2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

# Evaluation of a probabilistic approach to learn and reproduce gestures by imitation

Sylvain Calinon[†], Eric L. Sauser[‡], Aude G. Billard[‡] and Darwin G. Caldwell[†]

***Abstract*— We present an approach based on Hidden Markov Model (HMM) and Gaussian Mixture Regression (GMR) to learning robust models of human motion through imitation. The proposed approach allows us to extract redundancies across multiple demonstrations and build time-independent models to reproduce the dynamics of the demonstrated movements. The approach is systematically evaluated by using automatically generated trajectories sharing similarities with human gestures. The proposed approach is contrasted with four state-of-the-art methods previously proposed in robotics to learn and reproduce new skills by imitation. An experiment with a 7 DOFs robotic arm learning and reproducing the motion of hitting a ball with a table tennis racket is then presented to illustrate the approach.**

## I. INTRODUCTION

Robot Programming by Demonstration (PbD) covers methods by which a robot learns new skills through human guidance [1], [2]. It requires control strategies that can adapt in real time to perturbations, such as changes of position and orientation of objects. The present work addresses this challenge in investigating and comparing methods by which PbD is used to learn the dynamics of demonstrated movements, and, hence, provides the robot with a generic and adaptive model of control.

Most approaches to trajectory modeling estimate a time-dependent model of the trajectories, by either exploiting variants along the concept of spline decomposition [3], [4] or through statistical encoding of the time-space dependencies [5], [6]. Such modeling methods are very effective and precise in the description of the actual trajectory, and benefit from an explicit time-precedence across the motion segments to ensure precise reproduction of the task. However, the explicit time-dependency of these models require the use of other methods for realigning and scaling the trajectories to handle perturbation. As an alternative, other approaches have considered modeling the intrinsic dynamics of motion [7]–[9]. Such approaches are advantageous in that the system does not depend on an explicit time variable and can be modulated to produce trajectories with similar dynamics in areas of the workspace not covered during training.

*Hidden Markov Model* (HMM) has previously been reported as a robust probabilistic method to deal with the spatial and temporal variabilities of human motion across various demonstrations [9]. The approach that we propose relies on HMM to encode the motion and on *Gaussian Mixture Regression* (GMR) [10] to generalize the motion during reproduction. In comparison to other statistical regression methods such as *Locally Weighted Regression* (LWR) [11] or *Gaussian Process Regression* (GPR) [7], [12], GMR does not model the regression function directly, but models a joint probability density function of the data, and then derives the regression function from the density model [1]. This may be an advantage in some robotic applications since the input and output components are only specified in the very last step of the algorithm, and can be interchanged during reproduction (e.g., this can be used as a fast retrieval process with different sources of missing data, allowing us to consider simultaneously any input/output mappings). Vijayakumar *et al* suggested to improve the LWR approach so that it can operate efficiently in high dimensional space through *Locally Weighted Projection Regression* (LWPR) [13].

[†]Advanced Robotics Department, Italian Institute of Technology (IIT), 16163 Genova, Italy. {sylvain.calinon,darwin.caldwell}@iit.it.

[‡]Learning Algorithms and Systems Laboratory (LASA), Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland. {eric.sauser,aude.billard}@epfl.ch.

## II. PROPOSED PROBABILISTIC APPROACH

$M$ examples of a skill are demonstrated to the robot in slightly different situations. Each demonstration $m \in \{1,\dots,M\}$ consists of a set of $T_m$ $D$-dimensional positions $x = \{x_t\}_{t=1}^{T_m}$ and velocities $\dot{x} = \{\dot{x}_t\}_{t=1}^{T_m}$. The dataset is thus composed of a set of datapoints $\{x, \dot{x}\}$, where the joint distribution $\mathcal{P}(x, \dot{x})$ is encoded in a continuous *Hidden Markov Model* (HMM) of $K$ states. The output distribution of each state is represented by a Gaussian locally encoding variation and correlation information. The parameters of the HMM are defined by $\{\Pi, a, \mu, \Sigma\}$ and learned through *Baum-Welch* algorithm [14], which is a variant of *Expectation-Maximization* (EM) algorithm. $\Pi_i$ is the initial probability of being in state $i$, $a_{ij}$ is the transitional probability from state $i$ to state $j$. $\mu_i$ and $\Sigma_i$ represent the center and the covariance matrix of the $i$-th Gaussian distribution of the HMM. Input and output components in each state of the HMM are defined as

$$\mu_i = \begin{bmatrix} \mu_i^x \\ \mu_i^{\dot{x}} \end{bmatrix} \quad \text{and} \quad \Sigma_i = \begin{bmatrix} \Sigma_i^x & \Sigma_i^{x\dot{x}} \\ \Sigma_i^{\dot{x}x} & \Sigma_i^{\dot{x}} \end{bmatrix},$$

with $i \in \{1,\dots,K\}$, and where the indices $^x$ and $^{\dot{x}}$ refer respectively to position and velocity components.

