Discriminative and Adaptive Imitation in Uni-Manual and Bi-Manual Tasks ⋆

Aude G. Billard, Sylvain Calinon, Florent Guenter

Autonomous Systems Laboratory 3 (ASL3)
School of Engineering, EPFL
Swiss Institute of Technology Lausanne EPFL
EPFL-STI-I2S-ASL3, Station 9, 1015 Lausanne, Switzerland
Corresponding Author: aude.billard@epfl.ch

Abstract

This paper addresses the problems of *what to imitate* and *how to imitate* in simple uni- and bi-manual manipulatory tasks. To solve the *what to imitate* issue, we use a probabilistic method, based on Hidden Markov Models, for extracting the relative importance of reproducing either the gesture or the specific hand path in a given task. This allows us to determine a metric of imitation performance. To solve the *how to imitate* issue, we compute the trajectory that optimizes the metric, given a set of robot's body constraints. We validate the methods in a series of experiments, where a human demonstrator teaches through kinesthetic a humanoid robot how to manipulate simple objects.

Key words: Discriminative Imitation, Learning, Humanoid Robots, Kinesthetic Teaching, HMM, Lagrange Optimization

1 Introduction

Recent advances in Robot Programming by Demonstration RbD, also referred to as *Learning by Imitation*, have identified a number of key issues that need to be solved for ensuring a generic approach to transferring skills across various agents and situations [17, 18]. These have been formulated as a set of generic questions, namely *what to imitate*, *how to imitate*, *when to imitate* and *who to imitate*. These questions were formulated in response to the large

⋆ Status: Accepted, To Appear in Robotics & Autonomous Systems Journal (Dec. 2005).

body of work in RbD that emphasized ad-hoc solutions to sequencing and decomposing complex tasks into *known* sets of actions performable by both the demonstrator and the imitator, see, e.g. [4, 12, 15, 23, 26, 27]. In contrast to these other works,the above four questions and their solutions aim at being generic in the sense of making no assumptions on the type of skills that may be transmitted. The drawback of such a generic approach is that it has yet to show how the methods will scale up, from acquiring basic skills to acquiring complex sequences of those.

In our prior work, we have addressed the *what to imitate* question, by developing a general architecture to extract the relevant features of a given task. The methods relied on computing the statistical variability of each element of the task, see [9, 6]. In this paper, we present an extension of this work to address the "how to imitate" problem for the control of uni-manual and bi-manual manipulation of objects, using a pair of 4 degrees of freedom robot arms. This leads us to tackling more generic issues of motor control, namely that of optimizing the arm controller given specific constraints. Specifically, we extend the pseudo-inverse optimization method for solving the inverse kinematics, so as to determine the optimal imitation strategy, i.e. the strategy that satisfies best all the constraints of a given task.

The issue of "how to imitate", also referred to as the *correspondence problem* [17], were first addressed very generically in a number of studies with simulated abstract agents acting in a Markov world [1, 5]. More recent work by [2, 14], also, considered the application of such a system for controlling a 2 degrees of freedom robotic arm. The solution to the correspondence problem in the latter works was, however, constrained to a particular arm and did not provide a general solution for robotic arms with an arbitrary number of degrees of freedom. Moreover, in each case, the metric for the task was given. Here, we present a method that, first, discovers the task metric and, second, extends the classical inverse kinematics solution to solve generically the correspondence problem for an arbitrary robotic arm.

Next, we illustrate the key issues at stake and present briefly the approach we take in this paper to solving those.

1.1 What-to-imitate and How-to-imitate

When learning a new task by imitation, the robot must first determine what are the relevant features of the task to imitate ("what to imitate"), and, second how to adapt its own motor program to produce an optimal imitation ("how to imitate"). Figures 1 and 2 illustrate these issues in a simple uni-manual task. Let us first consider the problem of determining the relevant features of

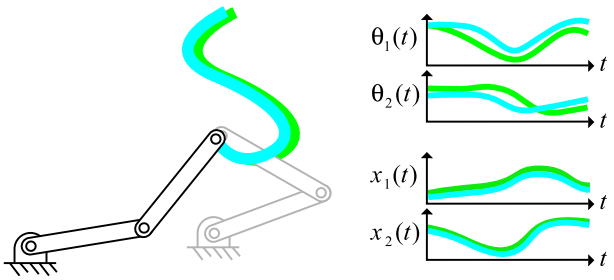

Fig. 1. Illustrative schema of the *what-to-imitate* issue. A 2 DOFs manipulator arm produces two demonstrations of a given task, namely drawing of an S figure, starting with different joint configurations. The path of the end effector given by $\mathbf{x} = \{x_1, x_2\}$ is invariant, while the joint trajectories $\boldsymbol{\theta} = \{\theta_1, \theta_2\}$ vary importantly.

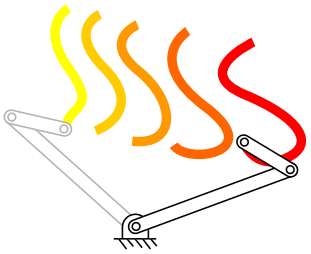

Fig. 2. Illustrative schema of the *how-to-imitate* issue. The manipulator of the imitator generates different alternatives to reproducing the task demonstrated in Figure 1 from satisfying purely the joint trajectories (left) to satisfying only the hand path (right).

a task and their relative importance.

What to imitate

Consider a two DOFs planar manipulator, performing two demonstrations of a given task and starting, in each case, with a different joint configuration, see Figure 1. If one was to record the trajectories of the joints $\boldsymbol{\theta} = \{\theta_1, \theta_2\}$ and that of the end-effector $\mathbf{x} = \{x_1, x_2\}$ of the manipulator, one would observe that the first set of variables, i.e. the joint angles, varies importantly, while the second set of variables remains quite constant across the demonstrations. Thus, the information conveyed by the hand path would appear more reliable than that conveyed by the joints. In other words, the task would appear to put stronger constraints on the hand path than on the joint trajectories. Thus, in order to reproduce the task, one would give more weight to reproducing the hand path than the joint trajectories.

In order to give a measure of the correctness of the reproduction, one needs a measure of imitation performance, i.e. a cost function. This cost function must encapsulate explicitly the task constraints as well as give a measure (metric) of the relative importance of the task constraints.

Let us define $H_1(\boldsymbol{\theta}, \boldsymbol{\theta'})$ and $H_2(\mathbf{x}, \mathbf{x'})$ as a measure of the discrepancy between

demonstrated and reproduced trajectories of the joints ($\boldsymbol{\theta}$, $\boldsymbol{\theta'}$) and hand path ($\mathbf{x}$, $\mathbf{x'}$) respectively, then, without loss of generality, we can define a global measure of the imitation performance by the weighted sum $H(\boldsymbol{\theta}, \boldsymbol{\theta'}, \mathbf{x}, \mathbf{x'}) = w_1 H_1(\boldsymbol{\theta}, \boldsymbol{\theta'} + w_2 H_2(\mathbf{x}, \mathbf{x'})$. The weights w_1 and w_2 give a measure of the relative importance of each signal. In the previous example, we would set $w_1 < w_2$ to give more importance to following the path of the end-effector than to reproducing the joint trajectories.

How to imitate

Once we have measured the relative importance of each task feature and have reported it in the cost function H, we must determine a trajectory for the joints and end-effector of the imitator's arm that is optimal with respect to the cost function. Note that the problem may not be as straightforward as it seems, since the demonstrator and the imitator may differ significantly in their embodiment (arm length, number of degrees of freedom for each arm). This is typically the case when transferring information from a human to a robot, even when the robot has a humanoid shape (as it is the case in our experiments). This idea is illustrated in Figure 2, where the 2 segments of the imitator's manipulator differ in length from those of the demonstrator. If the imitator's arm was let to replay directly the joint trajectories of the demonstrator's arm, we would end up with a very different path for the end-effector than that demonstrated. Conversely, replaying the hand path would result in major differences in the joint trajectories. Figure 2 illustrates the effect of generating different alternatives from satisfying purely the joint trajectories (left) to satisfying only the hand path (right).

