Learning for the Control of Dynamical Motion Systems

Pierre-François Marteau, Sylvie Gibet
VALORIA, University of Bretagne Sud, France
{pierre-francois.marteau, sylvie.gibet}@univ-ubs.fr

Abstract

This paper addresses the dynamic control of multi-joint systems based on learning of sensory-motor transformations. To avoid the dependency of the controllers to the analytical knowledge of the multi-joint system, a non parametric learning approach is developed which identifies non linear mappings between sensory signals and motor commands involved in control motor systems. The learning phase is handled through a General Regression Neural Network (GRNN) that implements a non parametric Nadarayan-Watson regression scheme and a set of local PIDs. The resulting dynamic sensory-motor controller (DSMC) is intensively tested within the scope of hand-arm reaching and tracking movements in a dynamical simulation environment. (DSMC) proves to be very effective and robust. Moreover, it reproduces kinematics behaviors close to captured hand-arm movements.

1. Introduction

For humans, moving skillfully and dynamically requires a long learning phase, acquired during infancy. After exploring the physical capabilities of the hand-arm system, both dynamically and kinematically for pointing, grasping or tracking tasks, in interaction with the environment, the performance exploits previous experiences and becomes more accurate, smooth and rapid for new tasks. Our approach is inspired by such a sensory-motor learning process. We propose a plausible biological model, which uses examples of previous performances, and is able to continuously select control variables, based on information coming from the sensorial receptors, given a specific task.

In our approach, the muscular-skeleton motion system, controlled through sensory feedback (visual or proprioceptive) involves nonlinear transformations that mix motor commands with sensory information. These transformations play a central role in the control of multi-articulated chain systems, whether they are biological or artificial. To cope with the need for plasticity (adaptation to changes), generic control performance (similar control principles for various kinds of mappings, various kinds of articulated chains or neuro-anatomical variability among individuals) and anticipation (predictive capability and optimization of movements), it is more or less accepted in the neuroscience community that these mappings are learned by biological organisms rather than pre-programmed. For the design of artificial control system, the biological plausibility of the control mechanisms involved is not really considered as an issue. Nevertheless, adaptive, predictive and generic capabilities of controlling components are indeed key characteristics that have been carefully addressed for a long time, in particular within the optimal control field.

In the scope of complex artificial system design, analytical equations that drive the dynamics and the kinematics of the motion system can be difficult to extract, and the corresponding solution to the set of differential equations fastidious to estimate. In particular, computational implementations of sensory-motor controllers require the complete or partial knowledge of transformations that are directly dependent on the multi-joint structure to control. Setting up control strategies for complex system control is consequently not a simple task. In this context, learning or identifying part of the control strategy from the observation of the system behavior is an appealing and efficient approach. The aim of this paper is to present a generic learning approach for the dynamical control of mechanical articulated systems, and more precisely hand-arm systems.

2. Related work

Previous work in learning motion control can be divided in two main classes, depending on the motivations: the first class concerns works which provide new insights into motor control. This kind of work may improve the understanding via simulation of hypothetical strategies that the Central Nervous System uses to control limb movements. The second

class concerns the design of artificial systems that mimic biological behaviors.

Within the first class of work, numerous approaches integrating learning mechanisms have been developed to control sensory-motor systems. Among them, several significant contributions highlight two main approaches: those which are looking for an *a priori* analogy with biological systems (identification of functions of the cerebellum) [1-3], and the others which are looking for an *a posteriori* analogy with biological systems [4-5].