With this representation, an unstable estimate of the motion dynamics can be estimated through *Gaussian Mixture Regression* (GMR). At each iteration a desired velocity command is estimated as

$$\hat{\dot{x}} = \sum_{i=1}^{K} h_i(x) \left[\mu_i^{\dot{x}} + \Sigma_i^{\dot{x}x}(\Sigma_i^x)^{-1}(x - \mu_i^x)\right]. \qquad (1)$$

Given the current position, a velocity command is estimated iteratively to control the system, see [15] for details. In the original GMR framework [10], the influence of the different Gaussians is represented by weights $h_i \in \mathbb{R}_{[0,1]}$, defined as the probability of an observed input belonging to each of the Gaussians. We propose to extend this estimation by recursively computing a likelihood through the HMM representation, thus taking into consideration not only the spatial information but also the sequential information probabilistically encapsulated in the HMM[1]

$$h_i(x_t) = \frac{\left(\sum_{j=1}^{K} h_j(x_{t-1})\ a_{ji}\right) \mathcal{N}(x_t;\ \mu_i^x, \Sigma_i^x)}{\sum_{k=1}^{K}\left[\left(\sum_{j=1}^{K} h_j(x_{t-1})\ a_{jk}\right) \mathcal{N}(x_t;\ \mu_k^x, \Sigma_k^x)\right]}.$$

Here, $h_i(x_t)$ represents the HMM *forward* variable [14], initialized with $h_i(x_1) = \frac{\pi_i \mathcal{N}(x_1;\, \mu_i^x, \Sigma_i^x)}{\sum_{k=1}^{K}\left[\pi_k \mathcal{N}(x_1;\, \mu_k^x, \Sigma_k^x)\right]}$, and corresponding to the probability of observing the partial sequence $\{x_1, x_2, \ldots, x_t\}$ and of being in state $i$ at time $t$.

To tackle the inherent instabilities of the model in Eq. (1), we add a secondary term that acts as a stabilizer on the proposed dynamical system. The stabilizer takes the form of a mass-spring-damper system that brings back the trajectories toward the centers of the Gaussians. Each of these centers acts as a "transient attractor" to this secondary system, hence driving the motion along the way. At each time step, a target velocity and target position are retrieved from our estimate of the dynamics of motion, following $\hat{\dot{x}} = \sum_{i=1}^{K} h_i(x) P(\dot{x}|x,i)$ as in (1) and $\hat{x} = \sum_{i=1}^{K} h_i(x) P(x|\dot{x},i)$.

Tracking of the desired velocity $\hat{\dot{x}}$ and desired position $\hat{x}$ is then insured by the proportional-derivative controller

$$\ddot{x} = \overbrace{(\hat{\dot{x}} - \dot{x})\kappa^{\mathcal{V}}}^{\ddot{x}^{\mathcal{V}}} + \overbrace{(\hat{x} - x)\kappa^{\mathcal{P}}}^{\ddot{x}^{\mathcal{P}}}, \qquad (2)$$

where $\kappa^{\mathcal{V}}$ and $\kappa^{\mathcal{P}}$ are gain parameters similar to damping and stiffness factors.

Note that the stabilizer may distort the original estimate of the dynamics (oscillations around the original demonstrations). Avoiding such oscillations and minimizing the distortions depends on choosing carefully the gains parameters. Finally, note that the addition of the stabilizing system above does not ensure stability of the complete system. However, in practice, for the experiments reported here (and for well chosen gains), the behavior of the system followed the desired dynamics. Analysis and solutions to the problem of stabilizing the first order system can be found in [16].