Minimizing the cost function determines a trade-off between reproducing faithfully either the hand path or the joint trajectories. Note that the cost function may have several minima. In other words, there may be several solutions to the problem. Note, also, that the hand path and the joint trajectories are not independent variables, but are related to one another through a forward and inverse kinematics function, which we discuss next.

1.2 The Inverse Kinematics Problem

In classical control theory, the *inverse kinematics* usually refers to the inverse computation required to determine the position of each of the robot's joint for a given location of the robot's end-effector (usually its arm). If we define the coordinates of a manipulator as the n-dimensional vector of joint angles $\vec{\theta}$ and the position of the m-dimensional vector $\vec{x}$, the forward kinematics is given by: $\vec{x} = f(\vec{\theta})$, while the inverse kinematics is given by

$$\vec{\theta} = f^{-1}(\vec{x}) \tag{1}$$

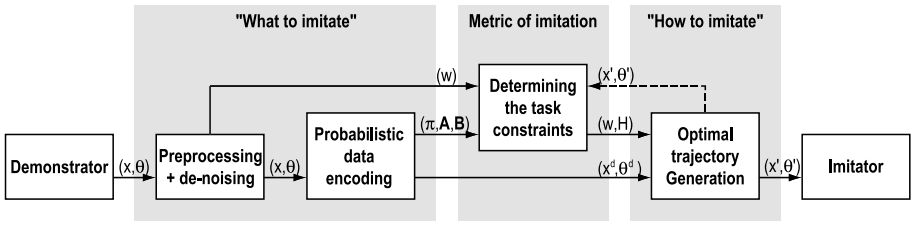

Fig. 3. Information flow across the model

where f is a continuous function $\in \mathbb{R}$.

When the problem is under constrained, i.e. when the number of degrees of freedom n exceeds the number of given variables m, i.e. $n > m$, e.g. when controlling a 4 degrees of freedom robot arm given the 3D position of the target, Equation 1 may have no solutions (degenerate case) or multiple solutions. Several solutions to this problem have been proposed, ranging from numerical solutions ([16], [24]) to geometrical solutions ([3], [25]). Moreover, in order to deal with the non-linearity of the transformation, various methods, based on multiple locally linear models, have also been explored, using either neural networks [19] or regression techniques [11]. Each solution has its advantages and drawbacks. In this paper, we redevelop the pseudo-inverse with optimization method proposed in [16] (a locally linear approximation) and adapt its form to optimize our global cost function H.

2 The Complete System

Figure 3 gives an overview of the input-output flow through the complete model. The model is composed of the following processes:

Preprocessing: We first discard the irrelevant signals (noise), by encoding the dataset $(\boldsymbol{\theta}, \mathbf{x})$ in left-right Hidden Markov Models (HMM) and by eliminating signals with too high variability.

Probabilistic Data Encoding: The remaining (relevant) signals are then encoded in a fully-connected HMM, in order to find an optimal probabilistic representation of the task.

Determining the task constraints: We then compute a measure of the relative importance of each variable of the data set, by using the probabilistic description of the task, and use this to determine the cost function (metric) of imitation performance

Optimal Trajectory Generation: We then compute (using Lagrange optimization) the trajectory that optimizes the cost function, given a set of robot's body constraints.

Next, we describe the computation carried out in each of these modules.

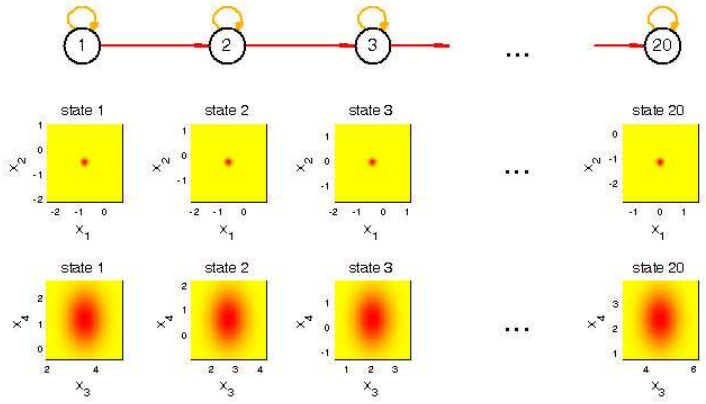

Fig. 4. Encoding of the dataset in a left-right HMM, with diagonal covariance output distribution matrix, used as a pre-processing step to extract the invariants. Here, the signals $\{x_3(t)\}$ and $\{x_4(t)\}$ present important variations, and will be discarded from the remaining processing steps.

2.1 Preprocessing-Denoising

Similarly to our prior work [7, 8], we encode the demonstrated gestures in Hidden Markov Models, so as to discard the intrinsic variability of each person's motion, and, so as to be able to regenerate a generalized form of the demonstrated gestures. However, encoding data using HMMs may lead to very poor performance, when part of the data is subjected to a noise with a large variance [1]. For this reason, we must first discard the data that may contribute to false generalization. In addition to reducing the noise, this procedure also enables us to reduce the dimensionality of the dataset, and, hence speed up learning, an advantage for applications that need to run in real time such as those we consider here.

[1] The *Expectation-Maximization* (EM) process used to train the HMMs aims at finding a local optimum to represent the data, by considering the transition probabilities and output distributions. When the variability of the output signals increases, the training process may not find a satisfying local solution. For example, when encoding two output signals, consisting of a random signal and an invariant signal, the EM training process will try to find a probabilistic representation for both signals, sometimes at the expense of the invariant signal, which is the relevant signal in our case. This problem is particularly evident when we try to model multiple signals with a fully-connected model and/or full covariance matrix, representing the output distribution, because a large set of parameters must be estimated with only a few examples. However, these HMM capabilities are still advantageous, to allow the encoding of signals presenting recurring patterns and/or correlations. Thus, a generic HMM model should nevertheless be used, at least at some point in the process, to encode general signals.

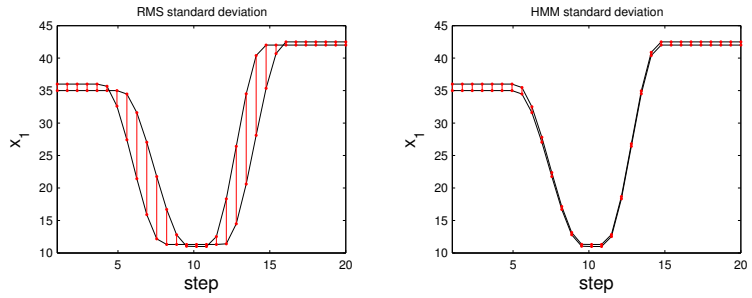

Fig. 5. Illustrative schema of the effect of applying different error measures on two instances of the signals x_1. *Left:* Euclidean distance error, *right:* error computed using the HMM representation. The vertical lines represent the error between the two signals.

