A Two-Level Model of Anticipation-Based Motor Learning for Whole Body Motion

Camille Salaün, Vincent Padois, Olivier Sigaud

A Two-Level Model of Anticipation-Based Motor Learning for Whole Body Motion

Camille Salaün, Vincent Padois, and Olivier Sigaud

Université Pierre et Marie Curie - Paris6
Institut des Systèmes Intelligents et de Robotique, CNRS UMR 7222,
4 place Jussieu, F-75005 Paris, France
{Camille.Salaun,Vincent.Padois,Olivier.Sigaud}@upmc.fr

Abstract. We present a model of motor learning based on a combination of Operational Space Control and Optimal Control. Anticipatory processes are used both in the learning of the dynamics model of the system and in the coordination between both types of control. In order to illustrate the proposed model and associated control method, we apply these principles to the control of a simplified virtual humanoid performing a stand-up task starting from a crouching posture.

1 Introduction

Early in the history of motor control studies, Bernstein raised the following paradox: thanks to the redundancy of their motor system, human beings can realize a task with an infinity of ways. However, for a given task, they always reproduce the same kind of stereotypic motion. The problem consists in explaining where these regularities come from, given the diversity of possible solutions [1].

An attractive solution to this problem consist in stating that motor control optimises some criterion, thus the performed motion among the many possible ones is the optimal one with respect to that particular criterion. Several potential criteria have been proposed such as the *Minimum jerk* [2], the *Minimum torque-change* [3] and their variants (*Minimum motor command change* [4] and *Minimum commanded torque change* [5]). These criteria are all based on the idea that human motion must optimise *smoothness*, but a principled explanation of why motion should optimise smoothness is missing.

By contrast, the *Minimum end-point variance* criterion comes with a convincing explanation: natural movements must reach their target accurately, in spite of a neural noise that is proportional to the motor signal. Thus motor control must minimise the error on the terminal position of the controlled effector. Harris and Wolpert [6] showed that an optimal controller based on this principle can reproduce all motor invariants on which previous criteria were based, but also additional invariants such as Fitts' law [7] which relates accuracy to the speed of execution.

The direct consequence of this criterion is the *Minimum intervention principle* [8] that stipulates that the motor system activates muscles as few as possible to

G. Pezzulo et al. (Eds.): ABiALS 2008, LNAI 5499, pp. 229–246, 2009.

achieve its task, so as to minimise the motor signal related noise. As a result, the generated control only acts in the dimensions that are relevant with respect to the task and rejects the noise to the other dimensions. This fact is validated experimentally by [9] who showed that fluctuations are larger on the *Uncontrolled manifold* consisting of the irrelevant dimensions than on the relevant ones.

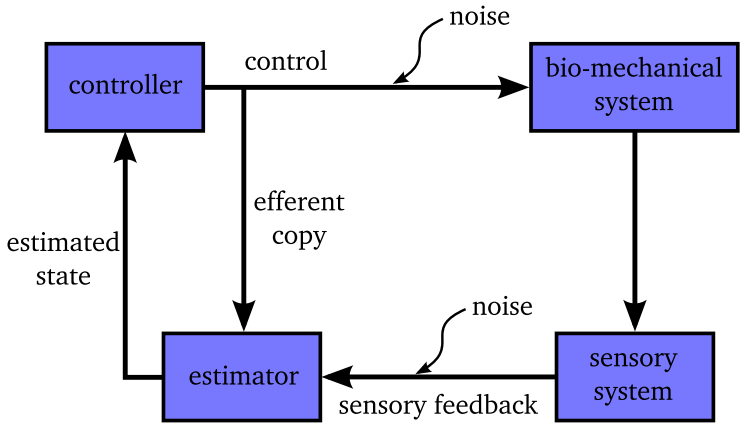


Fig. 1. A global view of SOFC: the controller must estimate the state of the system and maximise movement accuracy in the presence of sensory and motor noise

Based on these findings, the Stochastic Optimal Feedback Control (SOFC) method [10,11], illustrated in Fig. 1, and some variants (*Terminal Optimal Feedback Controller* [12,13,14] and *Task Optimisation in the Presence of Signal dependent noise* [15]) have received a wide agreement as good models of human motor control for elementary tasks for a single arm. However the computational cost of Optimal Control (OC) methods such as SOFC make them unsuitable to solve larger problems such as the synthesis of whole body motion. Thus an important issue in this domain consists in finding computationally less expensive approaches endowed with the same properties.

Another issue is concerned about learning and adapting to changing dynamic conditions. Several authors [16,17] propose a general framework in which human motor control is model-based, combines feedforward and feedback control processes [18] and calls upon optimisation processes as explained above. In this framework, motor adaptation results from learning the dynamics model of the system.

These principles were first implemented in *Modular Selection And Identification for Control* (MOSAIC) [19], a modular architecture where each module gets specialised to deal with particular dynamic circumstances. In MOSAIC, control learning is based on an handcrafted corrector that requires a priori knowledge

on the dynamics of the system. *Multiple Model-Based Reinforcement Learning* (MMRL) [20] solves this limitation by introducing a reinforcement learning process. Each module in MMRL calls upon an optimal control module, thus it still suffers from the computational cost of OC approaches.

One way to overcome this computational cost problem is suggested by Shadmehr's global view of the motor system [21]. From studies based on reaching and pointing movements, Shadmehr considers that motion generation can be decomposed into two parts: at the higher level, the central nervous system computes, in a visual frame of reference centred on the fixation point, a vector from the end-point effector to the target. This vector defines a trajectory in the operational visual space. Then this trajectory is converted into a muscular, low-level control of the joints angles by calling upon learned visuo-articular velocity and position mappings.

The most adequate robotics control tool to formalise Shadmehr's intuition consists in projection methods, such as Operational Space Control (OSC) [22]. These methods perform their computation in the so-called operational space, a space relative to the task, which is usually smaller than the joints space. They can thus be applied to large robotic systems [23]. They also give rise to a mathematically straightforward way of decoupling a set of tasks ranked by priority [24], but are sensitive to these priorities and cannot directly perform a global optimisation of a cost function associated to the motion of the system over time.