To avoid the distortion of the demonstrated dynamics, we define in further experiments the velocity gain $\kappa^{\mathcal{V}}$ in (2) such that, when $\kappa^{\mathcal{P}} \sim 0$, the system follows, after integration, the same velocity commands as for the simple version of the model. On the one hand, the proportional gain $\kappa^{\mathcal{P}}$ should be modulated such that the secondary system takes over in the face of a large perturbation, sending back the system toward the originally planned trajectory. On the other hand, this gain should not be too high to avoid that the system comes back to the trajectory and stops moving, instead of following the remainder of the motion. We thus define $\kappa^{\mathcal{P}} \in \mathbb{R}_{[0,\kappa^{\mathcal{P}}_{\max}]}$ as an adaptive gain that rapidly grows as the system departs from the area covered by the demonstrations, and remains null when the system is close to the demonstrations. We define $\kappa^{\mathcal{V}}$ and $\kappa^{\mathcal{P}}$ as

$$\kappa^{\mathcal{V}} = \frac{1}{\tau}\ , \quad \kappa^{\mathcal{P}} = \kappa^{\mathcal{P}}_{\max}\frac{\mathcal{L}_{\max} - \mathcal{L}(x)}{\mathcal{L}_{\max} - \mathcal{L}_{\min}}, \qquad (3)$$

$$\text{where} \quad \mathcal{L}_{\max} = \max_{i\in\{1,K\}} \log\left(\mathcal{N}(\mu_i^x; \mu_i^x, \Sigma_i^x)\right),$$

$$\mathcal{L}_{\min} = \min_{x\in\mathcal{W}, i\in\{1,K\}} \log\left(\mathcal{N}(x; \mu_i^x, \Sigma_i^x)\right).$$

In the above equation, $\mathcal{L}$ represents a log-likelihood.[2] $\kappa^{\mathcal{P}}_{\max}$ is the maximum gain allowed to attain a target position.[3] $\mathcal{W}$ defines the robot's workspace or a predetermined range of situations fixed a priori for the reproduction attempts. $\tau$ is the duration of an iteration step. At each iteration, $\kappa^{\mathcal{P}}$ is thus close to zero if $x$ is close to the Gaussian distributions. In this situation, the controller reproduces a motion with velocities similar to those in the demonstration sequences. On the other hand, if $x$ is far from the areas of demonstrations, the system comes back towards the closest Gaussians (in a likelihood sense) with a maximum gain of $\kappa^{\mathcal{P}}_{\max}$, still following the pattern of motion determined by $\hat{\dot{x}}$ in this region. By using $\ddot{x}^{\mathcal{V}}$ and $\ddot{x}^{\mathcal{P}}$ concurrently, the movement is thus distorted only very slightly, as $\ddot{x}^{\mathcal{P}}$ disappears when the system gets close to the demonstrated trajectories.

Parts of the movement where the variations across the demonstrations are high indicate that the position does not need to be tracked very precisely. This allows the controller to focus on the other constraints of the task, such as following a desired velocity. On the other hand, parts of the movement exhibiting strong invariance across the demonstrations will be tracked more precisely, i.e. the gain controlling the error on position will automatically be increased.

## III. EVALUATION THROUGH GENERATED DATA

To analyze systematically the proposed system, several sets of trajectories are randomly generated. First, a set of of $D$-dimensional keypoints $X$ is created (each variable $\{X_i\}_{i=1}^{D}$ is generated with a uniform random distribution $\mathcal{U}(0,1)$). A *Vector Integration To Endpoint* (VITE) system, which has been suggested as a biologically plausible model of human reaching movement [17], is then used to generate trajectories by starting from a first keypoint and recursively defining the next keypoint as the target. It is defined here as a critically damped mass-spring-damper controller $\ddot{x} = (X - x)\kappa^{\mathcal{P}} - \dot{x}\kappa^{\mathcal{V}}$ with parameters $\kappa^{\mathcal{V}} = 25$, $\kappa^{\mathcal{P}} = (\kappa^{\mathcal{V}})^2/4$, and time step $\tau = 0.003$ sec. Every 50 iterations, the target is switched to the next keypoint. For the last keypoint, 50 additional iterations are used to let the system converge to the last keypoint. To simulate motion variability, each dataset consists of 3 trajectories produced by slightly varying, with

[1] We will omit the index $t$ in further equations.

[2] Note that the log-likelihood measures correspond here to weighted distance measures.

[3] $\kappa^{\mathcal{P}}_{\max} = 2000$ has been fixed empirically in the experiments presented here.

a Gaussian noise $\mathcal{N}(0, 0.1)$, the positions of the keypoints. The resulting trajectories present natural looking motions that share similarities with those of humans. The automation of the generation process allows us to flexibly evaluate the imitation performance of our algorithm with respect to several datasets of different dimensionalities. An extensive description of this evaluation can also be found in [18].

### A. Comparison with other approaches

The approach that we propose in this paper will be further denoted as **HMM**, as its core representation is based on Hidden Markov Model. We compare this approach with four state-of-the-art methods that have proved good performance in robotics applications.