In the second class of work, the problem of learning motion control is encompassed by the highly developed field of neural network control [6]. A typical "intelligent" motion controller tends to output the control signals directly from a neural network, or a similar device. Two distinct and competing approaches are available when facing the problem of learning non linear transformations (NLT) and in particular non linear mappings involved in multi-joint control systems: parametric learning (PL) and non parametric learning (NPL) (see [7] and [8] for a review of PL and NPL models with biological relevance arguments regarding internal sensory-motor maps). The fundamental difference between PL and NPL is that PL addresses the learning essentially globally while NPL addresses it much more locally. In other words, PL methods try to learn non linear transforms over their whole domain of validity. This means that if a change in the environment occurs locally, it will potentially affect the learning process everywhere in the definition domain of the transform. Conversely, NPL learns the properties of the transform in the neighborhood of each point of interest within the definition domain of the transform. Thus, a local update in the learning process does not affect the rest of the learned definition domain. Multi layer Perceptron [9-10] are instances of the PL class with synaptic weights as parameters, while Probabilistic Networks or General regression Neural networks [11-12] are instances of the NPL class.

Biological relevance can be found for the two kinds of approaches. Nevertheless, local characteristics of NPL is undoubtedly a great advantage when addressing incremental learning in variable environments, since the local modification resulting from any change does not affect the overall structure of the non linear transform already learned.

Our paper proposes a new learning scheme for the dynamical control of muscular-skeleton systems, based on the learning of inverse sensory-motor transformations. Our learning algorithm is applied to the control of an anthropomorphic hand-arm system.

3. Inverse sensory-motor control

The controller system refers to the process of defining a sensory-motor control policy for a muscular-skeleton system and a particular task goal. In our model, we assume that the system to move is composed of a skeleton dynamics part, labeled (*D*), and a skeleton kinematics part, labeled (*K*) (Fig. 1).

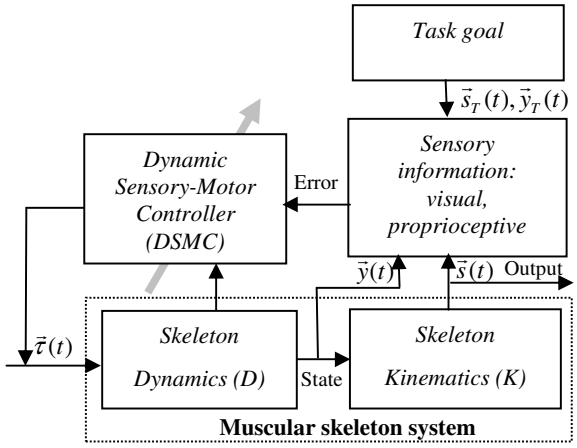


Figure 1. Inverse sensory-motor control

At each time, the muscular skeleton system can be characterized by its state vector $\vec{y}(t)$, represented by the joint coordinates and their derivatives $(\vec{q}, \dot{\vec{q}})$ ($\vec{q}$ being Euler angles or quaternion coordinates). Moreover, the sensory data are used in a feedback loop to iteratively compute the motor command: given the task goal $\vec{s}_T(t)$ or $\vec{y}_T(t)$, the dynamic sensory-motor controller (*DSMC*) generates from the error between the current values of $\vec{y}(t)$ and $\vec{s}(t)$ and the task goal a torque input $\vec{\tau}(t)$ applied to the direct (*DK*) system.

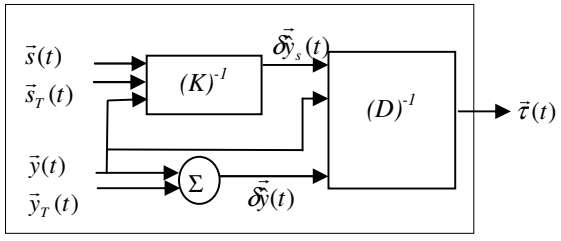


Figure 2. Dynamic sensory-motor controller

The controller itself is composed of two inverse transformations, as shown in figure 2, where $\vec{y}_T(t)$ is part of the command specified in the state space and $\vec{s}_T(t)$ part of the command specified in the output

space. The error signals measured between sensory outputs and task inputs are generally used as corrective information to update the torque command of the articulated system. The excess of degrees of freedom which characterizes the system (*DK*) makes the inversion of (*D*) and (*K*) a redundant problem since the same sensory outputs may be observed for numerous different torque commands and states of the system.

