the dumb-bell, the end-effector swung for two periods together lasting 5000 time steps and followed by 1000 time steps during which the end-effector had to stay in place (here, the control torques need to compensate for the swinging dumb-bell). In both tasks, a movement started and ended with zero velocity and acceleration. The velocity profile was sinusoidal with one peak.

For each task, three trajectories were pre-computed: eight-figures of three different sizes (Fig. 1 shows the big eight; small and medium eight are 0.9 and 0.95 of the big-eight's size, see Fig. 11) and three lines of different heights (Fig. 13). For training, data points were used from the two extremal trajectories, excluding the middle trajectory. For testing, all three trajectories were used.

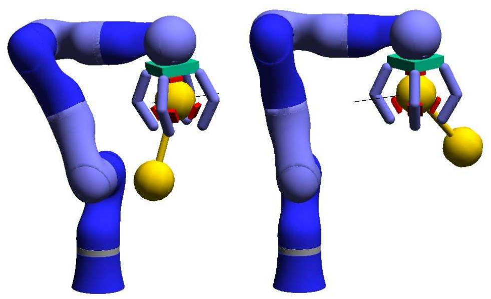

Figure 5: Robot swinging a dumb-bell. The black line shows the desired trajectory.

6.4 Control architecture

We used an adaptive controller and separated training and test phases. To generate training patterns, a proportional-integral-derivative (PID) controller provided the joint torques. The proportional gain was chosen to be sufficiently high such that the end-effector was able to follow the desired trajectories. The integral component was initialized to a value that holds the arm against gravity (apart from that, its effect was negligible).

For testing, a composite controller provided the joint torques (Fig. 6). The trained feed-

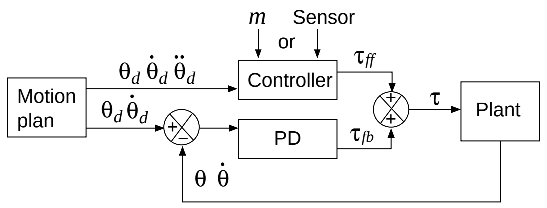

Figure 6: Composite control for the robot arm. A feed-forward controller is put in parallel with error feed-back (low PD gain).

forward controller was put in parallel with a low-gain error feed-back (its PD-gain was 1% of the gain used for training in the ball case and 4% in the dumb-bell case). For those feed-forward mappings that require an estimate of the object's mass, this estimate was computed using a Kalman filter as described above (the transition noise Ω of m was set to 10^{-9} for both dynamics and sensor case).

6.5 Controller learning

The feed-forward controller was trained on data collected during pure PID-control. For each mass context, 10 000 data points were collected in the ball case and 12 000 data points in the dumb-bell case. Half of these points (every second) were used for training and the other half for testing the regression performance. Three types of mappings were learned. The first maps the state and acceleration values $(\theta, \dot{\theta}, \ddot{\theta})$ onto joint torques. The second maps the same input onto the five sensory values, and the third maps the sensor augmented input $(\theta, \dot{\theta}, \ddot{\theta}, s)$ onto joint torques. The first two of these mappings were trained on two different labeled masses ($m_1 = 0.005$ and $m_2 = 0.03$ for ball or $m_2 = 0.06$ for dumb-bell). The last mapping used the data from 12 different mass contexts (m increased from 0.005 to 0.06 in steps of 0.005); here, the contexts were unlabeled.

Our learning task requires a non-linear regression technique that provides error boundaries for each output value (required for the Kalman-filtering step). Among the possible techniques, we chose locally-weighted projection regression (LWPR)[2] (Vijayakumar et al., 2005) because it is fast – LWPR is O(N), where N is the number of data points; in contrast, the popular Gaussian process regression is O(N^3) if making no approximation (Rasmussen and Williams, 2006).

7 RESULTS

The results of the robot-simulation experiments are separated into ball task – inference and control under varying mass – and dumb-bell task – inference and control under multiple varying context variables.