**TGMR**: *Time-dependent Gaussian Mixture Regression* [5] is based on our previous work, where time is used as an explicit input variable. The demonstrations are first aligned in time through *Dynamic Time Warping* (DTW), see [5] for details. Then, the distribution of temporal and spatial variables $\{t, x, \dot{x}\}$ is encoded in a *Gaussian Mixture Model* (GMM). At each time step during the reproduction process, a desired position $\hat{x}$ and a desired velocity $\hat{\dot{x}}$ are then retrieved through GMR by estimating $P(x, \dot{x}|t)$. The controller used by the robot to reproduce the skill is the mass-spring damper system defined in Eq. (2).

**LWR**: *Locally Weighted Regression* [11] is a memory-based probabilistic approach. It is used here to estimate at each time step a desired position $\hat{x}$ and a desired velocity $\hat{\dot{x}}$. Each datapoint of the dataset participates in the estimation of the solution by using a Gaussian kernel with fixed diagonal covariance matrix centered at the current position to weight the influence of each datapoint. The controller used by the robot is the mass-spring damper system defined in Eq. (2).

**LWPR**: *Locally Weighted Projection Regression* is an incremental regression algorithm that performs piecewise linear function approximation [13]. The algorithm does not require storage of the training data and has been proved to be efficient in a variety of robot learning tasks including high dimensional data. We use here an implementation of LWPR with the input space defined by a set of receptive fields with full covariance matrices. By detecting locally redundant or irrelevant input dimensions, the method locally reduces the dimensionality of the input data by finding local projections through *Partial Least Squares* (PLS) regression. The learning parameters have been set based on the recommendations provided in [13]. During reproduction, LWPR is used at each iteration to estimate a desired velocity $\hat{\dot{x}}$, given the current position $x$. The receptive fields are then used to determine a desired position $\hat{x}$ in a similar manner to the methods above. The controller used by the robot is the mass-spring damper system defined in Eq. (2).

**DMP**: The *Dynamic Movement Primitives* approach was originally proposed by Ijspeert *et al* [8]. The method allows a target to be reached by modulating a set of mass-spring-damper systems. This allows a particular path to be followed with the guarantee that the velocity vanishes at the end of the movement. A phase variable acts as a decay term to ensure that the system asymptotically converges to a reaching point. A reformulation of DMP similar to the one described in [19] is used in the experiment, see [20] for details.

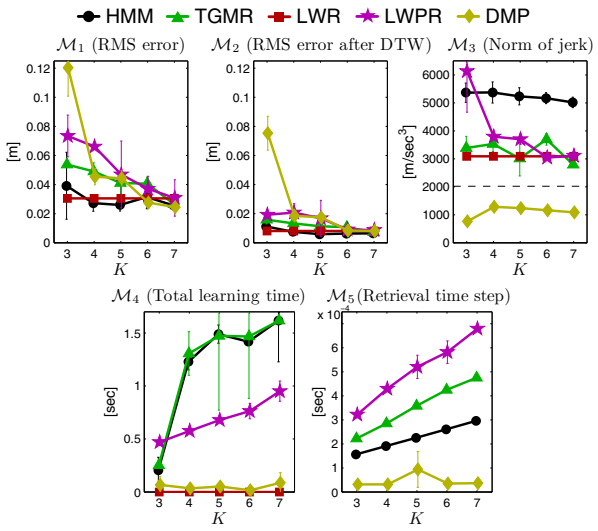


Fig. 1. Influence of the number of states $K$ on the metrics, for $D = 7$ dimensions. The dashed line in $\mathcal{M}_3$ represents the mean RMS jerk of the demonstrations.

### B. Metrics of imitation performance

Five metrics are used to evaluate a reproduction attempt $x' \in \mathbb{R}^{(D\times T)}$ with respect to the set of demonstrations $x \in \mathbb{R}^{(D\times M\times T)}$, rescaled in time with $T = \frac{\sum_{m=1}^{M} T_m}{M}$.

**RMS error** $\mathcal{M}_1$: This metric evaluates the accuracy of the reproduction in terms of spatial and temporal information, where a *root-mean-square* (RMS) error on position (with respect to the $M = 3$ demonstrations of the dataset) is computed along the reproduced motion $\mathcal{M}_1 = \frac{1}{MT}\sum_{m=1}^{M}\sum_{t=1}^{T}||x'_t - x_{m,t}||$.