By contrast, optimisation methods, such as OC perform their computation in the joints space, but benefit from the easy definition of the constraints and performance criteria [25] of the tasks. However, the only way to express priorities among tasks within this framework consists in tuning the relative weights of the corresponding performance criteria.

In this contribution, we present an hierarchical model combining the assets of both control methods. Our model is consistent with Shadmehr's view in that a high level control loop computes a global trajectory in the operational space and then a low level control loop dynamically realizes it at the joints level. Besides, our model calls upon a velocity kinematics model and a dynamics model of the system, respectively. We show how these models can be learned from experience within an anticipatory process. Finally, the coordination of both control levels also results from an anticipatory process, the control set point being chosen as a point that is forward in time with respect to the current time. Our model is functional rather than neuromimetic: It focuses on the level of computational principles of human motor control and does not claim anything about the underlying neuroanatomy.

The contribution is organised as follows. Section 2 gives the technical background of each component of our model. The global structure of the controller itself is presented in section 3. In section 4, we describe an empirical study about the control of a simplified virtual humanoid performing a stand-up task starting from a crouching posture. We discuss the results in section 5.

2 Background

We present a model of human motor learning that combines the definition of tasks in the operational space, the use of OC at the joints level and experience-based, incremental model learning processes. An inverse velocity kinematics model is learned with the *Locally Weighted Projection Regression* (LWPR) non-linear function approximation algorithm [26] whereas a direct linearised model of the dynamics is learned with a classical Recursive Least Squares algorithm. Learning is performed on-line, during the control of the system. The different components of the resulting adaptive control architecture are described below.

2.1 Forward and Inverse Velocity Kinematics

The operational space or task space is the most natural space to describe the motion of the end effector of a system, whereas the joints space is the most natural space to control the system. The OSC approach consists in specifying the goal in the operational space and then using a linear transformation from the operational space to the joints space to control the system. Let $\boldsymbol{\xi}$ be an m-dimensional vector of operational coordinates used to describe the task (e.g. position and orientation of the end-point effector) and let $\boldsymbol{q}$ be an n-dimensional vector of generalised coordinates – *i.e.* the vector of parameters necessary to describe the configuration of the system without ambiguity. For systems with a fixed base, the generalised coordinates are often chosen as the joints angles. The relation $\boldsymbol{\xi} = f(\boldsymbol{q})$ is called *forward kinematics model* (FKM). It describes the operational coordinates as a function of the configuration of the system. Differentiating with respect to time, we obtain

$$\dot{\boldsymbol{\xi}} = J(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{1}$$

where $J(\boldsymbol{q}) = \dfrac{\partial f(\boldsymbol{q})}{\partial \boldsymbol{q}}$ is the Jacobian matrix of f.

To control the system, we need $\dot{\boldsymbol{q}}$ as a function of $\dot{\boldsymbol{\xi}}$. For a redundant system ($rank(J) < n$) for which the inversion of (1) has an infinity of solutions, one particular solution is:

$$\dot{\boldsymbol{q}} = J(\boldsymbol{q})^{\sharp}\dot{\boldsymbol{\xi}} \tag{2}$$

where $J(\boldsymbol{q})^{\sharp}$ is a weighted pseudoinverse of $J(\boldsymbol{q})$ [27].

Weighted pseudo-inverse of $J(\boldsymbol{q})$ are written:

$$J(q)^{\sharp} = W^{-1}J(\boldsymbol{q})^{T}\left[J(\boldsymbol{q})W^{-1}J(\boldsymbol{q})^{T}\right]^{-1}$$

where W is a symmetric and positive matrix of dimension n. This particular set of solutions to (1) minimises the Euclidean *W-weighted* norm $\sqrt{\dot{\boldsymbol{q}}^{T}W\dot{\boldsymbol{q}}}$ of the solution. In the case where $W = I_n$, this solution is called the Moore-Penrose or pseudoinverse of the Jacobian matrix and is noted J^{+}.

The general form of the solution can be written as:

$$\dot{\boldsymbol{q}} = J(\boldsymbol{q})^{\sharp}\dot{\boldsymbol{\xi}} + \left(I_n - J(\boldsymbol{q})^{\sharp}J(\boldsymbol{q})\right)\dot{\boldsymbol{q}}_0, \tag{3}$$

where $\left(I_n - J(\boldsymbol{q})^{\sharp} J(\boldsymbol{q})\right)$ is a projector onto the nullspace of $J(\boldsymbol{q})$ (*i.e.* the space of internal mobility with respect to the main task) and $\dot{\boldsymbol{q}}_0$ any vector of dimension n. The second term of the right-hand term of equation (3) gives access to the redundancy of the system in a hierarchical manner: any secondary task or constraint projected onto the nullspace of the Jacobian matrix is given less priority than the main task. It is achieved as long as it does not disturb this main task.

2.2 Operational Space Control

OSC consists in reaching a target $\boldsymbol{\xi_{ref}}$ from an initial position. This control method uses the inverse velocity kinematics model described by equation (2) as well as a proportional controller to compute the desired operational velocity:

$$\dot{\boldsymbol{\xi}}_{\boldsymbol{des}} = K_P \left(\boldsymbol{\xi}_{des} - \boldsymbol{\xi}\right), \tag{4}$$

where K_p is a positive definite matrix used as a proportional gain. This method is called *resolved rate motion control*.

