A probabilistic approach based on dynamical systems to learn and reproduce gestures by imitation

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

***Abstract*—We present a probabilistic approach to learn robust models of human motion through imitation. The association of Hidden Markov Model (HMM), Gaussian Mixture Regression (GMR) and dynamical systems allows us to extract redundancies across multiple demonstrations and build time-independent models to reproduce the dynamics of the demonstrated movements. The approach is first systematically evaluated and compared with other approaches by using generated trajectories sharing similarities with human gestures. Three applications on different types of robots are then presented. An experiment with the iCub humanoid robot acquiring a bimanual dancing motion is first presented to show that the system can also handle cyclic motion. An experiment with a 7 DOFs WAM robotic arm learning the motion of hitting a ball with a table tennis racket is presented to highlight the possibility to encode several variations of a movement in a single model. Finally, an experiment with a HOAP-3 humanoid robot learning to manipulate a spoon to feed the Robota humanoid robot is presented to demonstrate the capability of the system to handle several constraints simultaneously.**

***Index Terms*—Robot programming by demonstration, Learning by imitation, Dynamical systems, Gaussian mixture regression, Hidden Markov Model.**

I. Introduction

ROBOT Programming by Demonstration (PbD) covers methods by which a robot learns new skills through human guidance. Also referred to as learning by imitation, lead-through teaching, tutelage or apprenticeship learning, PbD takes inspiration from the way humans learn new skills by imitation to develop methods by which new skills can be transmitted to a robot. PbD covers a broad range of applications. In industrial robotics, the goal is to reduce the time and costs required to program the robot. The rationale is that PbD would allow to modify an existing product, create several versions of a similar product or assemble new products in a very rapid way, and this could be done by lay users without help from an expert in robotics. PbD is perceived as particularly useful to service robots, i.e. robots deemed to work in direct collaboration with humans. In this case, methods for PbD go beyond transferring skills and offer new ways for the robot to interact with the human, from being capable of recognizing people's motion to predicting their intention and seconding them in the accomplishment of complex tasks. As the technology improved to provide these robots with more and more complex hardware, including multiple sensor modalities and numerous degrees of freedom, robot control and especially robot learning became more and more complex too.

Learning control strategies for numerous degrees of freedom platforms deemed to interact in complex and variable environments, such as households is faced with two key challenges: first, the complexity of the tasks to be learned is such that pure trial and error learning would be too slow. PbD appears thus a good approach to speed up learning by reducing the search space, while still allowing the robot to refine its model of the demonstration through trial and error [1], [2]. Second, there should be a continuum between learning and control, so that control strategies can adapt in real time to drastic changes in the environment. The present work addresses both challenges in investigating and comparing methods by which PbD is used to learn the dynamics of demonstrated movements, and, by doing so, provide the robot with a generic and adaptive model of control.

A. Related work and motivations

PbD is of interest for different levels of task representation. A large body of work in PbD follow a symbolic approach to representing and encoding the tasks, see e.g. [3]–[8]. Such a symbolic description offers the advantage that it provides a way to easily tackle sequences or hierarchies of actions. One major drawback however lies in that they rely on a large amount of prior knowledge to predefine the important cues and to segment those efficiently.

Most approaches to trajectory modeling estimate a time-dependent model of the trajectories, by either exploiting variants along the concept of spline decomposition [9]–[11] or through an explicit encoding of the time-space dependencies [12]. Such modeling methods are 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 spatial and temporal perturbations. As an alternative, other approaches have considered modeling the intrinsic dynamics of motion [13]–[18]. 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.

To embed multivariate data exhibiting temporal coherence such as human motion, a variety of approaches have been proposed, ranging from *Hidden Markov Model* (HMM) [19] to *spatio-temporal Isomap* (ST-Isomap) [20]. We use HMM

[†]Advanced Robotics Department, Italian Institute of Technology (IIT), 16163 Genova, Italy. name.surname@iit.it.

[‡]Learning Algorithms and Systems Laboratory (LASA), Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland. name.surname@epfl.ch.

in this work, which has previously been reported as a robust probabilistic method to deal with the spatial and temporal variabilities of human motion across various demonstrations [14], [17], [18]. Most of the approaches proposed sofar however require either a high number of states to reproduce the motion correctly (i.e. higher than for recognition purposes), or an additional smoothing process whose drawback is to cut down important peaks in the motion.

The proposed model also relies on *Gaussian Mixture Regression* (GMR) [21] to robustly generalize the motion during reproduction. The approach is contrasted with our previous work that employed GMR with time being considered as an explicit input variable [12]. We demonstrated in previous work that this framework can be used to learn a skill incrementally (without having to keep each demonstration in memory) [22]. We also showed that it allows simultaneous consideration of constraints in joint space and task space [23]. Muehlig *et al* [24] recently extended the GMR approach to learn bimanual skills by imitation. In this work, the authors used GMR as a compact probabilistic representation of the task constraints which is then used during reproduction by a gradient-based trajectory optimizer. This demonstrates that the generic formulation of GMR can be efficiently combined with optimal control methods.

In opposite to other statistical regression methods such as *Locally Weighted Regression* (LWR) [25], *Locally Weighted Projection Regression* (LWPR) [26], or *Gaussian Process Regression* (GPR) [16], [27], 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 joint density model [28].

This is an advantage in many robotic applications since the input and output components are only specified at the very last step of the algorithm. Density estimation can thus be learned in an offline phase, while the regression process can be computed very rapidly. It can also handle different sources of missing data, as the system is able to consider any combination of input/output mappings during the retrieval phase.

In the context of robot learning by imitation, the principal advantages of combining HMM and GMR are thus: (1) it allows us to deal with recognition and reproduction issues in a common probabilistic framework; and (2) the learning process is distinct from the retrieval process, where a standard *Expectation-Maximization* (EM) algorithm is first used to learn the demonstrated skill during the phases of the interaction that do not require real-time computation (i.e. after the demonstrations), and where a faster regression process is then used for controlling the robot in an online manner during the reproduction phases.