**RMS error after DTW** $\mathcal{M}_2$: For this metric, the reproduced motion is first temporally aligned with the demonstrations through *Dynamic Time Warping* (DTW) [5], and a RMS error on position similar to $\mathcal{M}_1$ is then computed.

**Norm of jerk** $\mathcal{M}_3$: This metric evaluates the smoothness of the reproduction, based on RMS jerk quantification $\mathcal{M}_3 = \frac{1}{T}\sum_{t=1}^{T}||\dddot{x}'_t||$.

**Learning time** $\mathcal{M}_4$: Computation time of the learning process.

**Retrieval duration** $\mathcal{M}_5$: Computation time of the retrieval process for one iteration. [4]

### C. Evaluation results

Three different sets of movements are generated with the approach presented above. For each set of movements, three reproduction attempts are performed. This process is then

[4] $\mathcal{M}_4$ and $\mathcal{M}_5$ are evaluated through non-optimized Matlab implementations of the algorithms running on a 2.5GHz Pentium processor. The aim here is to provide information on the range of values and scaling properties that one can expect from the various learning and reproduction processes. The standard versions of the algorithms have been used, but it would be possible to adapt each algorithm to make it run faster.

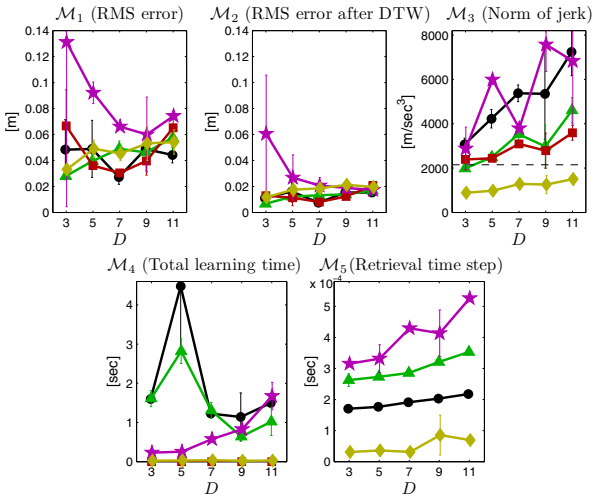


Fig. 2. Influence of the dimensionality $D$ of the dataset on the metrics, for $K = 4$ states (see Fig. 1 for legend).

repeated for various number of states, dimensionalities and ranges of perturbation.

Fig. 1 shows the influence of the number of states $K$ in the model (or basis functions), for the different methods (see Sec. III-A) and metrics (see Sec. III-B). As LWPR is an online incremental learning method, the parameter that determines when new basis functions are created (parameter $w_{\text{gen}}$ in [13]) has been gradually increased until the number of receptive fields matches the desired number of states.[5] We see with $\mathcal{M}_1$ and $\mathcal{M}_2$ in Fig. 1 that all methods perform very well, accurately following the demonstrated movements in terms of RMS errors. By encapsulating correlation information across input and output variables, HMM performs well with a very small number of states.

We see with $\mathcal{M}_3$ in Fig. 1 that DMP reproduces the smoothest movement (it actually smoothes the original demonstrations, see RMS jerk depicted in dashed line). It is noticeable that smoothness is not much affected by the number of states in general. For $\mathcal{M}_4$, DMP and LWR show the best performance in terms of the computation time used by the learning process (LWR is zero as it is a data-driven approach without learning), while HMM and TGMR (both trained by *Expectation-Maximization*) show a slightly worse performance. For a better comparison with the online learning nature of LWPR, 10 passes have been performed with the dataset shuffled randomly. It should thus be noted that by using a single pass, the computation time can be reduced by an order of magnitude. For $\mathcal{M}_4$, the computation time used by LWR for reproduction is not competitive and is thus not depicted here (it goes over $7 \times 10^{-2}$ sec. as in the proposed implementation, each datapoint contributes to the estimation). The other approaches show a linear dependency on the number of states and are suitable for online application in robotics (less than 1 millisecond per iteration for the considered number of states).

[5] $\mathcal{M}_4$ computation time is evaluated by taking the last learning session into consideration.

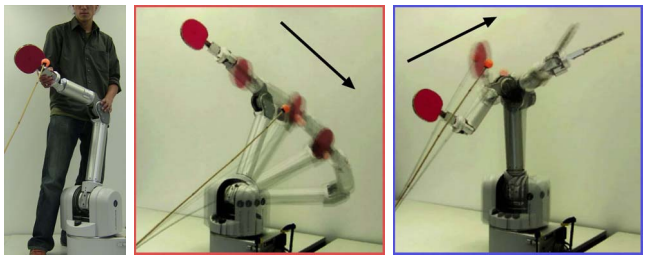

Fig. 3. *Left:* Experimental setup for the experiment of teaching the *Barrett WAM* robotic arm to hit a ball. *Center:* Reproduction of a *drive* stroke. *Right:* Reproduction of a *topspin* stroke.