We briefly present hereinafter our solutions that implicitly inverse the (*D*) and (*K*) models and show how to assemble these solutions to build up a dynamic sensory-motor controller (*DSMC*).

3.1 Inversion of the skeleton dynamics *D*

When controlling mechanical articulated systems, we have to design control laws which compute torque commands for each joints. This control problem is inherently non linear, which means that much of linear control theory is not directly applicable. Nonetheless, one solution, classically used in robotics and computer animation, called Proportional Integrative Derivative law (PID), is issued from linear control theory. For each internal joint, each PID controller takes as inputs angular position of the joint and its derivative as well as the desired angular position, and computes the torque output required to produce the desired displacement of the joint as expressed by equation 1.

$$\tau(t) = K_p(\vec{q}_T - \vec{q}) + K_d(\dot{\vec{q}}_T - \dot{\vec{q}}) + K_i \int \vec{q}(t)dt \qquad (1)$$

where $\vec{q}$ is the angle of the joint, $\vec{q}_T$ is the desired angle, K_p, K_d and K_i are the respective proportional, derivative and integral gains. The effect of the PID controller is to eliminate large step changes in the errors, thus smoothing the simulated motion.

3.2. Inversion of the skeleton kinematics *K*

Numerous solutions exist to control sensory-motor systems (see references [1-5] for some approaches that exploit learning mechanisms).

Numerical approaches have been used in computer graphics as numerical iterative approaches to solve Inverse Kinematics (*IK*). (*IK*) can be regarded as a nonlinear optimization problem based on the minimization of a scalar potential function defined by:

$$E(\vec{y}) = \left(M(\vec{y}) - \vec{s}_T\right)^T \left(M(\vec{y}) - \vec{s}_T\right) \qquad (2)$$

where $M(\vec{y})$ denotes the forward kinematics mapping from the state space to the observation space and $\vec{s}_T$ denotes the desired sensory output expressed in the output task space.

In previous works we developed a gradient-based controller in a sensory-motor closed-loop transformation which integrates neurophysiologic elements. This model has proved to control articulated chains and produce motion that globally respects human motion laws [15]. To implement such a model, all coefficients of the Jacobian matrix $J(t)$ have to be known for all values of the state vector. These coefficients directly depend on the structure of the articulated chain to control. Furthermore, for any articulated chain, a specific Jacobian matrix has to be calculated. One solution to overcome such limitation is to introduce a learning scheme, a functionality that most of biological systems have ingenuously implemented.

A first learning scheme of the inverse kinematics transformation was proposed in [16] and [17]. We extend here the learning to the inverse dynamics, by adding a set of distributed inverse dynamics modules associated to inverse kinematics modules in which the mappings are learned.

4. Learning kinematics using General Regression Neural Networks(GRNN)

GRNN or "General Regression Neural Networks" have been proposed by Donald Specht [11-12]. They are relevant to Nadaraya-Watson kernel regression method, or Parzen window methods [18-19]. Definitions and assumptions behind the derivation of the Nadaraya-Watson estimate are detailed briefly hereinafter.

Let X be an m-dimensional random variable in (R^m, B_m) and Y an n-dimensional dependent random variable in (R^n, B_n) such that $Y = f(X)$, where B_m and B_n are the borel σ-algebra over R^m and R^n respectively. Let $\varphi(X,Y)$ be the joint continuous density function. Assuming that $\vec{x}$ and $\vec{y}$ are measured values for X and Y respectively, the expected value of Y given $\vec{x}$ (the regression of Y upon $\vec{x}$) is given as:

$$E(Y/\vec{x}) = \frac{\int_{-\infty}^{+\infty} Y.\varphi(Y,\vec{x}).dY}{\int_{-\infty}^{+\infty} \varphi(Y,\vec{x}).dY} \qquad (3)$$

Let K_m be a probability density function on R^m. We assume that K_m satisfies the following conditions:

K_m is continuous, symmetric, bounded.