Using the inverse velocity kinematics model described by equation (2), we have:

$$\dot{\boldsymbol{q}}_{des} = J(\boldsymbol{q})^{\sharp} \dot{\boldsymbol{\xi}}_{des}. \tag{5}$$

Thus, given a desired task velocity $\boldsymbol{\xi_{ref}}$ and knowing $J(\boldsymbol{q})^{\sharp}$, one can derive the desired joints velocities $\dot{\boldsymbol{\xi}}_{\boldsymbol{des}}$. If one wishes to specify the desired joints accelerations, one may use a progressive differentiation:

$$\ddot{\boldsymbol{q}}_{des} = \frac{\dot{\boldsymbol{q}}_{des} - \dot{\boldsymbol{q}}}{\Delta t} \tag{6}$$

Independently of the formalism used (Lagrange or Newton-Euler), the equations of motion of a fully-actuated, holonomic system in the contact-free case can be written:

$$\boldsymbol{\Gamma} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) \tag{7}$$

where M is the symmetric positive definite inertia matrix of the system, $\boldsymbol{b}$ is the vector of nonlinear effects modelling centrifugal and Coriolis forces, $\boldsymbol{g}$ is the gravity vector and $\boldsymbol{\Gamma}$ represents the torques applied to the system. This inverse dynamics model is used to compute, for a given state, the torque that must be applied to the system to get the desired joint accelerations:

$$\boldsymbol{\Gamma} = M(\boldsymbol{q})\ddot{\boldsymbol{q}}_{des} + \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) \tag{8}$$

where $\ddot{\boldsymbol{q}}_{des}$ is computed from $\dot{\boldsymbol{\xi}}_{\boldsymbol{des}}$ using equations (4), (5) and (6).

2.3 Linearisation of the Dynamics Model

It is possible to linearise the dynamics model around a state o:

$$F_{1_{\boldsymbol{o}}}\ddot{\boldsymbol{q}} + F_{2_{\boldsymbol{o}}}\dot{\boldsymbol{q}} + F_{3_{\boldsymbol{o}}}\boldsymbol{q} = \boldsymbol{\Gamma},$$

We can reformulate this as:

$$\begin{bmatrix}\dot{\boldsymbol{q}}\\ \ddot{\boldsymbol{q}}\end{bmatrix} = \begin{bmatrix}0 & Id\\ -F_{1_{\boldsymbol{o}}}^{-1}F_{3_{\boldsymbol{o}}} & -F_{1_{\boldsymbol{o}}}^{-1}F_{2_{\boldsymbol{o}}}\end{bmatrix}\begin{bmatrix}\boldsymbol{q}\\ \dot{\boldsymbol{q}}\end{bmatrix} + \begin{bmatrix}0\\ F_{1_{\boldsymbol{o}}}^{-1}\end{bmatrix}\boldsymbol{\Gamma}.$$

This formula is called state representation and can be rewritten $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$.

The state and action matrices A and B generally depend on the state. This will not be the case in the study hereafter, the dynamics model being linearised. In its discrete form, this equation is written:

$$\boldsymbol{x}_{k+1} = A'\boldsymbol{x}_k + B'\boldsymbol{u}_k \tag{9}$$

where $\boldsymbol{x}_k = \begin{bmatrix}\boldsymbol{q}_k\\ \dot{\boldsymbol{q}}_k\end{bmatrix}$ and $\boldsymbol{u}_k = \boldsymbol{\Gamma}_k$.

$\boldsymbol{x}_{k+1}$ is the future state whereas $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ are the current state and control input. A' and B' are the current state and action matrices.

In this study, we learn a model of the dynamics using the state space form of equation (9).

2.4 Optimal Control

Optimal control is a family of control methods that optimize a cost function over the achievement of a task. In the *Linear Quadratic Regulator* (LQR) framework in discrete time, the cost function (or optimality criterion) is written:

$$J = \sum_{k=0}^{\infty}(\boldsymbol{x_k}^T Q\boldsymbol{x_k} + \boldsymbol{u_k}^T R\boldsymbol{u_k}),$$

where Q and R are semi-definite positive and definite positive matrices associated to the cost on the state error and on the control input respectively.

The minimization of the cost function is obtained by solving the Riccati equation whose unknown is P:

$$A'^T PA' - \left(A'^T PB\right)\left(R + B'^T PB'\right)^{-1}\left(B'^T PA'\right) + Q = P,$$

where A' and B' are the state representation matrices defined above.

Solving this equation provides $L = \left(R + B'^T PB'\right)^{-1} B'^T PA'$, a constant state feedback matrix used to generate the following feedback controller:

$$\boldsymbol{u} = L\left(\boldsymbol{x}_{des} - \boldsymbol{x}\right),$$

where $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix}$ and where $\boldsymbol{x}_{des} = \begin{bmatrix} \boldsymbol{q}_{des} \\ \dot{\boldsymbol{q}}_{des} \end{bmatrix}$. $\dot{\boldsymbol{q}}_{des}$ is computed using the inverse velocity kinematics model described by equation (2) whereas $\boldsymbol{q}_{des}$ is extrapolated from the current configuration:

$$\boldsymbol{q}_{des} = \boldsymbol{q} + \dot{\boldsymbol{q}}_{des} \Delta t.$$

A schematic view of LQR in shown in Fig. 2.

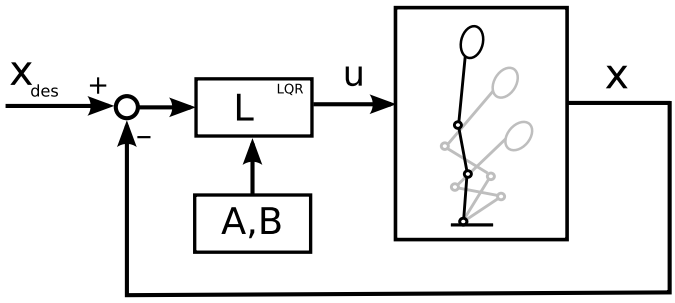


Fig. 2. LQR Feedback control: The feedback matrix L is computed from A' and B' using a discrete time Riccati equation

2.5 Model Learning

We want to learn models of our system incrementally and from experience. The nonlinear function we want to approximate can be written $\boldsymbol{y} = f(\boldsymbol{x})$. We use two techniques, one for obtaining a unique linear model β with $\boldsymbol{y} \approx \beta \boldsymbol{x}$ and the second for obtaining a nonlinear one $\hat{f}$ with $\boldsymbol{y} \approx \hat{f}(\boldsymbol{x})$ where $\hat{f}$ is a combination of linear models.