[2]LWPR uses locally-weighted linear-regression. Data points are weighted according to Gaussian receptive fields. Our setup of the LWPR algorithm was as follows. For each output dimension, a separate LWPR regressor was computed. The widths of the Gaussian receptive fields were hand-tuned for each input dimension, and these widths were kept constant during the function approximation. Other learning parameters were kept at default values.

7.1 Inference and control under varying mass

For only one context variable, namely the mass of the manipulated object, both inference strategies (Sections 3 and 4) allowed to infer the unknown mass accurately (Figs. 7 to 10).

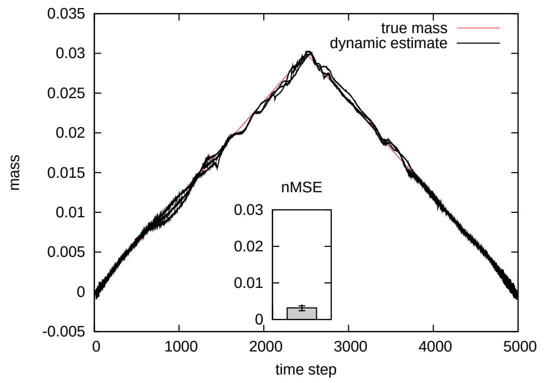

Figure 7: Inferring mass purely from dynamics. The inference results are shown for all three trajectories. The inset shows the normalized mean square error (nMSE) of the mass estimate. The error bars on the nMSE are min and max values averaged over an entire trajectory.

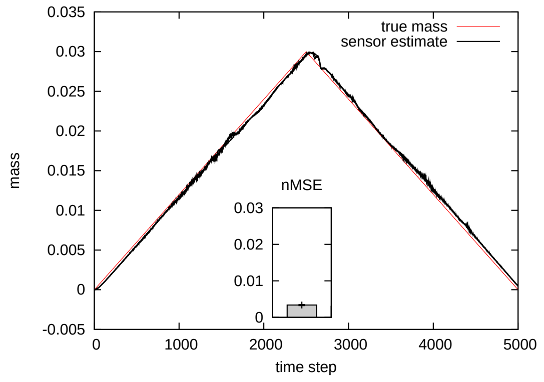

Figure 8: Inferring mass using tactile sensors. For details see Fig. 7.

Both types of mappings from state and acceleration either onto torques or onto sensors could be learned with low regression errors, which were of the same order (torques with m = 0.005: normalized mean square error (nMSE) = $2.9 * 10^{-4}$, m = 0.03: nMSE = $2.7 * 10^{-4}$; sensors with m =

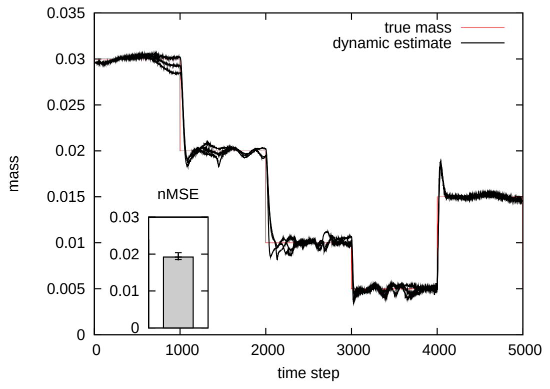

Figure 9: Inferring mass purely from dynamics. For details see Fig. 7.

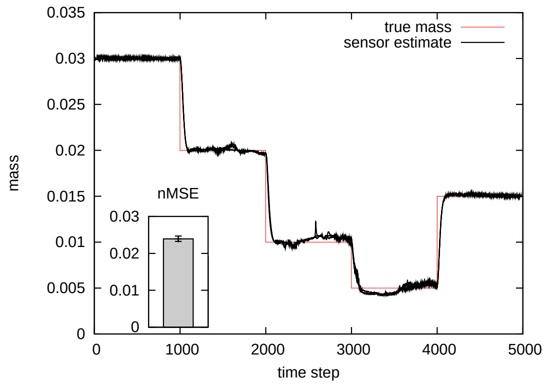

Figure 10: Inferring mass using tactile sensors. For details see Fig. 7.