The remainder of the paper is organized as follows. Sec. II presents the probabilistic approach. Sec. III evaluates and discusses the proposed approach with respect to 4 state-of-the-art approaches in robot learning by imitation. Sec. IV presents an experiment where the iCub humanoid robot learns a periodic bimanual gesture through the use of motion sensors. Sec. V presents an experiment where the WAM robotic arm learns two different ways of striking a ball in table tennis through kinesthetic teaching. Sec. VI finally presents an experiment where the HOAP-3 humanoid robot learns a feeding task requiring to consider several constraints and landmarks.

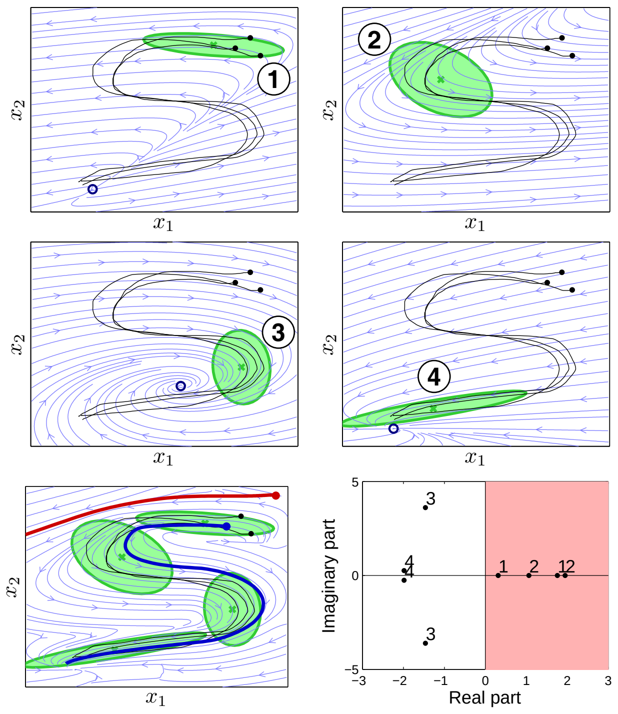

Fig. 1. Example of motion encoding and reproduction using the basic control model.

II. Proposed probabilistic approach

Multiple examples of a skill are demonstrated to the robot in slightly different situations, where a set of positions $x \in \mathbb{R}^{(D\times M\times T)}$ and velocities $\dot{x} \in \mathbb{R}^{(D\times M\times T)}$ are collected during the demonstrations (D is the dimensionality of the variable x, M is the number of demonstrations, and T is the length of a demonstration). The dataset is 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 [19], which is a variant of *Expectation-Maximization* (EM) algorithm. Π_i is the initial probability of being in state i, a_{ij} is the transitional probability from state i to state j. μ_i and Σ_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

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

where the indices ^{x} and $^{\dot{x}}$ refer respectively to position and velocity components.

A. Control model - Basic version

In the basic control model, a desired velocity $\hat{\dot{x}}$ is estimated through *Gaussian Mixture Regression* (GMR) as

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

which is used to control the system by estimating at each iteration a new velocity given the current position, see [29] for details.

In the GMR framework, the influence of the different Gaussians is represented by weights $h_i \in [0,1]$, originally defined as the probabilities of an observed input to belong to each of the Gaussians [21]

$$h_i(x) = \mathcal{N}(x;\ \mu_i^x, \Sigma_i^x),$$

and normalized such that $\sum_i^K h_i = 1$.

A direct extension of this estimation is to recursively compute 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

$$h_{i,t}(x) = \left(\sum_{j=1}^{K} h_{j,t-1}\ a_{ji} \right) \mathcal{N}(x;\ \mu_i^x, \Sigma_i^x),$$

and normalizing such that $\sum_i^K h_{i,t} = 1$. Here, $h_{i,t}$ represents the *forward* variable [19], which corresponds to the probability of observing the partial sequence $\{x_1, x_2, \ldots, x_t\}$ and of being in state i at time t.

At a given instant, the regression process described in (1) can be rewritten as a mixture of linear systems[1]

$$\dot{x} = \sum_{i=1}^{K} h_i (A'_i x + b'_i) \text{ with} \left| \begin{array}{l} A'_i = \Sigma_i^{\dot{x}x} (\Sigma_i^x)^{-1}, \\ b'_i = \mu_i^{\dot{x}} - \Sigma_i^{\dot{x}x} (\Sigma_i^x)^{-1} \mu_i^x. \end{array} \right. \qquad (2)$$

Fig. 1 presents an example of encoding and reproduction using this basic control scheme, where the number of states in the HMM has been deliberately fixed to a low value. The first four graphs show the dynamic behavior of the system when using each Gaussian separately, where the circles represent the equilibrium points defined by $-{A'}_i^{-1} b'_i$. The bottom-left graph shows results for two reproduction attempts represented by blue and red thick lines, where the initial positions are represented by points. The last graph shows the poles of the system, given by the eigenvalues of matrices A_i in (2). We observe in the first four graphs that each Gaussian representing the local distribution of $\{x, \dot{x}\}$ can retrieve curved trajectories (rotational fields induced by the poles).[2] The last graph shows that for the first two states, the poles have a real positive part, which may lead to unstable systems in some situations (i.e., the first two equilibrium points are unstable). By using the basic control method, the motion is correctly reproduced when starting in regions that have been covered during the demonstrations (trajectory represented by blue lines). The first two states of the system are unstable but bring the robot to asymptotically stable states after a few iterations. Note that the system in this basic version may provide poor solution when initiating the motion in a region that has not been covered yet (trajectory represented by red lines).

[1]Note that this representation is also similar to the Takagi-Sugeno fuzzy modeling technique [30].

[2]A condition for asymptotic stability is that the poles lie strictly in the closed left half of the complex plane (i.e. the real part of all the poles must be negative).

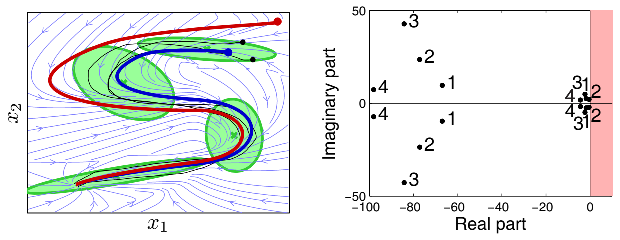

Fig. 2. Example of motion encoding and reproduction using the extended control model.