Least squares and recursive least squares. In the linear case, given some input-output data $X_k = [\boldsymbol{x_1}\ \boldsymbol{x_2}\ ...\ \boldsymbol{x_k}]$ and $Y_k = [\boldsymbol{y_1}\ \boldsymbol{y_2}\ ...\ \boldsymbol{y_k}]$ organised into a matrix, one can obtain the β matrix using the Least Squares approach: By multiplying $Y_k = \beta_k X_k$ by $X_k^T (X_k X_k^T)^{-1}$ on both sides, one gets $Y_k X_k^+ = \beta_k$ where $X_k^+ = X_k^T (X_k X_k^T)^{-1}$ and β_k represents the average for each term over β_1 ... β_k in the least squares sense.

This algorithm can be made incremental through a recursive formulation:

$$\beta_k = \beta_{k-1} + (\boldsymbol{y_k} - \beta_{k-1} \boldsymbol{x_k})\, \lambda$$

Thus, from this so-called Recursive Least Squares (RLS) method, we can get β_k for any k given the previous matrix and a forgetting factor λ that must be tuned.

LWPR. For most systems, linear models will not be accurate enough and must be replaced by nonlinear or piecewise-linear approximators such as neural networks, decision trees [28] or radial basis function networks [29].

The LWPR algorithm [30] is an incremental radial basis function approximator which provides accurate approximation in very large spaces in a $O(k)$ complexity, where k is the number of sample data. We use it here to learn the FKM of our robot. The algorithm uses a combination of linear models, which are valid on an elliptic zone of the input spaces and whose relative strengths are weighted by a gaussian. This space may evolve during training to match the training data. The domain of validity of each model is called a Receptive Field (RF). The prediction of an entire LWPR model on an input vector is the weighed sum of the results of all the active surrounding RFs. The RFs of a model are created when new input data are not part of any existing RF. Conversely, when a field overlaps another, some criterion can be used to delete it.

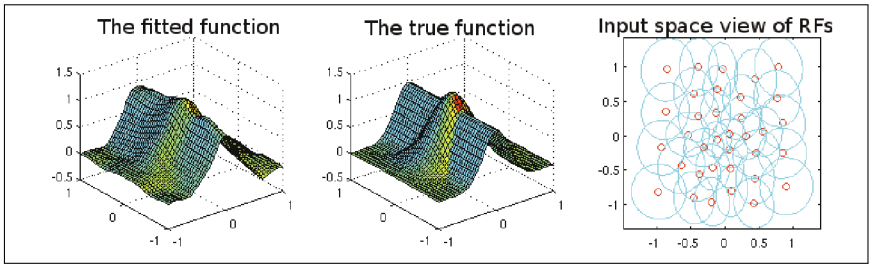


Fig. 3. Example of LWPR learning on a sample function

Each RF first projects the input vector on the most relevant dimensions for estimating the output vector by using Partial Least Squares [31]. During each update, the projector is updated and the algorithm checks whether increasing the complexity by adding another dimension to the input projection significantly reduces the estimation error. If it is so, it modifies the projector accordingly. The projected vector is then used in the n dimension linear model (n being the output dimension) to give the output of the RF.

During prediction with an input vector, the distance between the vector and the RF area is tested for activation and only the significant RFs are activated (see Fig. 3 for an example of RF repartition for function approximation).

The latest version of the algorithm also computes the first and second differential of the learned function with respect to each input dimension (Jacobian and Hessian matrices) when learning the model. LWPR can estimate them for any input value. This calculation is made easier by the fact that the model is a simple sum of multiple linear functions which are easily differentiated. We use this method to extract the Jacobian matrix from the learned model.

3 Our Motor Learning Model

Our global control scheme, illustrated in Fig. 4, consists of:

- a definition of the task in the operational space, resulting on the easy definition of tasks;
- learning the inverse velocity kinematics model with LWPR and a high level control law based on a proportional controller given by equation (4);
- learning the dynamics model with RLS and a low level optimal control law optimising a quadratic cost.

Our presentation below distinguishes three layers: high level control, low level control and the coupling between the two.

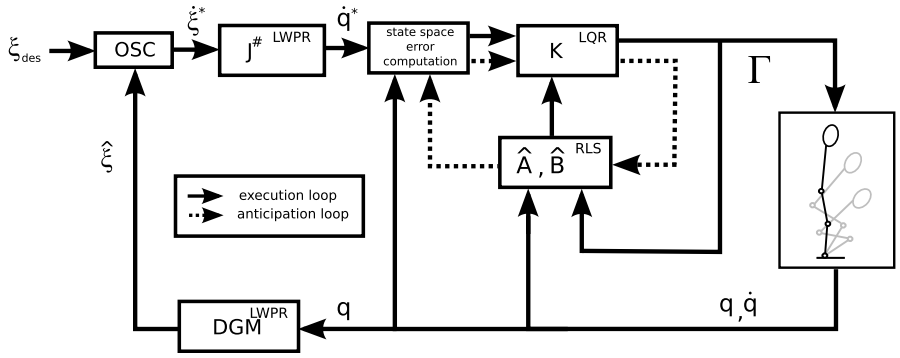


Fig. 4. Global control scheme

3.1 High Level Control

The operational space control part of our controller consists in the proportional controller given by equation (4) based on the difference between $\boldsymbol{\xi}_{des}$ and $\hat{\boldsymbol{\xi}}$. $\hat{\boldsymbol{\xi}}$ is the output of the FKM learned with LWPR, taking the joints positions as input and the operational space position as output.

After equation (4), we apply equation (5) to transform the error in the operational space into a desired velocity in the joints space. The inverse velocity kinematics model is learned with LWPR taking $\dot{\boldsymbol{\xi}}$ as input and $\dot{\boldsymbol{q}}$ as output, as shown in Fig. 5. This approach is similar to the one used in the work of D'Souza *et al.* in [32] who also choose to directly learn a specific inverse of the Jacobian matrix using the LWPR algorithm. Given the fact that there is an an infinity of inverses for the Jacobian matrix, the choice of a specific inverse leads to a loss of information regarding the redundant nature of the system. However, the learned

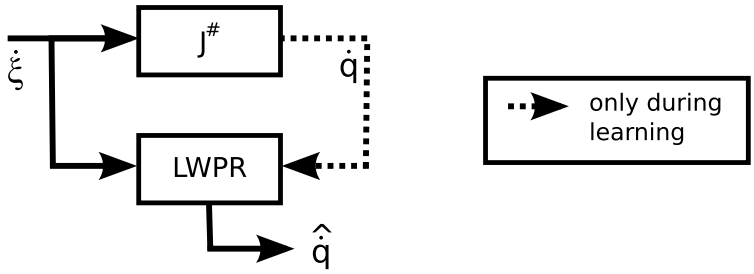


Fig. 5. Learning a generalised inverse with LWPR

inverse which we can expect to be a weighted pseudoinverse $J(\boldsymbol{q})^{\sharp}$, *i.e.* to have minimum norm properties, is the one that best fits the trajectories generated so far.