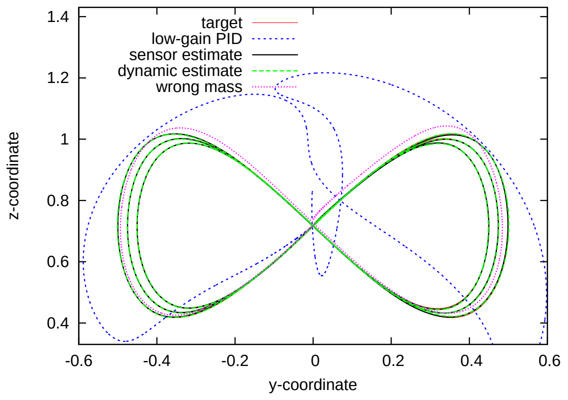

Figure 11: Following-the-eight task. The figure compares low-gain PID control (on large eight only) with the composite-controller that uses either the predicted mass estimate (from dynamics or sensors) or a wrong estimate ($m = 0.03$, on large eight only). The true mass decreased continuously from 0.03 to 0. The results for composite control based on the predicted mass overlap with the target for all three test curves.

0.005: nMSE $= 1.3 * 10^{-4}$, m = 0.03: nMSE $= 2.2 * 10^{-4}$). The error of the inferred mass was about the same for dynamics and sensor pathway. However, the variation between trials was lower in the sensor case.

Given the inferred mass via the torque and sensory pathways (Sections 3 and 4), our controller could provide accurate torques (Fig. 11). The results from both pathways overlap with the desired trajectory. As illustrated in the figure, a PID controller with a PD-gain as low as the gain of the error feed-back for the composite controller was insufficient for keeping the ball on the eight figure. The figure furthermore illustrates that without a correct mass estimate, tracking was impaired. Thus, correctly estimating the mass matters for this task.

7.2 Inference and control under multiple varying context

The inference of mass based on a single hidden variable failed if more variables varied (Fig. 12). We demonstrate this failure with the swinging dumb-bell, whose mass increased continuously during the trial. In the last part of the trial, when the dumb-bell was heavy and swung, the mass inference was worst. The wrong mass estimate further impaired the control of the robot arm (see Fig. 13).

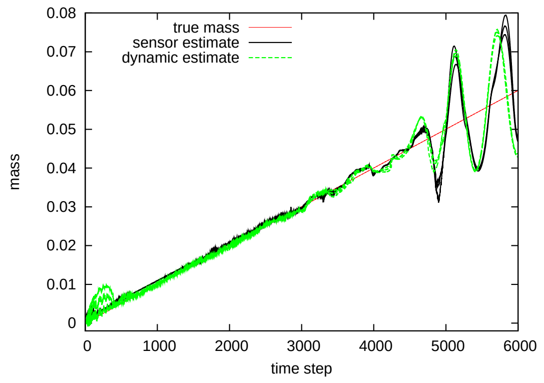

Figure 12: Inference of mass in the dumb-bell task.

During the last part of the movement, tracking was better with the direct mapping from $(\theta, \dot{\theta}, \ddot{\theta}, s)$ onto torques. For this mapping, the results still show some deviation from the target. However, we expect this error to reduce with more training data.