Irrelevant (noisy) signals are signals that do not show any consistency across demonstrations, and, thus, may convey no information on a given task constraint. A naive approach would simply compute the Euclidean distance across all instances of a given signal $x_i(t)$, i.e. $d'_i = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(x_i(t) - \bar{x}_i(t))^2}$. Such a measure would not be adequate as it would be very sensitive to temporal distortions across the demonstrations, which are frequent in the experiments we consider (as two demonstrators may not perform the same task at the same speed). Thus, in order to compare two signals that may encapsulate non-homogeneous distortions in time, we choose to encode those in HMMs (to get rid of the time-distortion) and, then, to compare the variability of their distribution in the HMM representations (see Figure 4). The HMM encoding of the two signals may be viewed as trying to stretch temporally one signal, so as to match the other signal. Once encoded in the HMM, one may retrieve for each signal $\{x_i(t)\}_{i=1}^{K}$, a time-series of variables (the observables), whose distribution follow a set of Gaussians centered on $\{\mu_i(t)\}_{i=1}^{K}$ with variance $\{\sigma_i^2(t)\}_{i=1}^{K}$ (see Figure 5). One can then measure the variability of each signal using:

$$d_i = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\sigma_i^2(t)} \tag{2}$$

For this pre-processing step, we used left-right HMM with up to 20 states and a diagonal covariance matrix. This is sufficient to find the relevant signals with only a few iterations (maximum number of EM iterations fixed to 2). Note that, in this pre-processing step, the role of the HMM is to extract robustly the spatial variations of the signals, but not to find an efficient encoding of the dataset.

The signals collected by the robot come from different modalities, with their own ranges of values, measurement units, and sensory resolution. In order to compare the intrinsic variability of different signals irrespective of their

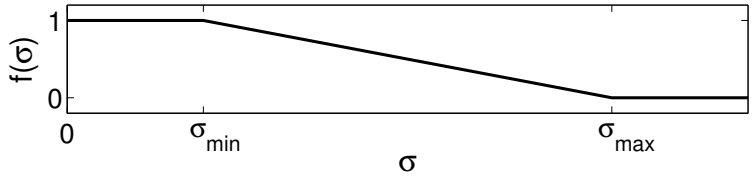

Fig. 6. Transfer function $f()$ used to transform the standard deviation of each signal to a value between 0 and 1, considering the sensibility and range of variation of the associated sensor.

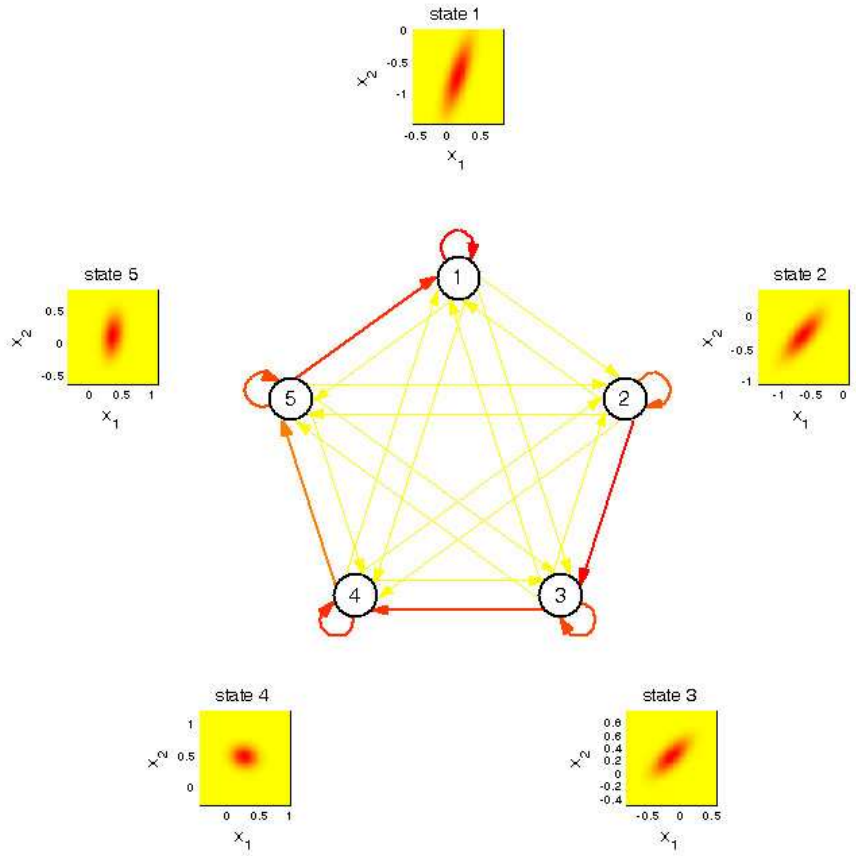

Fig. 7. Encoding of the dataset after preprocessing (see Figure 4) in a fully-connected HMM with 5 states. Each of the 5 states of the HMM outputs 2 distributions for the observable signals $\{x_1(t)\}$ and $\{x_2(t)\}$. The correlation across the signals is computed in the covariance matrices of the observable distributions attached to each state (tilted ellipses).

modality, we rescale those using a transfer function f,(see Figure 6). We then discard all the signals, such that $f(d_i, \sigma_i) < 0.1$

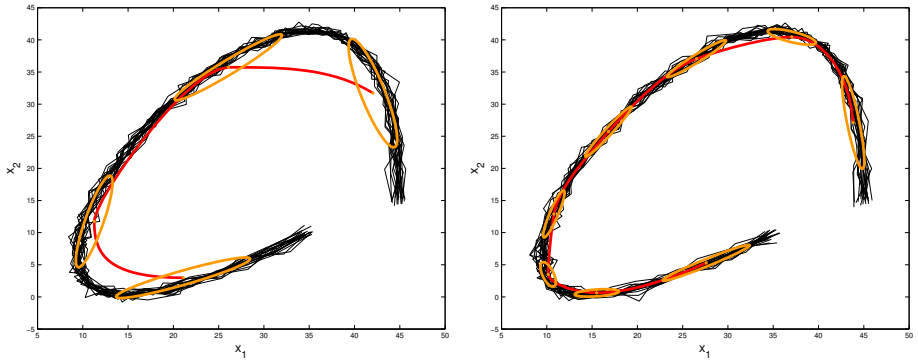

Fig. 8. Training data (thin lines) and generalized form of the data after retrieval (thick line), when encoding the data in a 4-states HMM (left) and in an 8-states HMM (right). The length of the axes of the ellipses (superimposed to the curves) correspond to the correlation factors learned by the HMM.

2.2 Probabilistic Data Encoding:

After preprocessing, the resulting signals are then encoded in a continuous fully-connected HMM with full covariance matrix. The HMM uses a set of parameters $\{\vec{\pi}, \mathbf{A}, \mathbf{B}\}$, representing, respectively, the initial states distribution, the states transition probabilities, and the observable distributions [2] . The distributions are continuous and modeled by a single Gaussian per state and output. That is, to each state $\{q(t)\}$ of the model is associated a distribution of observables with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, i.e. $\mathbf{B} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$.

Since the optimal number of states in the HMM may not be known beforehand, we use the *Bayesian Information Criterion* (BIC) criterion to select the optimal number of states for the model, by determining a trade-off between optimizing the model's likelihood (a measure of how well the model fits the data) and minimizing the parameters (i.e. the number of states used to encode the data).

$$BIC = -2\,\mathcal{L} + n_p\,log(T) \tag{3}$$

$\mathcal{L}$ is the log-likelihood of the model, given the observed dataset. n_p is the number of independent parameters in the HMM, and T the number of observation data used in fitting the model (in our case $T = K \cdot N$, for trajectories of size N). The first term of the equation measures how well the model fits the data, while the second term is a penalty factor that aims at keeping the total number of parameters low. In our experiments, we compute a set of candidate HMMs with up to 20 states and retain the model with the minimum score. Figure 8 shows the result of encoding data using a 4-state and a 8-state HMM (same data as in Figure 9, 4 and 7). The 8-state HMM fits better the data, but uses more parameters.