3.2 Low Level Control

At the low level, we use an LQR controller to track the desired joints velocities $\dot{\boldsymbol{q}}$ computed by the high level control. In the linearised dynamics model $\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k$, the state $\boldsymbol{x}_k$ is $[\boldsymbol{q}\ \dot{\boldsymbol{q}}]^T$, the action $\boldsymbol{u}_k$ corresponds to the vector of joints torques, and the matrices A and B are learned incrementally with the RLS algorithm as shown in Fig. 6, taking ($\boldsymbol{x}_k$,$\boldsymbol{u}_k$) as input and $\boldsymbol{x}_{k+1}$ as output. Note that, in order to bootstrap the learning process of RLS, rather than initialising A and B randomly, we rather store samples ($\boldsymbol{x}_k$,$\boldsymbol{u}_k$,$\boldsymbol{x}_{k+1}$) and apply a standard LS method until A and B are full rank.

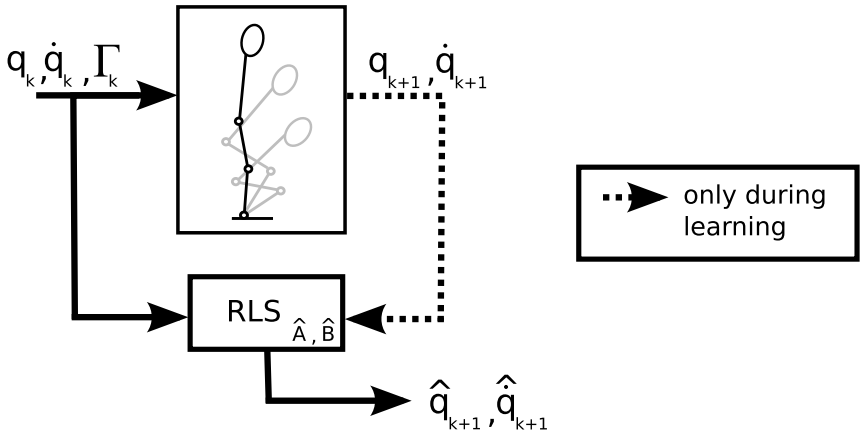


Fig. 6. Learning the dynamics model with RLS

3.3 Coupling Both Levels with Anticipation

If we take the current $\dot{\boldsymbol{q}}$ and $\boldsymbol{q}$ coming from the high level controller as reference, the tracking delay combined with the estimation errors results in an oscillatory behaviour. Thus, we couple the high level and low level control loops by anticipating the values of $\dot{\boldsymbol{q}}$ and $\boldsymbol{q}$ after some horizon H in the future and using these vectors in the low level control objective function to track them.

Algorithm 1. Anticipatory coupling algorithm.

1 $\boldsymbol{\xi}_0 = \boldsymbol{\xi}$;
2 **for** $i = 0$ *to* H **do**
3 $\quad \dot{\boldsymbol{\xi}}_{i+1} = K_p\left(\boldsymbol{\xi}_{des} - \boldsymbol{\xi}_i\right)$;
4 $\quad \dot{\boldsymbol{q}}_{i+1} = J^{\sharp}\dot{\boldsymbol{\xi}}_{i+1}$;
5 $\quad \boldsymbol{q}_{i+1} = \boldsymbol{q}_i + \dot{\boldsymbol{q}}_{i+1}\Delta t$;
6 $\quad \boldsymbol{\xi_{i+1}} = FKM\left(\boldsymbol{q}_{i+1}\right)$;
7 $\boldsymbol{u}_{0H} = LQR\left(A, B, [\boldsymbol{q}_H \dot{\boldsymbol{q}}_H]^T\right)$;

To perform such an anticipation, taking the current operational point $\boldsymbol{\xi}$ as initial value $\boldsymbol{\xi}_0$, we perform a loop on $\boldsymbol{\xi}_i$. First, we infer its derivative $\dot{\boldsymbol{\xi}}_i$ with a proportional controller (Algol. 1, line 3). This operational velocity can be translated into joints velocities through the weighted pseudoinverse of the Jacobian matrix (line 4). From these joints velocities, we estimate the next configuration (line 5) which, from the FKM, can be translated into an estimation $\boldsymbol{\xi_{i+1}}$ of the next operational position (line 6). We iterate this anticipation H times and this results in values for $\dot{\boldsymbol{q}}$ and $\boldsymbol{q}$ after H time steps. Finally, $\boldsymbol{u}_{0H}$ is the control input applied at moment 0 with horizon H.

Note that the way we estimate the next configuration does not take dynamics effects into account. We assume that our controller will reach the desired state at each step, which may not be true in practice if dynamics effects are not fully balanced by the low level controller.

By using the state representation $\boldsymbol{x}_i = \begin{bmatrix} \boldsymbol{q}_i \\ \dot{\boldsymbol{q}}_i \end{bmatrix}$, one can improve the estimation by using the $\boldsymbol{x}_{i+1} = A'\boldsymbol{x}_i + B'\boldsymbol{u}_i$ into the anticipation loop, but preliminary experiments have shown that this does not result in a significant difference in our context.