B. Control model - Extended version

For the reason mentioned above, we extended the basic control model with an acceleration-based controller similar to a mass-spring-damper system, where the model of the demonstrated trajectories acts as an attractor.[3] A target velocity $\hat{\dot{x}}$ and target position $\hat{x}$ are first estimated at each time step through GMR. Tracking of the desired velocity $\hat{\dot{x}}$ and desired position $\hat{x}$ is then insured by the proportional-derivative controller. The acceleration command is determined by[4]

$$\ddot{x} = \overbrace{(\hat{\dot{x}} - \dot{x})\kappa^{\mathcal{V}}}^{\ddot{x}^{\mathcal{V}}} + \overbrace{(\hat{x} - x)\kappa^{\mathcal{P}}}^{\ddot{x}^{\mathcal{P}}}, \qquad (3)$$

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

In the above equation, $\ddot{x}^{\mathcal{V}}$ allows the robot to follow the demonstrated velocity profile.[5] $\ddot{x}^{\mathcal{P}}$ prevents the robot to depart from a known situation, and forces it to come back to this observed subspace if a perturbation occurs. By using both terms concurrently, the robot follows the learned non-linear dynamics while tracking the movement. Similarly to (2), (3) can be formulated as a mixture of linear systems, see Appendix A.

Fig. 2 presents reproduction results with the extended control scheme, where the same dataset and HMM encoding as in Fig. 1 has been utilized. The left graph shows the two reproduction attempts where the robot smoothly comes back to the demonstrated movement when starting from a different initial situation. The right graph shows the poles of the corresponding linear systems (see Appendix A), consisting of four poles per Gaussian instead of two, as the system is now based on an acceleration command.

Here, we suggest to define the velocity gain $\kappa^{\mathcal{V}}$ in (3) such that, for low values of $\kappa^{\mathcal{P}}$, the model follows the motion with the same velocity profiles as the ones demonstrated in similar

[3]Sourcecode of the algorithm is available online [31].

[4]In the experiments presented here, velocity and position are updated at each iteration through Euler numerical integration.

[5]By setting $\kappa^{\mathcal{V}} = \frac{1}{\Delta t}$ and $\kappa^{\mathcal{P}} = 0$, the controller is similar to (1).

situations. The position gain $\kappa^{\mathcal{P}}$ can be modulated such that it acts as an attractor to the trajectory, which depends on the strength of the perturbation (or if the system needs to start from locations that have not been demonstrated yet). It should also not be too high to avoid that the system, acting only as an attractor, comes back to the trajectory and stops instead of following the remainder of the motion. We thus define $\kappa^{\mathcal{P}}$ as an adaptive gain that rapidly grows as the system departs from the area covered by the demonstrations, and is null when the system is close to the demonstrations. We define $\kappa^{\mathcal{V}}$ and $\kappa^{\mathcal{P}}$ as

$$\kappa^{\mathcal{V}} = \frac{1}{\Delta t}\;, \quad \kappa^{\mathcal{P}}(x) = \kappa^{\mathcal{P}}_{\max} \frac{\mathcal{L}_{\max} - \mathcal{L}(x)}{\mathcal{L}_{\max} - \mathcal{L}_{\min}}, \tag{4}$$

$$\text{where} \quad \mathcal{L}_{\max} = \max_{i\in\{1,K\}} \log\Big(\mathcal{N}(\mu_i^x;\mu_i^x,\Sigma_i^x)\Big),$$
$$\mathcal{L}_{\min} = \min_{\substack{i\in\{1,K\}\\ x\in\mathcal{W}}} \log\Big(\mathcal{N}(x;\mu_i^x,\Sigma_i^x)\Big).$$

In the above equation, $\mathcal{L}$ represents a log-likelihood.[6] $\kappa^{\mathcal{P}}_{\max}$ is the maximum gain allowed to attain a target position.[7] $\mathcal{W}$ defines the robot's workspace or a predetermined range of situations fixed a priori for the reproduction attempts. Δt is the duration of an iteration step. At each iteration, $\kappa^{\mathcal{P}}(x)$ 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 in this region (determined by $\hat{\dot{x}}$).

Parts of the movement where a strong inconsistency has been observed (i.e., where the variations in 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 position invariance across the multiple demonstrations will be tracked more precisely, i.e. the gain controlling the error on position will automatically be increased.

III. Evaluation through generated data

A. Generation of human-like motion data

To analyze systematically the proposed system, several sets of natural trajectories are created. First, a set of keypoints X of D dimensions is randomly generated (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 [32], 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 integration time step $\Delta t = 0.003$ 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 the positions of the keypoints with a Gaussian noise $\mathcal{N}(0, 0.1)$. An example of generated motion is presented in Fig. 3 *left*. The resulting trajectories present natural looking motion that share similarities with those of humans. The automatization of the generation process allows us to flexibly evaluate the imitation performance of our algorithm with respect to several datasets of different dimensionalities.

[6]Note that here, the log-likelihood measures corresponds to weighted distance measures.

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

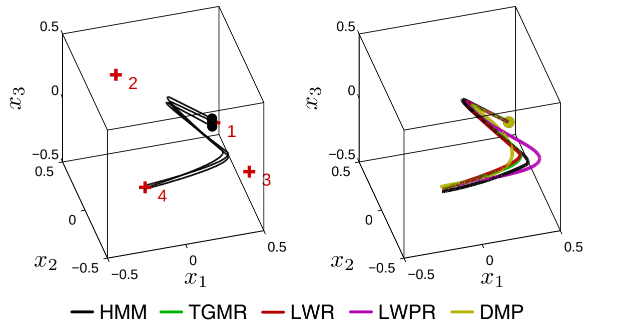

Fig. 3. *Left:* Example of a dataset of 3 dimensions randomly generated. The four plus signs represent the keypoints used to generate the continuous motion. *Right:* Reproductions by using the various methods proposed for comparison.

B. 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 alternative methods that have shown good performance in robotics experiments.

TGMR: *Time-dependent Gaussian Mixture Regression* [12] 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 [12] 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 (3).

LWR: *Locally Weighted Regression* [25] 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 (3).