[2] People unfamiliar with HMM should refer to [21]

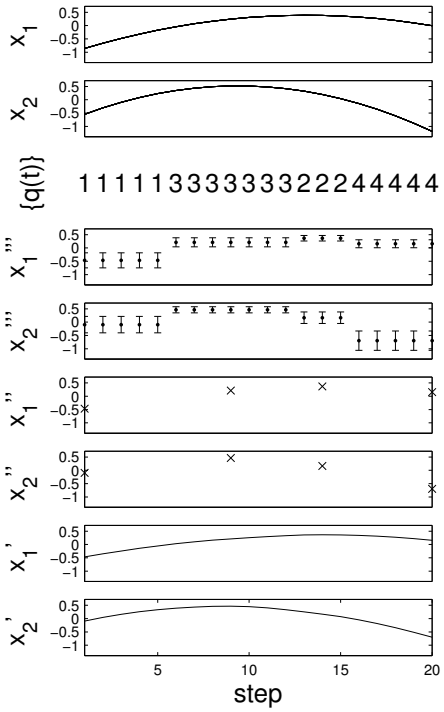

Fig. 9. Generalization and retrieval process, using the HMM representation of the data (see also Figure 8 and 7). Here a generated dataset of 2 signals is represented.

The transition probabilities $p(q(t)=j|q(t\text{-}1)=i)$ and the observation distributions $p(x(t)|q(t)=i)$ are estimated by the *Baum-Welch* algorithm, an *Expectation-Maximization* (EM) algorithm, that maximizes the likelihood that the training dataset can be generated by the HMM.

Once trained, the HMM can be use to recognize gestures (in our experiments, this is used to decide whether a new demonstration belongs or not to the same task). In order to measure the similarity between a new gesture and the ones encoded in the model, we run the *forward-algorithm*, an iterative procedure to estimate the likelihood that the observed data could have been generated by the model.

Once trained, the HMM can also be used to regenerate a generalized version of the gestures (in our experiments, this is used to generate a desired trajectory for the *Optimal Trajectory Generation* module). In order to regenerate a gesture from a given model, we reconstruct the optimal sequence of state transition using the *Viterbi algorithm*. Given this sequence of states, we retrieve a time-series of K variables $\{x_i'''(t)\}, i = 1, .., K$, by taking the mean values $\boldsymbol{\mu}$ of the Gaussian distributions for each observable (see Figure 9). We, then, extract a set of key-points $\{x_i''(t)\}$ from these time series. If there is a transition to a state n at time t_1 and if there is a transition to another state at time t_2, a key-point is created at the mean time $\frac{t_1+t_2}{2}$. By interpolating between these key-points and normalizing in time, we reconstruct the series of

observables $\{x'_i(t)\}$, using Piecewise cubic Hermite polynomial functions, see Figure 8). This gives us finally a set of desired trajectories $\vec{\theta}^d, \vec{x}^d\}$, which will then be used to compute the optimal trajectory for the imitator, as we will see next.

2.3 Determining the task constraints by a cost function

In [6], we proposed a general formalism for determining the cost function of an imitation task. The metric (or cost function) measures the quality of the reproduction, and, as such, drives the selection of an appropriate controller for the reproduction of the task.

Let $\{x_i(t)\}$ and $\{x'_i(t)\}$ with $i = 1, ...K$ be a set of demonstrated and a reproduced signals. Then, we define the global cost function H as:

$$H = \frac{1}{K} \sum_{i=1}^{K} w_i \cdot H_i(x_i, x'_i) \tag{4}$$

where $w_i \in [0, 1]$ are the weights measuring the relative importance of each signal. These are given by (see section 2.1):

$$w_i = f(d_i) \tag{5}$$

The cost function is bounded: $H \in [0, 1]$. H=0 corresponds to a perfect reproduction.

In the experiments reported here, we consider the particular case, where:

$$\begin{aligned} H(\vec{\theta}, \vec{\theta'}, \vec{x}, \vec{x'}) = & \frac{1}{2}(\vec{\theta} - \vec{\theta}^d)^T \mathbf{W}^\theta (\vec{\theta} - \vec{\theta}^d) \\ & + \frac{1}{2} (\vec{x} - \vec{x}^d)^T \mathbf{W}^x (\vec{x} - \vec{x}^d) \end{aligned} \tag{6}$$

Where $(\vec{\theta}, \vec{\theta}^d), (\vec{x}, \vec{x}^d)$ are, respectively, the current and desired positions of the joints and of the hand. $\mathbf{W}^\theta$ (4x4 matrix) and $\mathbf{W}^x$ (3x3 matrix) are the set of weights w_i associated, respectively, with the four joints trajectories and the 3D Cartesian path of the hand. By simplicity and for clarity, we will omit the vectorial notations for the rest of the developments.

Once the cost function and the relative influence of each constraint have been determined, we generate a trajectory that is optimal with respect to the cost function H. We must, however, take into account the body constraints of the robot, which are given by the inverse kinematics function $\theta = f^{-1}(x)$. Following [16], we consider an iterative, locally linear, solution to this equation, such that:

$$\dot{x} = \mathbf{J} \cdot \dot{\theta} \tag{7}$$

Where $\dot{\theta}(t) = \theta(t) - \theta(t-1)$ and $\dot{x}(t) = x(t) - x(t-1)$ are the velocity vectors of the joints and the hand path. $\mathbf{J}$ is the Jacobian, a $4x3$ matrix.

Similarly, by substituting $c(t) = \theta(t-1) - \theta^d(t)$ and $d(t) = x(t-1) - x^d(t)$ in (6), we have:

$$\begin{aligned} H(\dot{\theta}, \dot{x}) = & \frac{1}{2} \left(\dot{\theta} - c\right)^T \mathbf{W}^\theta \left(\dot{\theta} - c\right) \\ & + \frac{1}{2} \left(\dot{x} - d\right)^T \mathbf{W}^x \left(\dot{x} - d\right) \end{aligned} \tag{8}$$

The problem is now reduced to finding a minimum of (8) when subjected to (7). Since H is a quadratic function, the problem can be solved analytically by Lagrange optimization. We define the *Lagrangian* as:

$$\begin{aligned} L(\dot{\theta}, \lambda) = & \frac{1}{2} \left(\dot{\theta} - c\right)^T \mathbf{W}^\theta \left(\dot{\theta} - c\right) \\ & + \frac{1}{2} \left(\dot{x} - d\right)^T \mathbf{W}^x \left(\dot{x} - d\right) \\ & + \lambda^T (\dot{x} - \mathbf{J} \cdot \dot{\theta}) \end{aligned} \tag{9}$$

where λ is the vector of associated Lagrange multipliers and λ^T its transpose.

There are two necessary condition for $\dot{\theta}^*$ to be an optimum of H subjected to the condition given by (7):

$$\frac{\partial L(\dot{\theta}^*, \lambda)}{\partial \dot{\theta}} = 0; \tag{10}$$

$$\frac{\partial L(\dot{\theta}^*, \lambda)}{\partial \lambda} = 0; \tag{11}$$

When solving (11), we find again (7). When substituting $\dot{x}$ by $\mathbf{J} \cdot \dot{\theta}$ in (9), and, then, deriving along $\dot{\theta}$, we get:

$$W^{\theta}(\dot{\theta}^* + c) + \mathbf{J}^T(W^x(\mathbf{J}\dot{\theta}^* + d)) = 0 \tag{12}$$

where $\mathbf{J}^T$ is the transpose of $\mathbf{J}$.

Solving for $\dot{\theta}$:

$$\dot{\theta}^* = -(\mathbf{W}^{\theta} + \mathbf{J}^T\mathbf{W}^x\mathbf{J})^{-1}(\mathbf{J}^T\mathbf{W}^x d + \mathbf{W}^{\theta} c) \tag{13}$$

We can then recompute the joint trajectories, using $\theta(t) = \theta(t-1) + \dot{\theta}(t)$ and the hand path using $\dot{x}(t) = \mathbf{J}\dot{\theta}(t)$ and $x(t) = x(t-1) + \dot{x}(t)$.