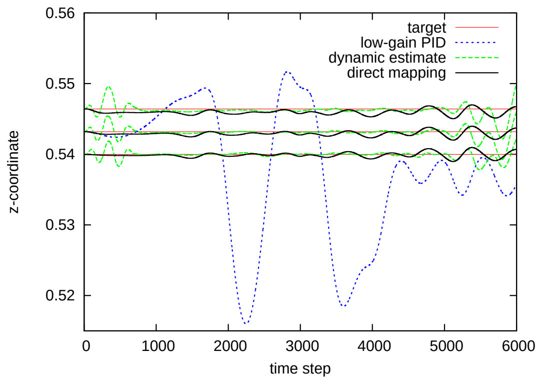

Figure 13: Swinging-the-dumb-bell task. The figure compares low-gain PID control (shown only on the middle trajectory) with the composite-controller that uses either the predicted mass estimate from dynamics or a direct mapping from $(\theta, \dot{\theta}, \ddot{\theta}, s)$ onto torques. For the estimate from sensors, the results were similar to the dynamics case and are omitted in the graph to improve legibility. The true mass increased continuously from 0 to 0.06.

8 CONCLUSIONS

We presented three strategies for adaptive motor control under continuously varying context. First, hidden inertia parameters of a manipulated object can be inferred from the dynamics and in turn used to predict control torques. Second, the hidden mass of an object can be inferred from the tactile forces exerted by the object. Third, correct control torques can be predicted by augmenting the input $(\theta, \dot{\theta}, \ddot{\theta})$ with tactile forces and by directly mapping this input onto the torques.

We demonstrated the first two strategies with object mass as the varying context. For more context variables, inferring the mass failed, and thus, the control torques were inaccurate. In principle, all inertia parameters could be inferred from the dynamics, but this inference requires modeling a 10-dimensional latent-variable space, which is unfeasible without extensive training data.

On the other hand, the direct mapping onto torques with sensors as additional input could predict accurate control torques under two varying context variables and in principle could cope with arbitrary changes of the manipulated object (including external forces). Further advantages of this strategy are its simplicity (it only requires function approximation), and for training, no labeled contexts are required.

In future work, we try to replicate these findings on a real robot arm with real tactile sensors. Real sensors might be more noisy compared to our simulated sensors; particularly, the interface between sensor and object is less well controlled.

ACKNOWLEDGMENTS

The authors are grateful to the German Aerospace Center (DLR) for providing the data of the Light-Weight Robot III and to Marc Toussaint and Djordje Mitrovic for their contribution to the robot-arm simulator. This work was funded under the SENSOPAC project. SENSOPAC is supported by the European Commission through the Sixth Framework Program for Research and Development up to 6 500 000 EUR (out of a total budget of 8 195 953.50 EUR); the SENSOPAC project addresses the “Information Society Technologies” thematic priority. G. P. was funded by the Greek State Scholarships Foundation.

REFERENCES

Haruno, M., Wolpert, D. M., and Kawato, M. (2001). Mosaic model for sensorimotor learning and control. *Neural Computation*, 13:2201–2220.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45.

Narendra, K. S. and Balakrishnan, J. (1997). Adaptive control using multiple models. *IEEE Transactions on Automatic Control*, 42(2):171–187.

Narendra, K. S. and Xiang, C. (2000). Adaptive control of discrete-time systems using multiple models. *IEEE Transactions on Automatic Control*, 45(9):1669–1686.

Petkos, G., Toussaint, M., and Vijayakumar, S. (2006). Learning multiple models of non-linear dynamics for control under varying contexts. In *Proceedings of the International Conference on Artificial Neural Networks*. Springer.

Petkos, G. and Vijayakumar, S. (2007). Context estimation and learning control through latent variable extraction: From discrete to continuous contexts. In *Proceedings of the International Conference on Robotics and Automation*. IEEE.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Sciavicco, L. and Siciliano, B. (2000). *Modelling and Control of Robot Manipulators*. Springer.

Vijayakumar, S., D’Souza, A., and Schaal, S. (2005). Incremental online learning in high dimensions. *Neural Computation*, 17:2602–2634.

Wolpert, D. M. and Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329.