LWPR: *Locally Weighted Projection Regression* is an incremental regression algorithm that performs piecewise linear function approximation [26]. The algorithm does not require to store 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 [33]. The learning parameters are fixed based on the recommendations provided in [26]. 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 $\hat{x}$ similarly as in the methods above. The controller used by the robot is the mass-spring damper system defined in (3).

DMP: The *Dynamic Movement Primitives* approach was originally proposed by Ijspeert *et al* [13], and further extended in [34], [35]. The method allows to reach a target by modulating a set of mass-spring-damper systems. This allows to follow a particular path 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 formulation of DMP similar to the one used for the HMM approach is detailed in Appendix B.

C. 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 $x \in \mathbb{R}^{(D\times M\times T)}$.

$\mathcal{M}_1$: This metric evaluates the generalization capability by measuring how well the reproduced trajectory matches the different demonstrations. It 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

$$\mathcal{M}_1 = \frac{1}{MT}\sum_{m=1}^{M}\sum_{t=1}^{T}||x'_t - x_{m,t}||.$$

$\mathcal{M}_2$: For this metric, the reproduced motion is first temporally aligned with the demonstrations through *Dynamic Time Warping* (DTW) [12], and a RMS error on position similar to $\mathcal{M}_1$ is then computed. In contrast with $\mathcal{M}_1$, spatial information is prioritized here (i.e., the metric compares the path followed by the robot instead of the exact trajectory along time). Depending on the skill that should be learned, metrics $\mathcal{M}_1$ and $\mathcal{M}_2$ have different importance. To reproduce a demonstrated motion from a distant initial position, it is sometimes desirable to first come back to the motion path, and then follow the motion (e.g., drawing an alphabet letter on a board requires $\mathcal{M}_2$ to be low). Other skills require to take into consideration the timing, although this may have a detrimental effect on the precision with which the path can be followed (e.g., intercepting a falling object requires $\mathcal{M}_1$ to be low). Indeed, the importance of the metric highly depends on the skill that one wants to transfer to the robot [36].

$\mathcal{M}_3$: This metric evaluates the smoothness of the reproduction based on RMS jerk quantification. This measure, based on the derivative of acceleration, has been shown to be a good candidate to evaluate smoothness of human motion [37]

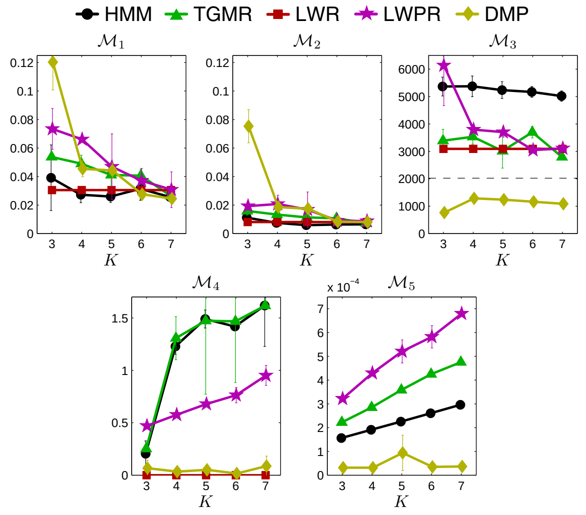

Fig. 4. Influence of the number of states K on the metrics, for $D = 7$ dimensions. The dashed line in $\mathcal{M}_3$ represents the mean RMS jerk of the demonstrations.

$$\mathcal{M}_3 = \frac{1}{T}\sum_{t=1}^{T}||\dddot{x}'_t||.$$

$\mathcal{M}_4$: Computation time (in seconds) of the learning process.

$\mathcal{M}_5$: Computation time (in seconds) of the retrieval process for one iteration.

$\mathcal{M}_4$ 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.[8]

We also evaluate the capability to handle external perturbations by generating a random force along the motion and superposing it with the acceleration computed in (3). Metrics $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ are then used to evaluate the reproduction attempts when faced with these perturbations. A continuous force is created by first generating a set of keypoints along the motion (with random time of occurrence and amplitude), and interpolating between these keypoints through a third-order spline fit. This process is similar to the Perlin noise originally proposed to generate naturally looking textures [38], and further extended to naturally looking perturbations in robot motion [39].

D. Evaluation results

Three different sets of movements are generated with the approach presented in Sec. III-A. For each set of movements, three reproduction attempts are performed. This process is then repeated for various number of states, dimensionalities and ranges of perturbation. Examples of reproduced trajectories are presented in Fig. 3 *right*. The quantitative results are presented in Figs 4-6.

[8]The standard versions of the algorithms have been used, but it would be possible to adapt each algorithm to make it run faster.

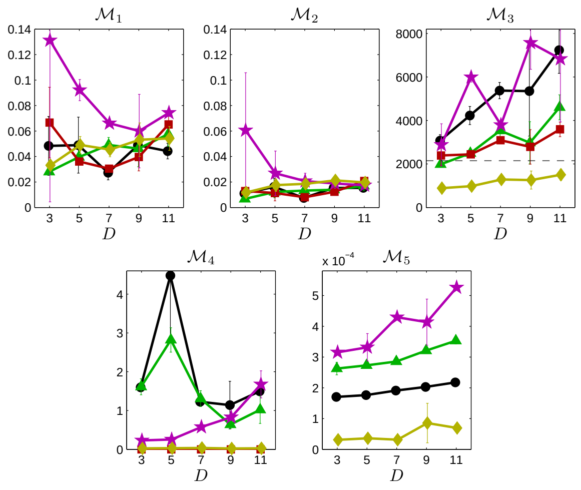

Fig. 5. Influence of the dimensionality D of the dataset on the metrics, for $K = 4$ states (see Fig. 4 for legend).

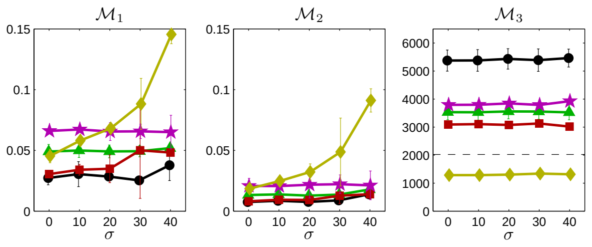

Fig. 6. Robustness to external perturbations, for $K = 4$ states and $D = 7$ dimensions (see Fig. 4 for legend).