3 Experiments

We conducted a set of three experiments to demonstrate the validity of our model for teaching a humanoid robot simple uni-manual and bi-manual manipulatory tasks. The tasks consisted in "stirring the fondue", "hammering a nail" and "grabbing a ball with two hands". In each case, the robot was shown the task five times. Small variations in the demonstrations were introduced to let the key features of the task (i.e. the invariants) be more salient. The model, then, extracted the relative importance of each variable (i.e. hand path, joint trajectories, position of the hand with respect to the objects) and constructed the weights of the cost function accordingly, see Section 2.3. Note that the weights for the position of the hand relative to the objects were used to determine which of the absolute or relative task description one should choose, by picking the representation with maximal weights.

Then, the robot was required to reproduce each task in different locations in space, by displacing the object to be manipulated. This procedure aimed at demonstrating the robustness of the system when the constraints were

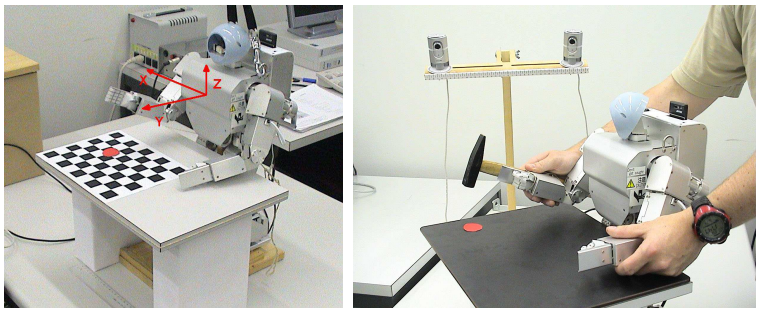

Fig. 10. Experimental Setup: **Left:** The robot stands upright in front of a table. A color-based stereoscopic vision system tracks the 3D-position of the different objects. **Right:** The robot is being taught how to hammer a nail through kinesthetics, i.e. by recording the kinematics of the joint trajectories while the demonstrator moves the robot's arms through each of the task's steps.

transposed to different locations within the robot's workspace. Would the optimal solution change? If yes, would the new trajectory remain stable and smooth?

4 Experimental setup

The experiments were conducted with a Fujitsu HOAP-2 humanoid robot with 25 degrees of freedom (DOF), of which only the 8 DOFs of the two arms were required in the experiments. The remaining DOFs of the torso and legs were set to a constant position, so as to support the robot in an upright posture, facing a table, see Figure 10.

A color-based stereoscopic vision system tracks the 3D-position of the different objects used in the experiments at a rate of 15Hz, with an accuracy of 10 mm. The system uses two Phillips web-cams with a resolution of 320x240 pixels. The tracking is based on color segmentation for detecting the skin color and the color of the different objects in the YCbCr color space [3].

In the experiments reported here, the robot was taught through kinesthetics, i.e. by the demonstrator moving its two arms through each of the task's steps, see Figure 10. To achieve this, the robot's motors were set in a passive mode, whereby each limb could be backdrived by the human demonstrator. The kinematics of each joint motions was recorded at a rate of 1000Hz during the demonstration and was, then, downrated to 15Hz to match the tracking rate of the objects.

[3] Only Cb and Cr are used, to be robust to changes in luminosity

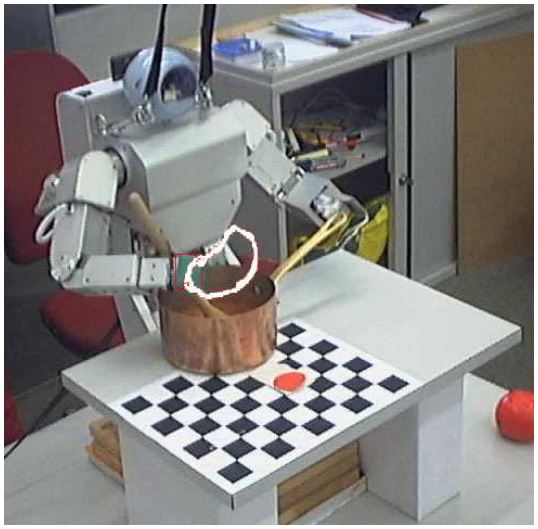

Fig. 11. The trajectory in white corresponds to the optimal reproduction of the gesture "stirring the Fondue", i.e. the solution leading to the lowest value of the cost function.

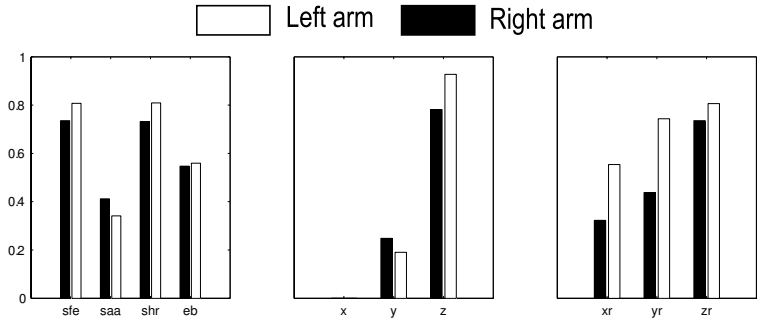

Fig. 12. Weights extracted after five demonstration of "stirring the Fondue". SFE: Shoulder Flexion-Extension, SAA: Shoulder Abduction-Adduction, SHR: Shoulder Humeral Rotation, EB: Elbow. x,y,z: Absolute position of the hand. xr,yr,zr: relative position of the hand with respect to the saucepan.

4.1 "Stirring the Fondue"

The first experiment consisted in teaching the robot how to "stir a fondue" (typical swiss meal), i.e. it had to learn to hold stiff the saucepan with the left arm and to perform cyclic rotational motions with its right arm, while keeping the stick inside the saucepan, see Figure 11. The robot was shown five demonstrations, during which the demonstrator made sure to keep the same rotational motions, while displacing the position of the saucepan on the table.

Figure 12 shows the values taken by the weights computed by the *determining the task constraints* module, see Section 2.3. As expected, the weights associated with the joints are very important (reflecting their relative invariance in the task), except for SAA (which varied more than the other DOFs, as an

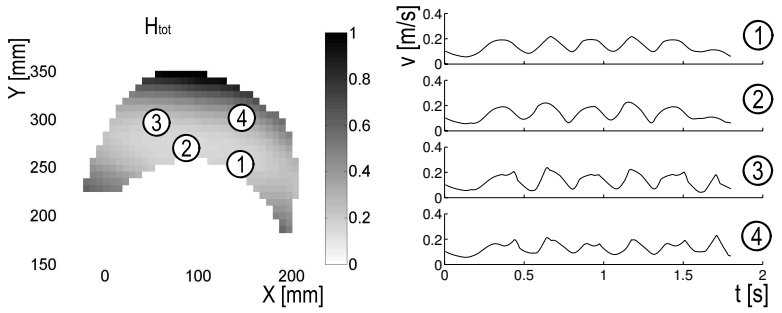

Fig. 13. **Left:** map of the values taken by the cost function H as a function of the position of the sauce pan on the table in the task "stirring the fondue". **Right:** Variation of the amplitude of the speed vector of the right hand along time.