Assuming that the underlying density is continuous and smooth enough (first derivatives of φ evaluated at any $\vec{x}$ are small), and based on a set of p observation samples $\{(\vec{x}_i, \vec{y}_i)\}_{i\in\{1,\dots,p\}}$, the joint probability density estimator $\hat{\varphi}(\vec{x}, \vec{y})$ using the non parametric Parzen's window method can be formulated as follows:

$$\hat{\varphi}(\vec{x}, \vec{y}) = \frac{1}{\alpha}\sum_{i=1}^{p} K_m\left(\frac{-(\vec{x}-\vec{x}_i)^T W_x(\vec{x}-\vec{x}_i)}{2(\sigma_i)^2}\right) K_m\left(\frac{-(\vec{y}-\vec{y}_i)^T(\vec{y}-\vec{y}_i)}{2(\sigma_i)^2}\right) \quad (4)$$

where α is a normalizing factor, W_x a positive diagonal matrix used as weighting coordinates of vector $\vec{x}$, and σ_i the local "bandwidth" of the kernel K_m centered on sample $(\vec{x}_i, \vec{y}_i)$. In general a Gaussian kernel is chosen such that K_m is identified to the exponential function.

Substituting equation (4) into equation (3) we get:

$$\hat{E}(Y/\vec{x}) = \frac{\sum_{i=1}^{p} \vec{y}_i . K_m\left(\frac{-(\vec{x}-\vec{x}_i)^T W_x(\vec{x}-\vec{x}_i)}{2(\sigma_i)^2}\right)}{\sum_{i=1}^{p} K_m\left(\frac{-(\vec{x}-\vec{x}_i)^T W_x(\vec{x}-\vec{x}_i)}{2(\sigma_i)^2}\right)} \quad (5)$$

or:

$$\hat{E}(Y/\vec{x}) = \frac{\sum_{i=1}^{p} \vec{y}_i . K_m(\vec{x}_i, \vec{x})}{\sum_{i=1}^{p} K_m(\vec{x}_i, \vec{x})} = \frac{\sum_{i=1}^{p} \vec{y}_i . w_i}{\sum_{i=1}^{p} w_i} \quad (6)$$

with $$w_i = K_m\left(\frac{-(\vec{x}-\vec{x}_i)^T W_x(\vec{x}-\vec{x}_i)}{2(\sigma_i)^2}\right) \quad (7)$$

Assumptions:

(A1) The $(\vec{x}_i, \vec{y}_i)$ are i.i.d. (This assumption will change later)

(A2) K_m is a symmetric function around zero with the following properties.

$$\int K_m(\psi) d\psi = 1$$

$$\int \psi^2 K_m(\psi) d\psi = \mu_2 \neq 0$$

$$\int {K_m}^2(\psi) d\psi < \infty$$

(A3) $\frac{\partial^2 \hat{\varphi}(x,y)}{\partial^2 x_k}$ exists for all k, is continuous, and is bounded in a neighborhood around x (and thus we can evaluate the bias and variance of $\hat{\varphi}$)

(A4) for all i, $\sigma_i \to 0 \quad as \quad p \to \infty$

(A5) for all i, $p.\sigma_i \to \infty \quad as \quad p \to \infty$

Under assumptions (A1) to (A5), the consistency of the kernel estimator is established, as the estimator $\hat{E}(Y/\vec{x})$ tends in probability towards $E(Y/\vec{x})$.

4.1. GRNN and the Nadaraya-Watson estimate

GRNN is a normalized Radial Basis Function (RBF) network for which a hidden unit is centered at every training sample. The RBF units of GRNN architecture are usually characterized by Gaussian kernels. The hidden-to-output weights are identified to the target values, so that the output is a weighted average of the target values of the training samples close to the given input case. The only parameters of the networks are the widths of the kernels associated to the RBF units. These widths (often a single width is used) are called "smoothing parameters" or "bandwidths" [18] [19]. They are usually chosen by cross-validation or by ad-hoc methods not well-described. GRNN is a universal approximator for smooth functions, so it should be able to solve any smooth function-approximation problem provided enough data is given. The main drawback of GRNN is that, like kernel methods in general, it suffers badly from the lack of learning data.