Fig. 4 shows the influence of the number of states K in the model (or basis functions), for the different methods (see Sec. III-B) and metrics (see Sec. III-C). As LWPR is an online incremental learning method, the threshold that determines the minimum activation before a new basis function is created (parameter w_{gen} in [26]) has been gradually increased until the number of receptive fields matches the desired number of states.[9] We see with $\mathcal{M}_1$ and $\mathcal{M}_2$ in Fig. 4 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. DMP also shows a low RMS error but requires at least 4 primitives acting as attractors.

We see with $\mathcal{M}_3$ in Fig. 4 that DMP reproduces the smoothest movement (actually smoother than the original demonstrations with 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 bit worse performance. To cope with the online learning nature of LWPR, 10 passes have been performed on 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.

[9]For LWPR, the $\mathcal{M}_4$ computation time is evaluated by taking only the last learning step into consideration.

In this experiment, we concentrated on a case where the learning process is separated from the retrieval process. In this context, both a batch learning process and an online learning process can be employed. The computation time needed for learning also has less importance than the one required for real-time reproduction of a skill. The most important aspect here is that the user should not wait too long for the robot to update its model of the skill after arrival of each new demonstration. In Fig. 4, as all the methods learned in less than 2 sec., the computation time remains acceptable for an efficient teaching interaction. For $\mathcal{M}_4$, the computation time used by LWR for reproduction is not competitive and is thus not depicted here (it goes over 7×10^{-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 all suitable for online application in robotics (less than 1 millisecond per iteration for the considered number of states).

Fig. 5 shows the influence of the dimensionality D on the metrics for the different approaches (see legend in Fig. 4), when considering $K = 4$ states in the model. We see with $\mathcal{M}_1$ and $\mathcal{M}_2$ that the methods perform similarly well in terms of RMS errors.

When the dimensionality is low, the difficulty is to deal with the redundancy in position 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 performance of LWPR is a bit worse, which can be explained by the online nature of the learning process, that cannot determine in advance whether loops in the motion will be encountered, while a batch learning process can cluster these crossings more easily.

For $\mathcal{M}_4$ in Fig. 5, 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 once a local maximum likelihood is reached. Depending on the initialization, a very different number of iterations may be required to reach the 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. 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 quasi linear trend for dimensionalities below $D = 12$).

Fig. 6 evaluates the robustness to external perturbations, for $K = 4$ states and $D = 7$ dimensions. Perlin noise is generated by selecting randomly 4 keypoints along the motion

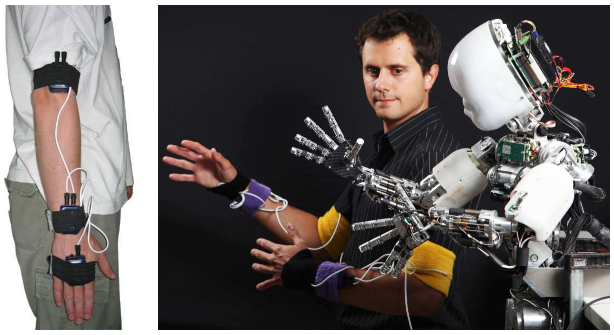

Fig. 7. *Left: X-Sens* motion sensors used to record the user's gesture by collecting joint angle trajectories of the two arms (14 DOFs). *Right:* Demonstration of the skill with simultaneous reproduction on the robot.

and generating a force through a random uniform distribution with standard deviation σ. A continuous force signal is then retrieved by interpolating between the keypoints. We see that HMM, TGMR, LWR and LWPR are robust to perturbations, but that the performance of DMP decreases when σ increases. This difference can however be explained by the fact that DMP uses a fixed value for $\kappa^{\mathcal{P}}$, while the other approaches have been implemented with an adaptive gain as defined in (4). For all approaches, smoothness is nearly not affected by the perturbation for the values of σ considered, principally due to the proportional-derivative controller.

We can conclude from this evaluation that HMM remains competitive with respect to the other approaches considered. The next sections present three robot learning applications that are aimed at demonstrating the strengths of the proposed approach in contexts where the other approaches would not handle the transfer of the skill efficiently.

IV. Experiment with iCub humanoid robot

The aims of this experiment are to show that: (1) the proposed approach can be used to learn periodic motion containing crossings (e.g. such as in a “8” figure); and (2) the algorithm can efficiently handle bimanual movements in joint angle space.

A. Experimental setup

The iCub robot is used in the experiment, which is an open-source humanoid robot resulting from the European project RobotCub [40]. 14 DOFs out of the 53 degrees-of-freedom (DOFs) are used to control the two arms of the robot.

A set of motion sensors are used to record the user's gestures by collecting joint angle trajectories of the upper-body torso, see Fig. 7. 6 *X-Sens* motion sensors are attached to the upper-arms, lower-arms, and at the back of the hands of the user. The data are sent to the robot either by wireless *Bluetooth* communication or by *USB* connection.

Each sensor provides the 3D absolute orientation of each segment by integrating the 3D rate-of-turn, acceleration and earth-magnetic field at a rate of 50 Hz and with a precision of 1.5 degrees. For each joint, a rotation matrix is defined as the orientation of a distal limb segment expressed in the frame of reference of its proximal limb segment. The kinematics motion of each joint is then computed by decomposing the rotation matrix into joint angles, see [22] for details. The upper torso is defined here as a kinematic chain where the *shoulder joint* connects the girdle and the upper arm (3 DOFs), the *elbow* connects the upper arm and the forearm (1 DOF), the *wrist* connects the forearm and the hand (3 DOFs).