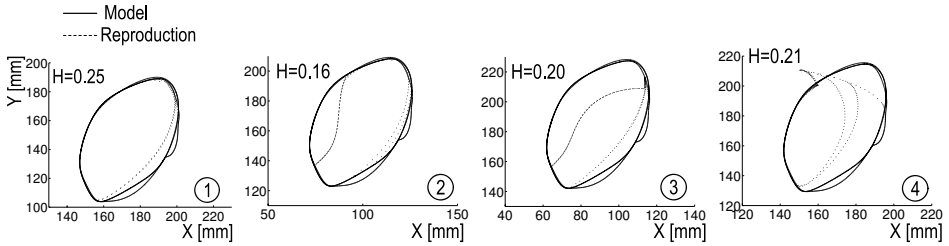

Fig. 14. (Straight Line:) Trajectory followed by the hand path when the saucepan is located at positions 1, 2, 3 and 4 in the workspace, see Figure 13, superimposed to the demonstrated trajectory (thin line). Since the hand path is the criterion with lowest importance, we observe large distortions, while the corresponding value of the cost function remain low, see Figure 13.

effect of displacing the saucepan across the table during the demonstrations). The absolute hand path is on the other hand much more variable, except along z (i.e. no vertical motion). Note that the saucepan was displaced principally along the x axis rather than along the y axis, to ensure that all locations would be reachable by the robot (thus, leading to a larger weight for y than for x). Similarly, the relative position of the hand with respect to the saucepan was pretty invariant, leading to very high weights for these features.

Figure 13 shows the values taken by the cost function H when estimating the optimal trajectory, by moving systematically the location of the saucepan on the table. The white areas correspond to the locations where there are no solutions (i.e. areas not reachable by the robot). As the saucepan is moved across the table, the robot must adapt its posture to satisfy best the different constraints, such as, e.g., remaining within its joint limits. This results in important distortions of the hand path (whose reproduction has lowest importance according to the cost function) at locations 2 and 3 (that reach out of the joint limits), see Figure 14, while the cost function at the same locations remain very low, see Figure 13.

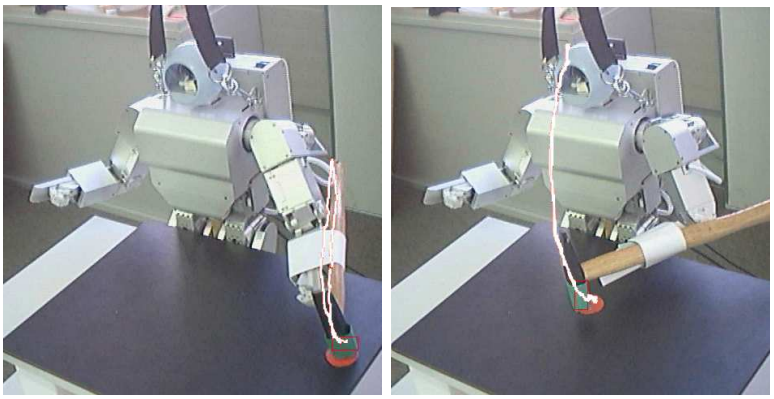

Fig. 15. Reproduction of the movement "hammering a nail" at 2 different locations in the task workspace.

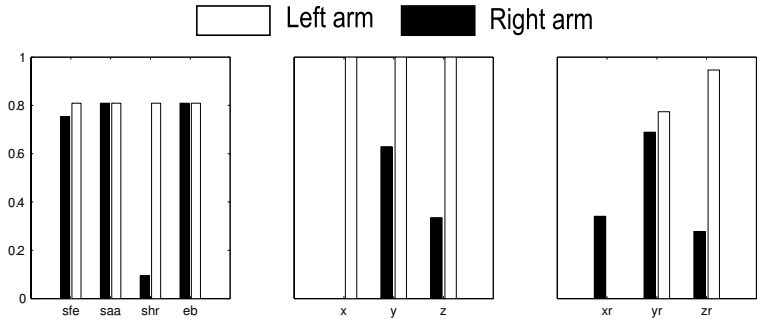

Fig. 16. Weights extracted after five demonstrations of "hammering a nail" while keeping the left arm in the same position.

4.2 Hammering a nail

In the second experiment, the task consisted in "hammering a nail", see Figure 10 right, i.e. to let the right arm slam down on a color dot placed on the table. In order to demonstrate the sensitivity of our system to the experimental conditions, we conducted two sets of experiments. In the first set, the robot was shown the gesture with the right arm only, leaving the left arm in a rest position. In the second set, the left arm was moved randomly, during each of the five demonstrations.

Figures 16 and 17 show the values of the weights extracted in each of the two conditions. In the first condition, see Fig. 16, the system has found a correlation along the y and z axes between the left arm (static) and the nail. That is due to the fact that, during the demonstration, the "nail" was moved mainly along the x axis. In contrast, in the second condition, no such correlation was found, since the left arm was moved randomly, see Fig. 17.

Once trained, the robot was, then, requested to reproduce the motion at different locations on the table.

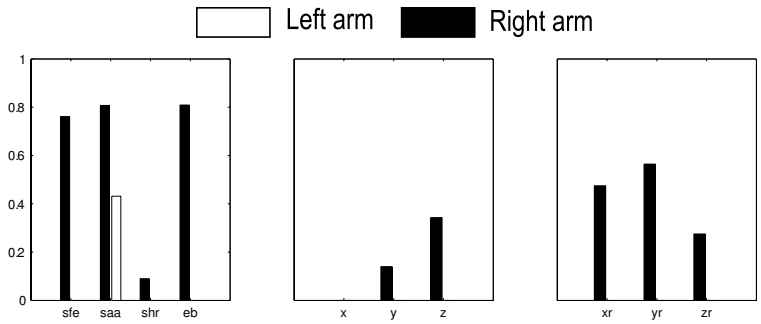

Fig. 17. Weights extracted after five demonstrations of "hammering a nail" while setting the left arm to a random position at the start of each demonstration.

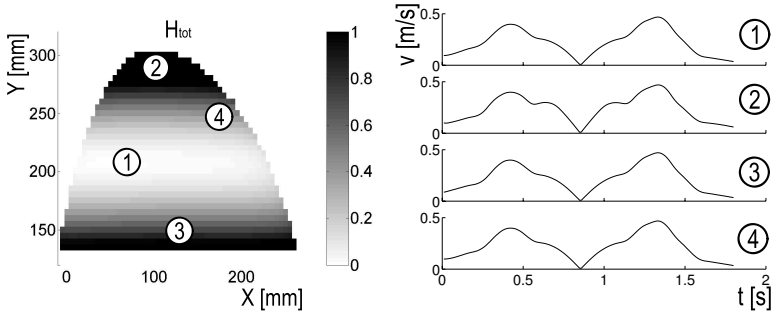

Fig. 18. **Left:** map of the values taken by the cost function H as a function of the position of the "nail" on the table in the task "hammering a nail" (first condition: left arm static). **Right:** Variation of the amplitude of the speed vector of the right hand along time.

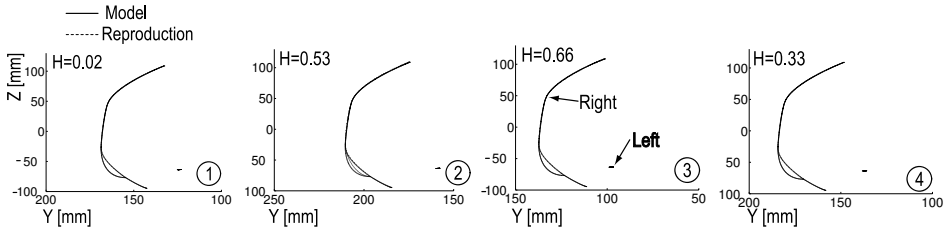

Fig. 19. Trajectory followed by the hand path when the "nail" is located at positions 1, 2, 3 and 4 in the workspace, see Figure 18, superimposed to the demonstrated trajectory in the task "hammering a nail".

As expected, the optimal value of the cost function H increases when the target approaches the limits of the workspace, i.e. when the constraints on the position of the hand and joints can no longer be fulfilled, see Fig.18 and Fig.20. The optimal trajectory, i.e. the optimum of the cost function, is reached in the middle of the workspace.