4.2. Learning SMC maps using GRNN

To estimate the normalized gradient of the error, the following map f is defined:

$$\delta\vec{y}_s = f(\vec{y}, \delta\vec{s}) \quad (8)$$

Where $\delta\vec{s}$ is the 3D directional vector towards the task $\vec{s}_T$ specified in the sensory space, $\vec{y}$ the vector of the state variable. $\delta\vec{y}_s$ is the estimated normalized modification vector within the state space that minimizes the error between the current output $\vec{s}$ and the task specification $\vec{s}_T$. Following GRNN memory based approach, the calculation of the map f is approximated through a variable Gaussian kernel density estimator as explained below.

Given a set p of learning samples, $\{(y_i, \delta y_i, \delta s_i)\}_{i=1\dots p}$, the state update $\delta\vec{y}_s$ that minimizes the error signal calculated from a current state $\vec{y}$ and a 3D normalized directional vector $\delta\vec{s}$ is estimated as the conditional expectation of $\delta\vec{y}$ given $\vec{\xi}$:

$$\delta\vec{y}_s = \hat{E}(\delta\vec{y}/\vec{\xi}) = \frac{\sum_{i=1}^{p} \delta\vec{y}_i . K(\vec{\xi}_i, \vec{\xi})}{C} \quad (9)$$

where $\vec{\xi} = [\vec{y}, \delta\vec{s}]^T$, $\vec{\xi}_i = [\vec{y}_i, \delta\vec{s}_i]^T$, C is a normalizing factor, and K a variable Gaussian kernel:

$$K(\vec{\xi}_i, \vec{\xi}) \approx \exp\left(-\frac{(\vec{\xi}-\vec{\xi}_i) W (\vec{\xi}-\vec{\xi}_i)^T}{\sigma}\right) \quad (10)$$

W is a weighting diagonal matrix used to balance the weighting of sensory information $\delta\vec{s}$ with state information $\vec{y}$, σ is a parameter that scales the local density, both in the state space and in the sensory space: if the density is low, σ is increased and conversely, if the density is high, σ is lowered.

4.3. Naive GRNN learning algorithm

σ is selected empirically, since an optimum value cannot be determined from a set of observations.

- Initialisation: select a small value ε, an integer value p and set i to *0* (ε can be a function of p).
- Select randomly a state vector $\vec{y}$, position the multi-joint system according to $\vec{y}$, and observe the corresponding sensory outputs $\vec{s}$.
- Select a small normalized change $\delta\vec{y}$, position the multi-joint system according to $(\vec{y}+\delta\vec{y})$, and observe the change in sensory outputs $\delta\vec{s}$.
- Calculate $\delta\vec{y}$ using $\vec{\xi}=[\vec{y},\delta\vec{s}]^T$ according to equations (9) and (10).

If $\left\|\delta\vec{y}-\delta\vec{y}\right\|>\varepsilon$, save the association $([\vec{y},\delta\vec{s}],\delta\vec{y})$ as a new learning sample $(\vec{\xi}_i,\delta\vec{y}_i)$, create a corresponding neuron and increment i.

- If $i<p$, loop in 1), stop otherwise

4.4. Implementation issues

When estimating the expectation of the state update $\delta\vec{y}$ given $\vec{\xi}$, the computations of distances in the d-dimensional $\vec{\xi}$ space are required (d = dimension of the state space + dimension of the sensory space). When summing Gaussian kernels (eq. (9) and (10)), only the $\vec{\xi}_i$ vectors belonging to the neighborhood of $\vec{\xi}$ are retained. To speed up the computation process, a kd-tree [14] for identifying neighborhoods in logarithmic time with p can be advantageously used. (The kd-tree representation of the stored data leads to reconsider the architecture of GRNN to implement similar neighborhood search).