4 Empirical Study

4.1 A Simplified Virtual Humanoid System

To evaluate our control architecture, we work with a simplified 3 degrees of freedom (DOFs) mannequin model whose feet are fixed to the ground so as to alleviate the need to care about equilibrium. Thechoice of this simple model is intented to facilitate the presentation of our approach which we expect to be easily scalable to much higher dimension systems. The mannequin is simulated

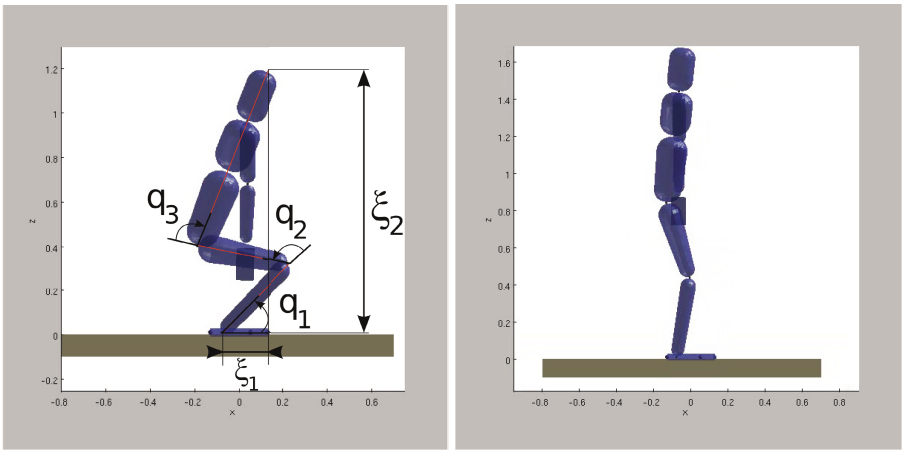


Fig. 7. Our 3 DOFs mannequin in its initial and final configurations

with Arboris [25], a Matlab simulator dedicated to the study of the dynamics of poly-articulated tree-like systems. Its parameters of our simplified mannequin model are extracted from anthropometric tables [33]. It is 1.74 meters tall and weighs 73 kg.

4.2 Task and Parameters

The task consists in making the mannequin stand up from a crouching posture, so that the vertex (upper extremity of the head) reaches 1.70 meters and stabilises over 1.65 meters, as shown in Fig. 7. Thus our operational task consists in reaching $\boldsymbol{\xi}_{des} = \begin{bmatrix}\boldsymbol{\xi}_x \; \boldsymbol{\xi}_y\end{bmatrix}^T = \begin{bmatrix}1.70 \; 0.05\end{bmatrix}^T$.

As for parameters, in (4), after testing diverse values, we take $K_P = 10$. Furthermore, a preliminary study of the low level control loop resulted in tuning the Q and R matrices of the LQR controller as follows:

$$Q = \begin{bmatrix} 0.4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.05 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.3 \end{bmatrix} \quad R = 10^{-4} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.3 Empirical Results

Fig.8 shows that, for a controller without an anticipation loop, the system is able to stand-up, but a vertical oscillation may be observed. With a short anticipation ($h = 2$ to 6), the system still reaches the operational target but is able to stay

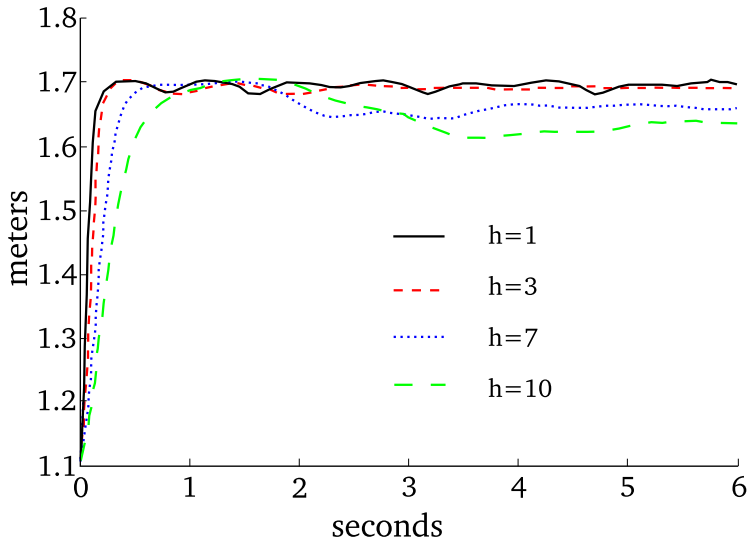
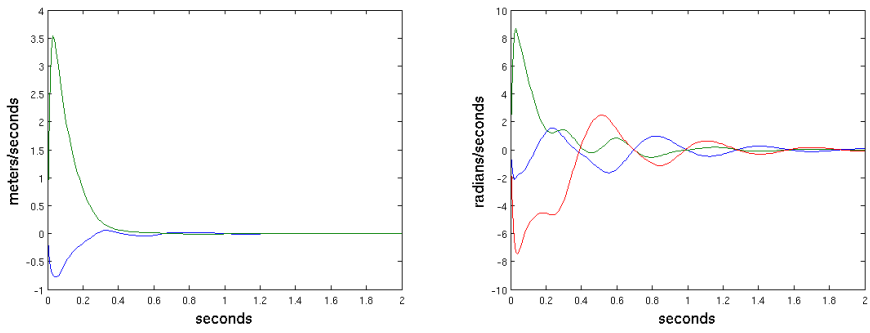


Fig. 8. Trajectory as a function of the anticipation horizon H

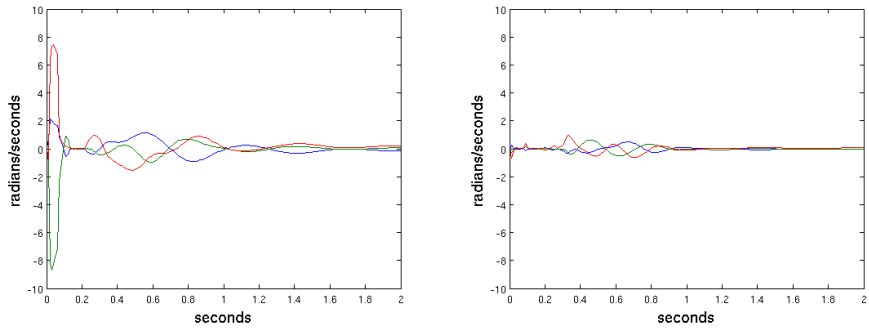


(a) *(b)*

Fig. 9. Learning the weighted pseudoinverse Jacobian matrix of the vertex of the mannequin. (a) Inputs are operational velocities ; (b) Output are joints velocities.

there for a longer time period and without oscillation. For a larger anticipation, the system is not able to maintain the desired position any more. Fig.9 shows an example of the input/output data used by LWPR to learn. After building the corresponding model, LWPR can predict the output $\hat{\boldsymbol{q}}$ from the input $\dot{\boldsymbol{\xi}}$.