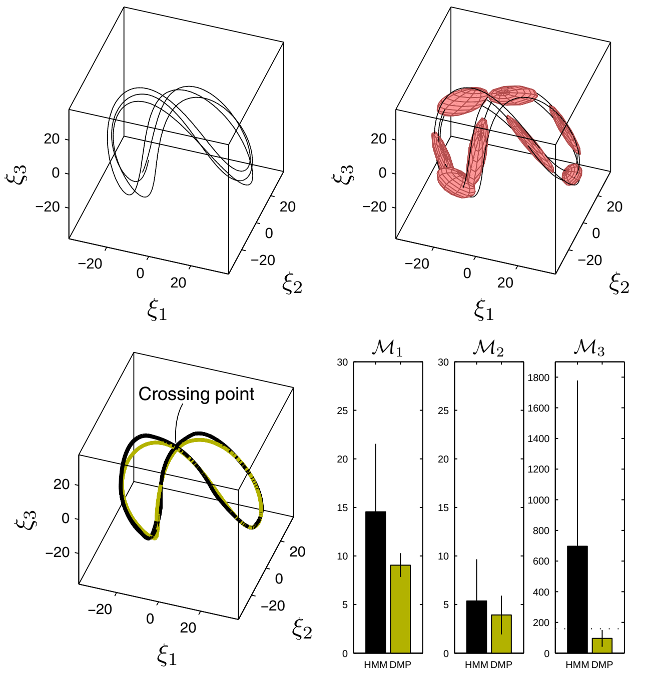

Fig. 8. Demonstration (*top-left*), model (*top-right*), reproductions (*bottom-left*) and evaluation (*bottom-right*) of the dancing motion. For visualization purpose, the 14 DOFs periodic trajectories and associated HMM have been projected into a latent space of 3 dimensions $\{\xi_1, \xi_2, \xi_3\}$ through *Principal Component Analysis* (PCA). The reproductions with the HMM and DMP processes are respectively represented with black and yellow lines. For $\mathcal{M}_3$ in the last graph, the dotted line depicts the RMS jerk value for the training data.

A simple rhythmic movement is demonstrated through the motion sensors and simultaneously reproduced on the iCub. After having observed 3-4 periods of the movement, the robot learns a model of the cyclic motion. The motion is reproduced by the HMM approach presented in Section II, and compared to DMP. For DMP, the version of the algorithm for periodic motion is employed, see Appendix B.[10]

B. Experimental results

Fig. 8 presents the encoding, reproduction and evaluation results. The 14 DOFs motion contains a crossing in joint space, which is also observed in the PCA projection of the data. At a given iteration, the robot must thus move differently depending on the precedent postures along the motion. We see that the high-dimensional periodic movement with crossing is correctly handled by DMP and HMM (8 states have been used in both cases). DMP shows the best score in terms of accuracy and smoothness. The drawback is that the cyclic form must be set beforehand (discrete and periodic signals use a different

[10]For DMP, the period of the movement has been defined explicitly here.

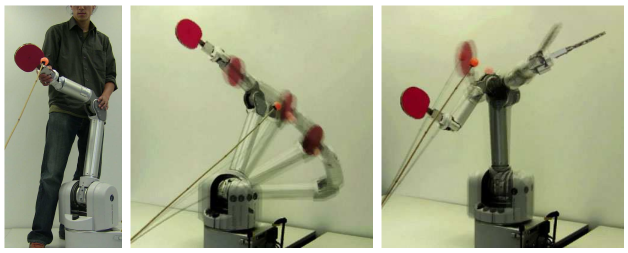

Fig. 9. *Left:* Experimental setup for the experiment of teaching the *Barrett WAM* robotic arm to hit a ball. *Center:* Reproduction of a *drive* stroke. *Right:* Reproduction of a *topspin* stroke.

representation in DMP), and an external method is required to estimate the period of the motion.

In contrast to HMM, LWR and LWPR have problems to correctly handle the crossing point during the movement. From an algorithmic point of view, passing through the same point several times along the motion (or along the cycle in the case of periodic movement) can not be handled by LWR and LWPR. This is confirmed practically by running the algorithms on the dancing dataset. When reaching the crossing point, the two methods provide inadequate motion behaviors. The controller can produce an undesired average of the different motion behaviors learned at this point. The system can also follow indefinitely only a single part of the periodic movement (e.g., by circularly following only the upper part of a "8" figure).

For this reason, LWR and LWPR have not been quantitatively evaluated here. Similarly, TGMR has not been evaluated as it cannot efficiently encode periodic motion due to the explicit encoding of time in the model. A video of the experiment accompanies the submission, and is available online [31].

V. Experiment with WAM robotic arm

This experiment aims at demonstrating that the framework can be used in an unsupervised learning manner. By that we mean that several movements can be encoded in a single HMM, without specifying the number of movements, and without associating the different demonstrations with a class or label.

A. Experimental setup

The experiment consists of learning and reproducing the motion of hitting a ball with a table tennis racket by using a *Barrett WAM* 7 DOFs robotic arm, see Fig. 9 *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 [13], [34] by assuming that the robot must hit the ball with a desired velocity retrieved from the demonstrations. The robot thus hits the ball, continues its motion and stops, which is more natural than 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

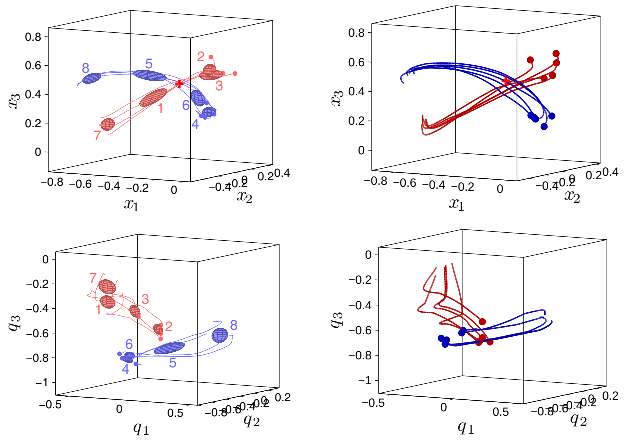

Fig. 10. Encoding and reproduction results of the table tennis experiment (in position space). *Left:* Demonstrated movements and associated Hidden Markov Model, where 8 Gaussians are used to encode the two categories of movements (the learned transitions are represented in Fig. 11). The position of 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 are respectively represented in blue and red for visualization purposes, 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.

is used by players to allow the ball to be hit harder but still land on the table. The stroke with no spin (or with a small amount of topspin) 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 stereoscopic vision 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) [41].

A damped least square inverse kinematics solution with optimization in the null space of the Jacobian matrix is used to reproduce the task, see [23] for details.