Figures 19 and 21 show the optimal trajectories, i.e. the trajectory corresponding to the minimum of the cost function, for each of the two conditions. We observe that, after training in the first condition, the left arm moves along y with the target during reproduction, in order to satisfy the erroneous relation

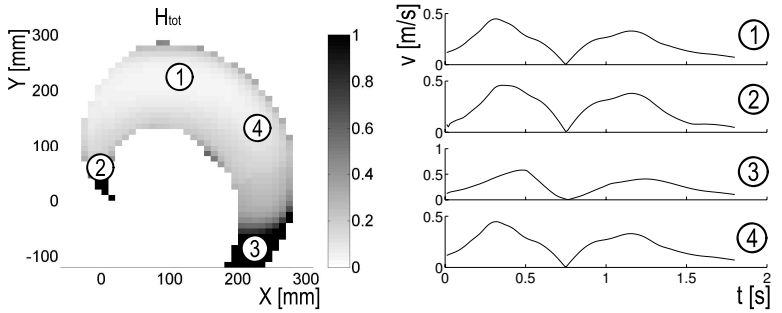

Fig. 20. **Left:** map of the values taken by the cost function H as a function of the position of the "nail" on the table in the task "hammering a nail" (second condition: left arm random). **Right:** Variation of the amplitude of the speed vector of the right hand along time.

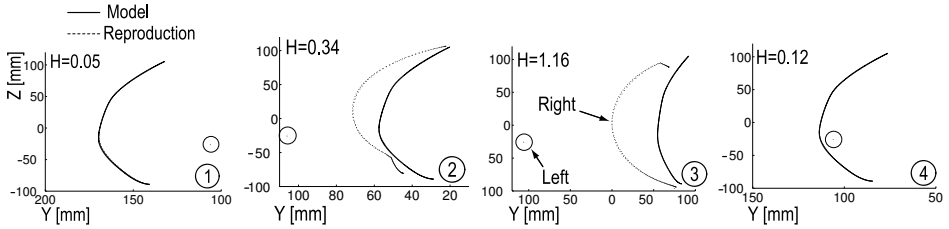

Fig. 21. Trajectory followed by the hand path when the nail is located at positions 1, 2, 3 and 4 in the workspace (see Fig 20), superimposed to the demonstrated trajectory.

between the target position and that of the left arm, see Figure 19. Thus, when the target comes too close to the front of the robot, the left arm is blocked and the system no longer finds a plausible solution. Such constraints are inexistent in the second training condition and, thus, ones does not observe the same limitation (compare the valid areas of the cost function for $y < 150$ in Figures 18 and 20).

We, also, note that in all four examples the reproduced hand path fits very closely the desired hand path, even when the "nail" is at the limit of the task workspace. This is due to the fact that the workspace of the task, in the first condition, is now at the intersection between the workspaces of the two arms. Thus, when the left arm is blocked at the limit of the task workspace, the right arm can still reproduce the movement because its workspace is not limited in the same way.

In the second condition(Fig. 21), the valid workspace depends only on the right arm (since the left arm is static). Thus, at the extreme parts of the workspace (trajectory 2 and 3), the reproduced hand path no longer matches the demonstrated one.

Fig. 22. HOAP2 is grabbing a ball with to hands

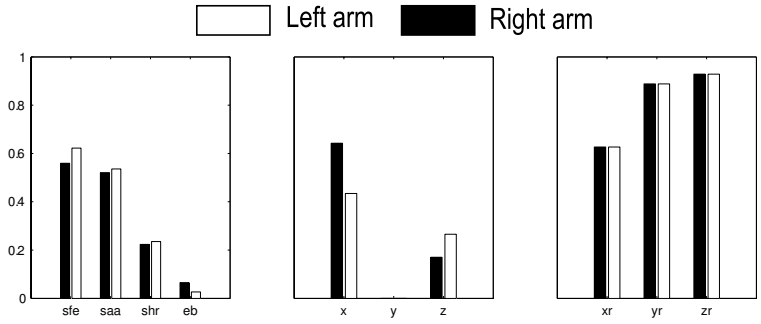

Fig. 23. Weights extracted after five demonstrations of the task "Grabbing a ball with 2 hands"

4.3 Grabbing a ball with two hands

This experiment consisted in grasping a ball with two hands and lifting it up over a short distance, see Figure 22. During training and testing, the location of the target (the ball) was varied along a vertical plan perpendicular to the HOAP2's torso. While, in the first two experiments, we had tested our system with static targets, we here consider a task, composed of two parts. In the first part, the object is static, while in the second part, it moves together with the two hands. This creates a strong correlation between the hand path and the object path, which is correctly extracted by the weights, see Figure 23. Note that, because we compute the weights by averaging over the whole trajectory, we loose the information that this correlation applies only to the second part of the task. In future work, we will extend the metric to make the weights time dependent, i.e. to replace $\mathbf{W}^\theta, \mathbf{W}^x$ in Equation 6 by $\mathbf{W}^\theta(t)$ and $\mathbf{W}^x(t)$.

However, this limitation does not prevent us in this particular task to obtain good performance during testing. Figure 24 shows that, once again, the best area to reproduce the movement is in front of the robot at about half the height (around $z = -100$) covered during testing. Figure 25 displays four trajectories reproduced in different parts of the task workspace. In each case, the reproduction is qualitatively good and we observe only very small distortions of the hand path at the edge of the workspace. In other words, in each case, the goals of the task, i.e. grabbing the ball and lifting it, are satisfied. When

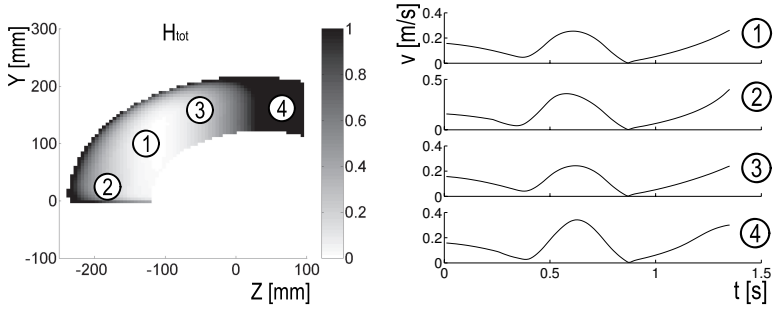

Fig. 24. **Left:** map of the values taken by the cost function H as a function of the position of the "ball" on the table in the task "Grabbing a ball with 2 hands" (first condition: left arm static). **Right:** Variation of the amplitude of the speed vector of the right hand along time.

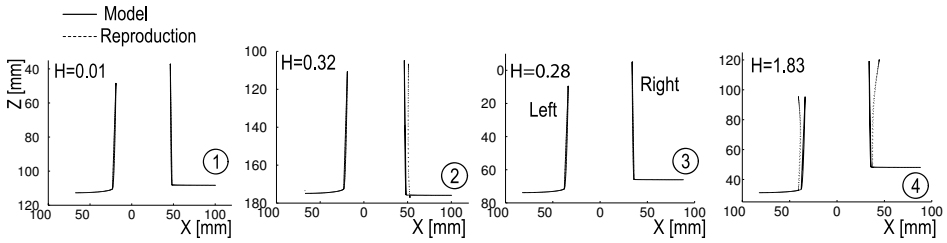

Fig. 25. Trajectory followed by the hand path when the ball is located at positions 1, 2, 3 and 4 in the workspace, see Figure (See Fig 24), superimposed to the demonstrated trajectory.

reaching the limits of its joints, the robot adapted the trajectory of its two arms simultaneously, resulting in a slight shift from the original trajectory, while still keeping the ball firm in its grasp.