Fig. 2 shows the influence of the dimensionality $D$ on the metrics for the different approaches (see legend in Fig. 1), when considering $K = 4$ states in the model. We see with $\mathcal{M}_1$ and $\mathcal{M}_2$ that the methods perform equally well in terms of RMS errors. When the dimensionality is low, the difficulty is to correctly handle the crossing points that can appear when randomly generating trajectories (i.e., when passing through the same point several times during a demonstration). When the dimensionality is high, these crossings are less likely to occur. However, the difficulty is in this case to efficiently handle the sparsity of the data (curse of dimensionality). This fact is reflected by the data, and is especially noticeable for LWR. The comparison with LWPR is not very informative here, as the lower performance is related to the online nature of the learning process. Indeed, an online algorithm cannot determine in advance whether loops in the motion will be encountered, while a batch learning process can cluster the crossing points more easily.

For $\mathcal{M}_4$ in Fig. 2, we see that the computation time of *Expectation-Maximization* (EM) used by HMM and TGMR produces very variable results. Indeed, EM is a local search procedure that starts randomly (with *k-means* initialization), and stops, for example, once a local maximum likelihood is reached (other stopping criterions can be defined). Depending on the initialization, a very different number of iterations may be required to reach a local optimum. For example, in low dimensions, the local optimum may not be trivial to find, as crossings are more likely to occur in the motion. Here, a single initialization for the search has been fixed, and no constraint has been fixed on the number of iterations, which may explain the high computation time of nearly 5 sec.[6] required by EM to learn the dataset generated for $D = 5$. For reproduction, $\mathcal{M}_5$ shows that the different methods remain competitive in terms of online retrieval of data (less than 1 millisecond, and nearly linear increase for dimensions below $D = 12$).

## IV. EXPERIMENT WITH ROBOTIC ARM

### A. Experimental setup

The experiment consists of learning and reproducing the motion of hitting a ball with a table tennis racket by using

[6] In practice, a maximum number of iterations can be set (which was not the case in this experiment) to guarantee that the learning time remains short.

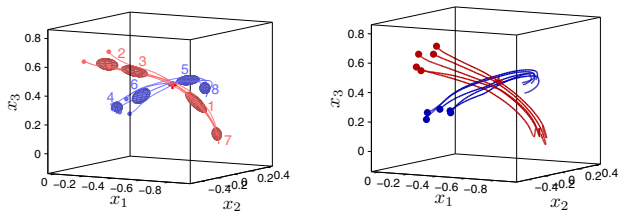


Fig. 4. Encoding and reproduction results of the table tennis experiment. *Left:* Demonstrated movements and associated Hidden Markov Model, where 8 Gaussians are used to encode the two categories of movements. The position of the ball is depicted by a plus sign, and the initial points of the trajectories are depicted by points. The trajectories corresponding to *topspin* and *drive* strokes have been represented in different colors for visualization purpose, but the robot does not have this information and is also not aware of the number of categories that has been demonstrated. *Right:* 10 reproduction attempts by starting from new random positions in the areas where either *topspin* and *drive* strokes have been demonstrated.

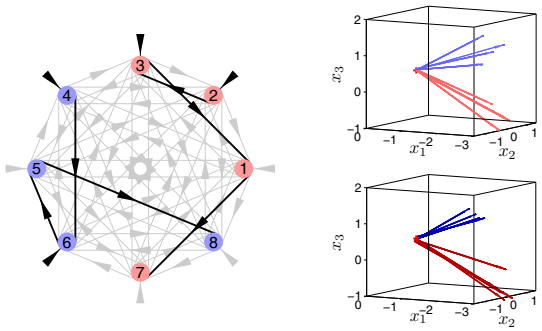


Fig. 5. *Left:* HMM representation of the transitions and initial state probabilities (the corresponding state output distributions are represented in Fig. 4). Probabilities above 0.1 are represented by black lines (self-transitions are not represented here), showing two different sequences, defined by states transitions 2-3-1-7 and 4-6-5-8. *Right:* Position and velocity of the racket at the time of the impact with the ball for the 8 demonstrations (*top*) and for the 10 reproduction attempts (*bottom*).

a *Barrett WAM* 7 DOFs robotic arm, see Fig. 3 *left*. One objective is to demonstrate that such movements can be transferred using the proposed approach, where the skill requires that the target be reached with a given velocity, direction and amplitude. In the experiment presented here, we extend the difficulty of the tennis task described in [8] by assuming that the robot must hit the ball with a desired velocity set by the demonstrations. The robot thus hits the ball, continues its motion and stops, instead of reaching it with zero velocity.