5. Results

The learning approach is applied on a simulated mechanical system composed of two arms and two hands, submitted to successive reaching tasks. The mechanical systems are modeled with ODE (Open Dynamic Engine) [20], with a 3D custom rendering (see Fig. 4.b for the visualization of the 3D character). Each arm is composed of three joints with six degrees of freedom, and each finger is composed of three joints with four degrees of freedom.

The arms are controlled by (*DSMC*) controllers, where the mapping between $\vec{y}$ and $\vec{s}$ is learned on the basis of a gradient descent strategy. The learning processes are carried out for increasing values of the number of learning samples p. For each process, a thousand of 3D spatial target positions and initial conditions have been selected randomly to test the correctness of the learning process. For these *1000* conditions, the error rate (number of cases where the arm is not able to reach the target) is calculated.

The experimental settings for this test are the following. A target is considered to be reached when the residual distance between the arm end-point and the target is below 1% of the total length of the extended hand-arm chain. The size of σ is selected such that at least *40* neighbors can be provided to evaluate $\delta\vec{y}$.

The results of this test are reported in Figure 3. For about *60000* learning samples, the map f is apparently well modeled, since the residual error rate is low (about 0.5%) and very few improvements are gained when increasing p.

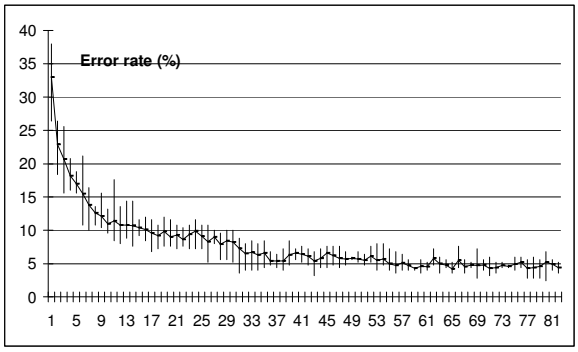
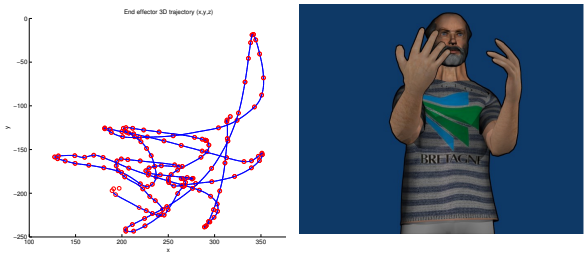


Figure 3. Error rate as a function of the number of learning samples *p*.

The fact that p can be chosen very low while maintaining good performances is a major result. Generally, for estimating a multivariate function with *9* variables (e.g. 6 degrees of freedom and 3D coordinates), a kernel density estimator requires above *500,000* samples adequately selected. In our case, *p=80,000* seems to be sufficient for the considered task. One reasonable explanation is that the sensory-motor loop performs a time average over successive gradient estimate values which compensates small errors due to the coarse estimation. A rough gradient mapping estimation is consequently quite accurate for the reaching task considered in our experiments.

After the learning phase, a simulation process was carried out, for a tracking task which consisted in following discrete targets extracted from a motion capture hand trajectory, according to an adaptive sub-sampling algorithm [21] (see figure 4 a). Furthermore, the simulation of *(DSMC)* is linked to this tracking task, and applied to a virtual character (see figure 4 b).



a) b)

Figure 4. a) Trajectory of the human wrist in the Cartesian space with the localization of the targets; b) Simulation by a virtual character for a tracking task

For this hand-tracking task, the resulting hand trajectories obtained through dynamical simulation can be superimposed with the captured trajectories, as illustrated in figure 4 a).

6. Conclusion

In this paper, we proposed a Dynamical Sensory-motor controller *(DSMC)* for controlling a dynamical hand-arm system. The controller combines both the inversion of the kinematics model, from the learning of sensory-motor mappings, and the inversion of the dynamical system using classical PID controllers. The learning of sensory-motor mappings was performed with non parametric learning approaches (GRNN), based on a variable kernel density estimator and the use of a kd-tree architecture to simulate neuron activation according to a near neighbor search. Despite the apparent high memory requirement needed by this kind of estimator, the proposed learning scheme behaves properly when used to control articulated systems with six degrees of freedom simulated in a dynamical environment. This result is obtained even if the number of learning samples is reasonably low.