5 Discussion and Conclusion

This paper presented a method to 1) extract the important features of a given task, 2) to determine a generic cost function to evaluate the robot's performance, and, finally, 3) to optimize the robot's reproduction of the task in a new context based on this cost function. The method was validated in three sets of experiments that set different constraints. The tasks consisted in either reproducing the gesture (stirring a stick in a saucepan), the path of the end effectors relative to the object (hammering a nail) or the path of the effectors relative to one another (grasping a ball with 2 hands).

We showed that, in each case, the robot managed to adapt its motions "cleverly", so as to reproduce correctly the important qualitative features of each task (i.e. keeping the stick in the saucepan, hitting the nail, keeping the ball firmly in its grasp). However, none of these high-level goals were explicitly

represented in the robot's control system, but, were nevertheless correctly extracted by our probabilistic system.

Note, however, that this statistical method has limitations. Because the relative importance of each task constraint is computed over the whole demonstration, it blurs out changes in the relative importance of the constraints that may appear over the course of the demonstrations (as in the example of grasping the ball and lifting it). These changes could, easily, be determined, by looking at the variability of each state of the HMM separately. Thus, in further work, we will extend the method to take into account time-varying constraints. Moreover, this will be used to determine transitions across subtasks, a subtask consisting of a region of the cost function with equal value for all constraints. We will also investigate ways of setting priors on the weights, so that the system could converge faster to a task description, when two tasks are very similar.

The system we presented for solving the "how to imitate" issue is generic, in the sense that it makes no assumption on the robot's configuration (number of degrees of freedom and length of segments). Moreover, it produces smooth trajectories, even at the limits of the robot's workspace, which makes it a suitable robot controller.

As first stressed out by Nehaniv and colleagues [18], there is a multitude of correspondence problems, when trying to transfer skills across various agents and situations. One may consider

- **Different embodiments:** the demonstrator does not share the same morphology and characteristics as the imitator.
- **Different situations:** the environment and constraints during the demonstration and the reproduction are different

In the experiments we presented in this paper, the robot was being taught through kinesthetics. By showing kinesthetically how to perform a task, the user "embodies" the robot's body. Thus, this way, we simplified the correspondence problem and overlooked the problem of having different embodiements. However, by testing the system in different situations than those taught, we tackled the second aspect of the correspondence problem.

Note that, in these experiments reported here, we assumed implicitly that the kinematics of joint angle trajectories and hand path is sufficient to describe the skill, and that dynamics is of less importance. This may not be true and taking into account the forces applied on the object may certainly be very important in certain task. However, it is very likely that these are not learned through imitation, but, through more generic motor learning processes.

Note that the correspondence problem or "how to imitate" question relates,

also, to a more generic issue of imitation, that of defining a common representation for sharing the information across various modalities [10, 20, 22, 13], which we did no address here. Indeed, in the present work, we assumed that both agents would have access to very low-level information, namely to the position of each joint and of the end-effector in real time. Such an assumption is certainly not valid in all scenarios and does not account for human capacity to imitate.

References

[1] A. Alissandrakis, C.L Nehaniv, and K. Dautenhahn. Imitating with alice: Learning to imitate corresponding actions across dissimilar embodiments. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 32:4:482–496, 2002.

[2] A. Alissandrakis, C.L Nehaniv, and K. Dautenhahn. Towards robot cultures? - learning to imitate in a robotic arm test-bed with dissimilar embodied agents. *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, 5:1:3–44, 2004.

[3] T. Asfour and R. Dillman. Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem. In *IEEE/RSJ Intl Conference on intelligent Robots and Systems*, 2003.

[4] P. Bakker and Y. Kuniyoshi. Robot see, robot do : An overview of robot imitation. In *AISB Workshop on Learning in Robots and Animals, Brighton, UK*, April 1996.

[5] T. Belpaeme, B. de Boer, and B. Jansen. The role of population dynamics in imitation. In *In Dautenhahn, K. and Nehaniv, C.L. (eds.) Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions. Cambridge University Press. Cambridge, UK.* MIT Press, 2004. To appear.

[6] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal. Discovering optimal imitation strategies. *Robotics & Autonomous Systems*, 47:2-3:69–77, 2004.

[7] S. Calinon and A. Billard. Stochastic gesture production and recognition model for a humanoid robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[8] S. Calinon and A. Billard. Recognition and reproduction of gestures using a probabilistic framework combining pca, ica and hmm. In *Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany*, 2005.

[9] S. Calinon, F. Guenter, and A. Billard. Goal-directed imitation. In *Proceedings of the International Conference on Robotics and Automation, Barcelona.*, 2005.

[10] J. Demiris and G. Hayes. Imitation as a dual-route process featuring pre-

dictive and learning components: A biologically-plausible computational model. In C. Nehaniv and K. Dautenhahn, editors, *Imitation in Animals and Artifacs*. MIT Press, 2001 (In Press).

[11] A. D'Souza, S. Vijayakumar, and S. Schaal. Learning inverse kinematics. In *IEEE/RSJ Intl Conference on Intelligent Robots and Systems*, 2001.

[12] M. Ehrenmann, O. Rogalla, R. Zoellner, and R. Dillmann. Teaching service robots complex tasks: Programming by demonstation for workshop and household environments. In *Proc. of the IEEE Int. Conf. on Field and Service Robotics. (FRS), Finnland*, 2001.

[13] M. Ito and J. Tani. On-line imitative interaction with a humanoid robot using a dynamic neural network model of a mirror system. *Adaptive Behavior*, 12:2:93–115, 2004.

[14] B. Jansen and T. Belpaeme. Goal-directed imitation through repeated trial-and-error interactions between agents. *Robotics & Autonomous Systems*, 2005. Submitted.

[15] M. Kaiser and R. Dillmann. Building elementary robot skills from human demonstration. In *Proceedings. International Conference on Robotics and Automation*, volume 3, 1996.

[16] A. Liegeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics, vol. 7 (12), pp. 868-871*, 1977.

[17] C. Nehaniv and K. Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In J. Demiris and A. Birk, editors, *Learning Robots: An Interdisciplinary Approach*. World Scientific Press, 1999.

[18] C. L. Nehaniv. Nine billion correspondence problems and some methods for solving them. In *Proceedings of the Second International Symposium on Imitation in Animals & Artifacts, The Society for the Study of Artificial Intelligence and Simulation of Behaviour*, pages 93–95, 2003.

[19] E. Oyama, A. Agah, K. MacDorman, T. Maeda, and S. Tachi. A modular neural network architecture for inverse kinematics model learning. *Neurocomputing*, 2001.

[20] E. Oztop and M. A. Arbib. Schema design and implementation of the grasp-related mirror neuron system. *Biological Vybernetics*, 2002. In Press.

[21] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:2:257–285, February 1989.

[22] E. Sauser and A. Billard. Three dimensional frames of references transformations using recurrent populations of neurons. *Neurocomputing*, 64:5–24, 2004.

[23] M. Skubic and R.A. Volz. Acquiring robust, force-based assembly skills from human demonstration. In *IEEE Transactions on Robotics and Automation*, volume 16:6, pages 772 –781, 2000.

[24] G. Tevatia and S.Schaal. Inverse kinematics for humanoid robots. In

IEEE Intl Conference on Robotics and Automation, 2000.

[25] X. Wang and J.P. Verriest. A geometric algorithm to predict the arm reach posture for computer-aided ergonomic evaluation. *The Journal of Visualization and Computer Animation, vol. 9, pp. 33-47*, 1998.

[26] M. Yeasin and S. Chaudhuri. Toward automatic robot programming: learning human skill from visual data. In *IEEE Transactions on Systems, Man and Cybernetics, Part B*, volume 30:1, 2000.

[27] J Zhang and B Roessler. Self-valuing learning and generalization with application in visually guided grasping of complex objects. *Robotics & Autonomous Systems*, 47:2-3:772 –781, 2004.