In table tennis, *topspin* occurs when the top of the ball is going in the same direction as the ball is moving. Topspin causes the ball to drop faster than by gravity alone, and is used by players to allow the ball to be hit harder but still land on the table. The stroke with no spin is referred to as *drive*. The motion and orientation of the racket at the impact thus differ when performing a *topspin* or a *drive* stroke. Training was done by an intermediate-level player demonstrating several *topspin* and *drive* strokes to the robot by putting it in an active gravity compensation control mode, which allows the user to move the robot manually. Through this *kinesthetic teaching* process, the user *molds* the robot behavior by putting it through the task of hitting the ball with a desired spin. The ball is fixed on a stick during demonstration, and its initial position is tracked by a color-based stereoscopic vision tracking system.

The recordings are performed in Cartesian space by considering the position $x$ and orientation $q$ of the racket with respect to the ball, with associated velocities $\dot{x}$ and $\dot{q}$. A quaternion representation of the orientation is used, where three of the four quaternion components are used (the fourth quaternion component is reconstructed afterwards). The user demonstrates in total 4 *topspin* strokes and 4 *drive* strokes in random order. The categories of strokes are not provided to the robot, and the number of states in the HMM is selected through *Bayesian Information Criterion* (BIC), see [5] for details.

### B. Experimental results

Fig. 4 presents the encoding and reproduction results for the positions $x$ in Cartesian space. We see that the HMM model reproduces an appropriate motion in the two situations. Fig. 5 *left*, presents the states transitions learned by the HMM. We see that the model has correctly learned that two different dynamics can be achieved here, depending on the initial position of the robot, thus correctly generalizing the skill. Fig. 5 *right* presents the results at the time of the impact. We see that the system correctly strikes the ball at a velocity similar to the ones demonstrated (in terms of both amplitude and directions). The video of this experiment and the sourcecodes of the proposed approach are available online [21].

## V. DISCUSSION

The evaluation presented in Sec. III showed that HMM is competitive with respect to state-of-the-art approaches in robot learning by observation. However, the evaluation remains valid only for the specific context presented here, i.e., by determining an acceleration command recursively after having observed a set of position and velocity data. It does not necessary reflect the efficiency of the algorithms in other contexts.

The proposed HMM approach shares many characteristics with the DMP approach, but has some advantages for the subset of tasks that we considered. First, it is able to encode several motion alternatives in the same model. Partial demonstrations can be provided, which is a very important advantage for the teaching interaction (e.g., to refine one part of the movement without having to demonstrate the whole task again). Compared to DMP that must explicitly embed the cyclic or discrete form of the motion, the HMM approach allows periodic and reaching movements to be handled in a unified way (and simultaneously), without having to specify the representation beforehand.

For some tasks, DMP requires a heuristic to be defined to recompute the value of the canonical variable $s$ (re-parametrization of time through a decay term), see [19], [20] for details. Indeed, DMP is robust to spatial perturbation, but requires an external mechanism to handle temporal perturbations such as delay and pauses in the motion. Thus,

the perturbation needs to be detected in order to re-estimate the value of the decay term $s$. For example, if the robot needs to reproduce only one part of the motion, or if the target is moving, $s$ must be re-evaluated in consequence. Handling this type of perturbation is in contrast inherently encapsulated in the proposed model, without parametrization of a temporal decay (spatial and temporal distortions are handled through the HMM representation).

As demonstrated in the robotic application, the proposed HMM approach is not constrained to movements with a unique zero-velocity attractor point. A single model is used to encode multivariate data, which allows automatic learning of the correlations between the different variables, and the use of this information for reproduction. To handle multivariate data, DMP considers the different variables as separate processes synchronized by the phase variable, while HMM encapsulates the complete correlation information, which may be an advantage for some tasks. The covariance matrices given by GMR provide local information on the spread of each center $\mu_i$. This therefore allows building a regression estimate even if a low number of Gaussians is considered.