B. Experimental results

Figs 9 and 10 present the encoding and reproduction results. We see that the HMM approach reproduces an appropriate motion in the two situations. Fig. 11 *left*, presents the states transitions learned by the HMM. We see that the model has correctly learned that two different dynamics can be

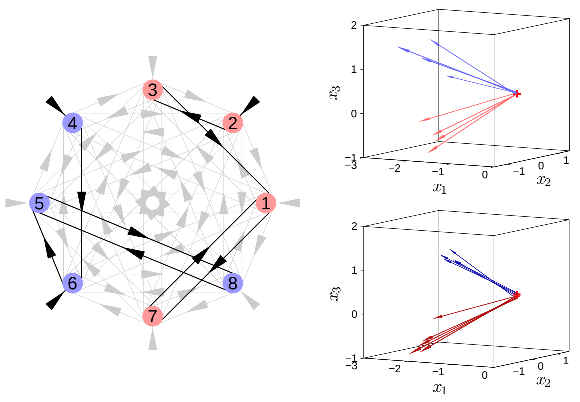

Fig. 11. *Left:* HMM representation of the transitions and initial state probabilities (the corresponding state output distributions are represented in Fig. 10). The states of the HMM are spatially organized around a circle for representation purposes. The possible transitions are depicted inside the circle by arrows, while the probabilities of starting from an initial state are represented outside the circle by arrows. Probabilities above 0.1 are represented by black lines (self transitions probabilities are not represented here). From this representation, two different sequences defined by states transitions 2-3-1-7 and 4-6-5-8 appear, initiated by 2 for the first one, and by 4 or 6 for the second one. *Right:* Position and velocity of the racket at the time of the impact for the 8 demonstrations (*top*) and for the 10 reproduction attempts (*bottom*).

TABLE I
POSITION AND VELOCITY OF THE RACKET AT THE TIME OF THE IMPACT FOR DEMONSTRATIONS AND REPRODUCTIONS.

		Demonstrations		Reproductions	
		position *[m]*	velocity *[m/s]*	position [m]	velocity *[m/s]*
Drive	x	-0.43± 0.01	-2.19± 0.04	-0.41± 0.01	-2.37± 0.12
	y	0.17± 0.01	0.27± 0.30	0.17± 0.06	0.20± 0.27
	z	0.43± 0.01	-1.19± 0.31	0.39± 0.02	-1.24± 0.32
Topspin	x	-0.43± 0.01	-2.12± 0.38	-0.44± 0.01	-1.66± 0.12
	y	0.13± 0.02	-0.03± 0.19	0.12± 0.04	0.17± 0.18
	z	0.44± 0.02	0.73± 0.38	0.43± 0.03	0.73± 0.16

achieved here, depending on the initial position of the robot. It is thus possible to encode several motion alternatives in a single model, without having to provide the number and labels of alternatives during the demonstration phase. The alternatives are then automatically retrieved depending on the initial situation.

Fig. 11 *right*, and Table I present the results of the strokes at the time of the impact with the ball. We see that the system correctly attains the ball at a velocity similar to the one demonstrated (in terms of both amplitude and direction). A video of the experiment accompanies the submission, and is available online [31].

VI. EXPERIMENT WITH HOAP-3 AND ROBOTA

This experiment aims at demonstrating that the framework can be used to learn a controller by taking simultaneously into account several constraints. Here, we consider the case where a set of movements relative to a set of landmarks must be considered for a correct reproduction of the skill (i.e., where several actions on objects are relevant for the task).

A. Experimental setup

In the previous experiment, we learned trajectories in the frame of reference of a single object (the ball). This experiment

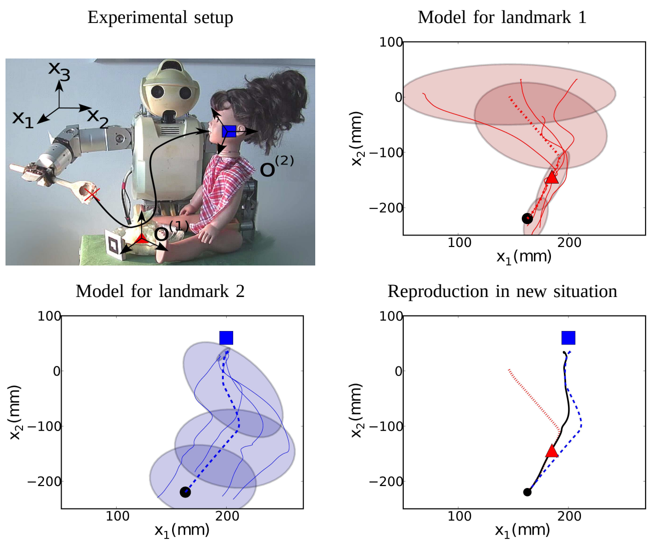

Fig. 12. *Top-left:* Experimental setup to teach the HOAP-3 humanoid robot to feed a *Robota* robotic doll. *Top-right, bottom-left:* Trajectories relative to the two landmarks are encoded in two HMMs of 4 states. Each Gaussian encodes position and velocity information along the task. Generated trajectories using the corresponding models are represented with dashed lines, where the dots show the initial positions. The position of the landmarks are represented with a triangle for the plate and with a square for Robota's mouth. *Bottom-right:* The final reproduction is represented by a solid line. The reproduction shows that the robot tends to satisfy the first constraint first (to reach for the plate) and then switches smoothly to the second constraint (to reach for Robota's mouth).

with a humanoid robot extends this approach by considering trajectories with respect to multiple landmarks. A *HOAP-3* humanoid robot from *Fujitsu* is used in the experiment. It has in total 28 DOFs, of which the 8 DOFs of the upper torso are used in the experiment (4 DOFs per arm). A *kinesthetic teaching* process is used for demonstration. The selected motors are set to passive mode, which allows the user to move freely the corresponding degrees of freedom while the robot executes the task. The kinematics of each joint motion are recorded at a rate of 1 kHz.

The experiment consists of feeding a *Robota* robotic doll [42], where *HOAP-3* first brings a spoon to a plate of mashed potatoes and then moves it towards *Robota*'s mouth, see Fig. 12. Four kinesthetic demonstrations are provided by changing the initial positions of the landmarks from one demonstration to the other. The experimenter explicitly signals the start and the end of the recording to the robot.