Fig.10 shows that, if we perform only one learning epoch along the trajectory, the predicted output cannot be used. We must perform 20 learning epochs where each set of input/output data is presented ten times so as to get a satisfactory model of the weighted pseudoinverse of the Jacobian matrix.



(a) *(b)*

Fig. 10. Error in prediction of joints velocities after one (a) and twenty (b) learning epochs where each input/output data was presented ten times

Note that, once the system is stabilised, we must stop learning with LWPR. Indeed, if the system is stable, its output is constant and the learned models degenerate. The persistent activation principle used in adaptive control [34] can be used here to stop the learning process.

5 Discussion and Perspectives

5.1 Learning the Dynamics Model

Learning a linear model of the dynamics with RLS is a limited approach that can tackle easy problems such as the control of our 3 DOFs system, but as soon as the dynamics gets more complicated, the approximation error will get too large. Our most immediate future work will consist in using LWPR to learn a piecewise linear model of the dynamics. As explained in section 2.5, one can learn a nonlinear evolution of the A and B matrices with the state $[\boldsymbol{q}_k\ \dot{\boldsymbol{q}}_k]^T$ and the control input $[\boldsymbol{u}_k]$ as input at any instant and the next state $\left[\boldsymbol{q}_{k+1}, \dot{\boldsymbol{q}}_{k+1}\right]^T$ as output.

However, it is not trivial to exploit the explicit dynamics model learned by LWPR under the form of matrices as it was done with RLS. The output given by LWPR is a weighted sum of the outputs of linear models multiplied by a gaussian. The global dynamics model matrices cannot be reconstructed easily from this collection of linear models. Nevertheless, we are working on directly learning an inverse dynamics model in order to control our system with a computed torque [35] approach. Furthermore, we need to control the internal mobility (see equation 3) of the system when it is redundant, which boils down to learning a specific weighted pseudoinverse. This is a matter for future work.

5.2 Related Models

An important related model is the work of Mitrovic [36], who combines learning a dynamics model of a 6 DOFs arm with LWPR and performing *Iterative Linear Quadratic Gaussian* (iLQG) optimal control [37] based on that learned model. To our knowledge, Mitrovic *et al.* are the only authors who succeeded in using a dynamics model learned with LWPR so far. By contrast, N'Guyen-Tuong [38] performed such a learning process on more general systems in the context of a comparison with other learning approaches, but without using the model to control a system. Note that iLQG does not use the operational versus joints space distinction, thus it does not benefit from the dimensionality reduction properties of OSC and is limited to addressing systems with no more than 10 DOFs.

Another important related model is MMRL. As in MMRL, the low level controller of our model is based on LQR. So far, our model does not benefit from the modular specialisation mechanism that MMRL shares with MOSAIC. However, such a modularity property will come into play as soon as we will use LWPR at the dynamics level, since LWPR calls upon a collection of local linear models. But here again, the main originality of our approach results from the way we use OSC to specify the task in the operational space. To our knowledge, the only other model that is endowed with the same property is presented in [39] and is based on neural networks.

5.3 State Estimation

In our model, we considered the state of the system exactly known at any moment. In robots, if the FKM of the system is known and if fast enough sensors in servo-motors can give an accurate estimation of the configuration, one can get a good estimation of the current operational state of the system. But when the FKM is learned and inaccurate, state estimation must be used. To perform such an estimation, one must combine informations coming from several exteroceptive and proprioceptive sensors whose accuracy is varying under the environmental conditions and whose delay is generally not compatible with the constraints of feedback control-based motion. Even motor control itself is noisy, as we highlighted in the introduction. Thus maintaining an accurate estimation of the joints state of the musculoskeletal system is a central issue that we must address in the future.

6 Conclusion

We have presented a model of human motor learning that combines two levels. At the higher level, we specify a task in the operational space and use the OSC approach to derive a target trajectory that will be tracked in the joints space by the lower level. The transformation from the operational space to the joints space is performed with an inverse Jacobian matrix that learned with the LWPR algorithm. At the lower level, we use LQR control to compute the optimal joints

torques that realizes the trajectory specified in the operational space, based on a simple dynamics model learned with RLS. In our approach, optimal control does not lead to a globally optimal motion but rather to a piecewise optimal solution with respect to the learned system dynamics. In that respect, we can expect our method to be more easily scalable to large dimension systems and it would be of interest to study the best compromise between the computational complexity and the chosen optimal control horizon.

We illustrated this model on a stand-up task. One advantage of the combination of OSC with learning is its flexibility in the definition of the task. Indeed, if we change the end-point effector with respect to which the goal is expressed, or if we change the actuators with which the low-level control is performed, we should just need to learn again the corresponding dynamics to obtain a different controller. Validating this property is in our future work agenda.

Furthermore, we have shown that, due to the approximation error of the linear model learned with RLS, we need to stabilise the control architecture by tuning the anticipation horizon of the low level control loop with respect to the high level one. Thus anticipation appears in our control architecture both in model learning processes and in the coupling between the high and the low level control loops.