These appealing properties however comes with a drawback that requires further investigation. In DMP, the weights are determined through a decay term, which allows the system to guarantee convergence to the last target point. In contrast, the HMM method has the disadvantage that its stability relies on the proper choice of the gains in Eq. (2). These gains must be set by estimating in advance the perturbations that one can expect during reproduction and/or the range of novel initial positions that the system is expected to handle. Moreover, even if each subsystem is stable, the stability of the complete system is not necessary guaranteed, and it remains difficult to find criterions that are valid for both discrete and periodic motion. Analyzing the stability of such systems, and deriving the associated learning algorithms are part of future work [16].

## VI. CONCLUSION

We presented and evaluated a method based on Hidden Markov Model (HMM) and Gaussian Mixture Regression (GMR) to allow robots to acquire new movements by imitation. The use of GMR was extended to a dynamical system approach in order to get rid of the explicit time dependency that was considered in our previous work [5]. We also extended the regression process by taking advantages of the HMM capability to represent robustly sequential motion. For the context of separate learning and reproduction processes, this novel formulation was systematically evaluated with respect to our previous approach, as well as to *Locally Weighted Regression* (LWR) [11], *Locally Weighted Projection Regression* (LWPR) [13], and *Dynamic Movement Primitives* (DMP) [8], [19]. The approach was finally illustrated through an experiment where a robot learned new skills through kinesthetic teaching, where we showed that the proposed framework can be efficiently used in an unsupervised learning mode.

## REFERENCES


[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Secaucus, NJ, USA: Springer, 2008, pp. 1371–1394.

[2] S. Calinon, *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press, 2009, EPFL Press ISBN 978-2-940222-31-5, CRC Press ISBN 978-1-4398-0867-2.

[3] J. Aleotti and S. Caselli, "Robust trajectory learning and approximation for robot programming by demonstration," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 409–413, 2006.

[4] A. Ude, "Trajectory generation from noisy positions of object features for teaching robot paths," *Robotics and Autonomous Systems*, vol. 11, no. 2, pp. 113–127, 1993.

[5] S. Calinon and A. Billard, "Statistical learning by imitation of competing constraints in joint space and task space," *Advanced Robotics*, vol. 23, no. 15, pp. 2059–2076, 2009.

[6] M. Muehlig, M. Gienger, S. Hellbach, J. Steil, and C. Goerick, "Task-level imitation learning using variance-based movement optimization," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2009.

[7] D. Grimes, R. Chalodhorn, and R. Rao, "Dynamic imitation in a humanoid robot through nonparametric probabilistic inference," in *Proc. Robotics: Science and Systems (RSS)*, 2006, pp. 1–8.

[8] A. Ijspeert, J. Nakanishi, and S. Schaal, "Trajectory formation for imitation with nonlinear dynamical systems," in *Proc. IEEE Intl Conf. on Intelligent Robots and Systems (IROS)*, 2001, pp. 752–757.

[9] D. Kulic, W. Takano, and Y. Nakamura, "Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains," *Intl Journal of Robotics Research*, vol. 27, no. 7, pp. 761–784, 2008.

[10] Z. Ghahramani and M. Jordan, "Supervised learning from incomplete data via an EM approach," in *Advances in Neural Information Processing Systems*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6. Morgan Kaufmann Publishers, Inc., 1994, pp. 120–127.

[11] S. Schaal and C. Atkeson, "Constructive incremental learning from only local information," *Neural Computation*, vol. 10, no. 8, pp. 2047–2084, 1998.

[12] D. Nguyen-Tuong and J. Peters, "Local gaussian process regression for real-time model-based robot control," in *IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2008, pp. 380–385.

[13] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[14] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77:2, pp. 257–285, February 1989.

[15] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Trans. on Robotics*, vol. 24, no. 6, pp. 1463–1467, 2008.

[16] M. Khansari and A. Billard, "Learning stable non-linear dynamical systems with gaussian mixture models," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, Anchorage, Alaska, USA, May 2010.

[17] D. Bullock and S. Grossberg, "Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation," *Psychological Review*, vol. 95, no. 1, pp. 49–90, 1988.

[18] S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, and A. Billard, "A probabilistic approach based on dynamical systems to learn and reproduce gestures by imitation," *IEEE Robotics and Automation Magazine*, 2010, submitted.

[19] H. Hoffmann, P. Pastor, D. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2009, pp. 2587–2592.

[20] S. Calinon, F. D'halluin, D. Caldwell, and A. Billard, "Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework," in *Proc. IEEE-RAS Intl Conf. on Humanoid Robots (Humanoids)*, Paris, France, December 2009, pp. 582–588.

[21] S. Calinon, "Robot programming by demonstration: A probabilistic approach," `http://programming-by-demonstration.org`, February 2010.