7. References

[1] Kawato M., Maeda Y., Uno Y., Suzuki R.. Trajectory Formation of Arm Movement by Cascade Neural Network Model Based on Minimum Torque Criterion. Biological Cybernetics, vol. 62, 1990, pp. 275-288.

[2]Wolpert D.M., Miall R.C., Kawato M. Internal models in the cerebellum. *Trends in Cognitive Science*, vol. 2, n°9, 1998, pp. 338-347.

[3] Spoelstra J., Schweighofer N., Arbib M.A. Cerebellar learning of accurate predictive control for fast reaching movements. *Biological Cybernetics*, 82, 2000, pp. 321-333.

[4] Bullock D., Grossberg S., Guenther F.H. A Self-Organizing Neural Model of Motor Equivalent Reaching and Tool Use by a Multijoint Arm. Journal of Cognitive Neuroscience, vol. 54, 1993, pp. 408-435.

[5] Jordan M.I. Computational motor control. In M. S. Gazzaniga (Eds.), The cognitive neurosciences. Cambridge, MA: MIT Press, 1995 pp. 587-609.

[6] Werbos P.J. An overview of neural networks for control. *IEEE Control Systems Magazine*, January 1991.

[7] Duda, R. O., & Hart, P. E. Pattern classification and scene analysis. New York, NY: Wiley, 1973.

[8] Schaal, S. (in press). Nonparametric regression for learning nonlinear transformations. In: Ritter et al. eds. Prerational Intelligence in Strategies, High-Level Processes and Collective Behavior. Kluwer Academic.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Representation by Back-Propagating Errors. Nature 323:533-536, 1986.

[10] Bishop, C.M. Neural Networks for Pattern Recognition, Oxford University Press, Oxford, UK, 1995.

[11] D.F. Specht, A General Regression Neural Network IEEE Trans. Neural Networks, Vol.2, No.6, p568-576, 1991.

[12] D.F. Specht, Probabilistic Neural Networks, Neural Networks, 3, 1990, 109-118.

[13] Churchland, P. S. and SeSejnowski, T. J, The computational brain, MA:MIT Press, 1992.

[14] Friedman, J.H., Bentley J.L. and Finkel R.A., "An algorithm for finding best matches in logarithmic expected time", *ACM Trans. Math. Software*, 3, 209-226, 1977.

[15] Gibet S. and Marteau P.F., A Self-Organized Model for the Control, Planning and Learning of Nonlinear Multi-Dimensional Systems Using a Sensory Feedback, Journal of Applied Intelligence, Vol.4, 1994, pp. 337-349.

[16] Marteau P.F., Gibet S., Juliard F., Non Parametric Learning of Sensori Motor Maps. 5th WSES/IEEE Int. Conf. on Neural, Fuzzy and Evolutionary Computation, Rethymnon, Crete, 2001.

[17] Gibet S., Marteau P.F. Expressive Gesture Animation Based on Non Parametric Learning of Sensory-Motor Models, CASA 2003, Computer Animation and Social Agents, 7-9 mai 2003.

[18] Watson, G. S. Smooth regression analysis. Sankhya, Series A, 26, 359-372, 1964

[19] Nadaraya, E. A. (1964). On estimating regression. Theory Probab. Applic., 10, 186-190.

[20] Open Dynamics Engine, 2000-2003 Russell Smith. http://opende.sourceforge.net/

[21] P.F. Marteau, S. Gibet. Adaptive Sampling of Motion Trajectories for Discrete Task-Based Analysis and Synthesis of Gesture. In Gesture in Human-Computer Interaction and Simulation, 6th GW, Revised Selected Papers, LNCS, Volume 3881, pp.168-171, 2006.