References

1. Bernstein, N.: The Co-ordination and Regulation of Movements. Pergamo, Oxford (1967)
2. Flash, T., Hogan, N.: The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model. Journal of Neuroscience 5(7), 1688–1703 (1985)
3. Uno, Y., Kawato, M., Suzuki, R.: Formation and control of optimal trajectory in human multijoint arm movement. Biological Cybernetics 61(2), 89–101 (1989)
4. Kawato, M.: Optimization and learning in neural networks for formation and control of coordinated movement. In: Attention and performance XIV (silver jubilee volume): synergies in experimental psychology, artificial intelligence, and cognitive neuroscience, pp. 821–849. MIT Press, Cambridge (1993)
5. Nakano, E., Imamizu, H., Osu, R., Uno, Y., Gomi, H., Yoshioka, T., Kawato, M.: Quantitative examinations of internal representations for arm trajectory planning: Minimum commanded torque change model. Journal of Neurophysiology 81(5), 2140–2155 (1999)
6. Harris, C.M., Wolpert, D.M.: Signal-dependent noise determines motor planning. Nature 394, 780–784 (1998)
7. Fitts, P.M.: The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology 47(6), 381–391 (1954)
8. Todorov, E., Jordan, M.: A minimal intervention principle for coordinated movement. In: NIPS, pp. 27–34 (2003)
9. Scholz, J.P., Schöner, G.: The uncontrolled manifold concept: identifying control variables for a functional task. Experimental Brain Research 126(3), 289–306 (1999)
10. Todorov, E., Jordan, M.I.: Optimal feedback control as a theory of motor coordination. Nature Neurosciences 5(11), 1226–1235 (2002)

11. Todorov, E.: Optimality principles in sensorimotor control. Nature Neurosciences 7(9), 907–915 (2004)
12. Guigon, E., Baraduc, P., Desmurget, M.: Computational motor control: Redudancy and invariance. Journal of Neurophysiology 97(1), 331–347 (2007)
13. Guigon, E., Baraduc, P., Desmurget, M.: Optimality, stochasticity and variability in motor behavior. Journal of Computational Neuroscience 24(1), 57–68 (2008)
14. Guigon, E., Baraduc, P., Desmurget, M.: Computational motor control: Feedback and accuracy. European Journal of Neuroscience 27(4), 1003–1016 (2008)
15. Miyamoto, H., Wolpert, D.M., Kawato, M.: Computing the optimal trajectory of arm movement: the TOPS (task optimization in the presence of signal-dependent noise) model. In: Biologically inspired robot behavior engineering, pp. 395–415. Physica-Verlag GmbH, Germany (2003)
16. Wolpert, D.M., Ghahramani, Z.: Computational principles of movement neuroscience. Nature Neuroscience 3, 1212–1217 (2000)
17. Wolpert, D.M., Kawato, M.: Multiple paired forward and inverse models for motor control. Neural Networks 11(7-8), 1317–1329 (1998)
18. Davidson, P.R., Wolpert, D.M.: Widespread access to predictive models in the motor system: a short review. Journal of Neural Engineering 2(3), S313–S319 (2005)
19. Haruno, M., Wolpert, D.M., Kawato, M.: MOSAIC model for sensorimotor learning and control. Neural Computation 13(10), 2201–2220 (2001)
20. Doya, K., Samejima, K., Katagiri, K., Kawato, M.: Multiple model-based reinforcement learning. Neural Computation 14(6), 1347–1369 (2002)
21. Shadmehr, R., Wise, S.: The Computational Neurobiology of Reaching and Pointing. MIT Press, Cambridge (2005)
22. Khatib, O.: A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE Journal of Robotics and Automation 3(1), 43–53 (1987)
23. Sentis, L., Khatib, O.: Control of free-floating humanoid robots through task prioritization. In: IEEE Conference on Robotics and Automation (ICRA), pp. 1718–1723 (April 2005)
24. Chiaverini, S.: Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. IEEE Transactions on Robotics and Automation 13(3), 398–410 (1997)
25. Barthlemy, S., Bidaud, P.: Stability measure of postural dynamic equilibrium based on residual radius. In: RoManSy 2008: 17th CISM-IFToMM Symposium on Robot Design, Dynamics and Control (2008)
26. Vijayakumar, S., DSouza, A., Schaal, S.: LWPR: A scalable method for incremental online learning in high dimensions. Technical report. Press of University of Edinburgh, Edinburgh (2005)
27. Golub, G.H., Van Loan, C.F.: Matrix Computations. Johns Hopkins University Press, Baltimore (1996)
28. Potts, D., Sammut, C.: Incremental learning of linear model trees. Machine Learning 61(1-3), 5–48 (2005)
29. Sun, G., Scassellati, B.: A fast and efficient model of learning to reach. International Journal of Humanoid Robotics 2(4), 391–414 (2005)
30. Vijayakumar, S., Schaal, S.: Local dimensionality reduction for locally weighted learning. In: IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp. 220–225 (1997)
31. Tenenhaus, M.: La régression PLS: théorie et pratique. Editions Technip (1998)

32. D'Souza, A., Vijayakumar, S., Schaal, S.: Learning inverse kinematics. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 1, pp. 298–303 (2001)
33. Wieber, P.B., Billet, F., Boissieux, L., Pissard-Gibollet, R.: The HuMAnS toolbox, a homogeneous framework for motion capture, analysis and simulation. In: Proceedings of the ninth ISB Symposium on 3D analysis of human movement, Valenciennes, France. Academic, San Diego (2006)
34. Sastry, S., Bodson, M., Bartram, J.F.: Adaptive control: Stability, convergence, and robustness. The Journal of the Acoustical Society of America 88, 588 (1990)
35. Siciliano, B., Khatib, O.: Springer Handbook of Robotics. Springer, New York (2007)
36. Mitrovic, D., Klanke, S., Vijayakumar, S.: Adaptive optimal control for redundantly actuated arms. In: Asada, M., Hallam, J.C.T., Meyer, J.-A., Tani, J. (eds.) SAB 2008. LNCS, vol. 5040, pp. 93–102. Springer, Heidelberg (2008)
37. Todorov, E., Li, W.: A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In: Proceedings of the American Control Conference, pp. 300–306 (2005)
38. Nguyen-Tuong, D., Peters, J., Seeger, M., Scholkopf, B.: Learning inverse dynamics: a comparison. Technical report, Max Planck Institute for Biological Cybernetics, Spemannstrae 38, 72076 Tubingen - Germany (2008)
39. Butz, M.V., Herbort, O., Hoffman, J.: Exploiting redundancy for flexible behavior: Unsupervised learning in a modular sensorimotor control architecture. Psychological Review 114(4), 1015–1046 (2007)