The set of landmarks (or objects) tracked by the robot is pre-defined. The position of the plate is recorded through a patch attached to it, which is tracked by an external vision system placed to the side of the robot. The position of *Robota*'s mouth is tracked by proprioception through the robot's motor encoders. *HOAP-3*'s left arm is rigidly attached to *Robota* and *HOAP-3* is connected to *Robota*'s head encoders. *Robota*'s head is thus considered as an additional link to the kinematic model of the robot. This allows to precisely track the position of the mouth during demonstration and reproduction, without the use of a visual marker that would easily be occluded by the spoon moving around the mouth.

In the demonstration phase, the position x of the end-effector is collected in the frame of reference of the robot's torso (fixed frame of reference as the robot is seated during the experiment). This trajectory is expressed in the frames of reference of the different landmarks (moving frames of references) defined for each landmark n by position $o^{(n)}$ and orientation matrix $R^{(n)}$

$$x^{(n)} = \left(R^{(n)}\right)^{\top} \left[x - o^{(n)}\right] , \qquad \dot{x}^{(n)} = \left(R^{(n)}\right)^{\top} \dot{x}.$$

Encoding in a *Hidden Markov Model* is computed for each landmark as described in Section II. During the reproduction phase, for new position $o'^{(n)}$ and orientation $R'^{(n)}$ of the landmarks, the generalized position $\hat{x}$ and velocity $\hat{\dot{x}}$ of the end-effector with respect to the different landmarks is projected back to the frame of reference attached to the torso

$$\hat{x}'^{(n)} = R'^{(n)} \hat{x}^{(n)} + o'^{(n)} , \qquad \hat{\dot{x}}'^{(n)} = R'^{(n)} \hat{\dot{x}}^{(n)}.$$

The associated covariances matrices are transformed through the linear transformation property of Gaussian distributions

$$\begin{aligned} \hat{\Sigma}'^{x(n)} &= R'^{(n)}\, \hat{\Sigma}^{x(n)}\, \left(R'^{(n)}\right)^{\top}, \\ \hat{\Sigma}'^{\dot{x}(n)} &= R'^{(n)}\, \hat{\Sigma}^{\dot{x}(n)}\, \left(R'^{(n)}\right)^{\top}. \end{aligned}$$

At each time step, the command defined in (3) is used to retrieve the desired velocity $\hat{\dot{x}}'$ and desired position $\hat{x}'$, where the resulting distributions $\mathcal{N}(\hat{\dot{x}}', \hat{\Sigma}'^{\dot{x}})$ and $\mathcal{N}(\hat{x}', \hat{\Sigma}'^{x})$ are respectively computed through the Gaussian products $\prod_{n=1}^{N} \mathcal{N}(\hat{\dot{x}}'^{(n)}, \hat{\Sigma}'^{\dot{x}(n)})$ and $\prod_{n=1}^{N} \mathcal{N}(\hat{x}'^{(n)}, \hat{\Sigma}'^{x(n)})$. This allows the system to combine automatically the different constraints associated with the landmarks.

B. Experimental results

Fig. 12 presents the encoding results. The *top-right* graph highlights through the forms of the Gaussian distributions that parts of the motion are more constraints than others. With respect to landmark 1, strong consistency among the demonstrations have been observed at the beginning of the gesture (motion of the spoon in the mashed potatoes), which is reflected by the narrower form of the ellipses at the beginning of the motion.

With respect to landmark 2 (*bottom-left* graph), strong consistency among the demonstrations have been observed at the end of the gesture (when reaching for Robota's mouth). Fig. 12 *bottom-right*, presents the reproduction results. We see that the robot automatically combines the two sets of constraints (associated with the plate and with Robota's mouth) to find a trade-off satisfying probabilistically the constraints observed during the demonstrations. A video of the experiment accompanies the submission, and is available online [31].

VII. Discussion

We presented an evaluation experiment based on randomly generated data and three applications highlighting different capabilities of the model. The aim of the experiment presented in Section III was to conduct a systematic evaluation for various dimensionalities, for models of various complexity and for perturbations of varying amplitudes. It however remains valid only for a specific case, that is, in the context where an acceleration command is recursively evaluated after having observed a set of position and velocity data.

The proposed HMM approach shares many characteristics with the DMP approach, but has some advantages that have been highlighted through the experiments. In DMP, the weights are determined through a decay term, which allows the system to guarantee convergence to the last attractor μ_K^x. In contrast, the HMM method has the disadvantage that its stability lies on proper choice of the gains in (3). An improper choice would directly affect the stability of the system. These gains must be set by estimating in advance the perturbations that are expected during reproduction and/or the range of novel initial positions that the system is expected to handle.

On the other hand, the HMM approach has the advantage of being able to encode several motion alternatives in the same model (see the table tennis experiment in Sec. V). Partial demonstrations can be provided, which is a clear 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 (see the dance learning experiment in Sec. IV). It is also not necessary to specify the frequency of the movement in contrast with DMP that requires to first use an external system to estimate the fundamental frequency of the system [43], [44].

Another drawback of DMP is that a heuristic must be explicitly defined to let the system recompute the value of the canonical variable s in case of a strong perturbation or when one wants to reproduce only a subpart of the motion. Indeed, DMP is robust to spatial perturbation but requires some heuristics to handle temporal perturbations such as delay and pauses in the motion (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, which then does not need the explicit parametrization of a temporal decay. Spatial and temporal distortions are handled very flexibly through the HMM representation.

The proposed HMM approach is not constrained to movements with a unique zero-velocity attractor point. As highlighted in Sec. V, several points of interest to be attained with a desired velocity can be automatically extracted along the motion. 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. The covariance matrices in (1) provide local information on the spread of each center μ_i. This therefore allows building an efficient regression estimate, even if a low number of Gaussians is considered.

We plan in future work to extend the framework to skills requiring more complex dynamics. Of particular interest is the consideration of force signals in the learning by imitation framework. This would allow the transfer of tasks requiring specific compliance to the robot, such as handling manipulation skills collaboratively with a human user. Current work also investigates how the proposed approach can be combined with *Reinforcement Learning* techniques, which would allow the robot to reuse its knowledge for the exploration of new solutions [2].

VIII. Conclusion

We presented and evaluated a probabilistic approach combined with dynamical systems to allow robots to acquire new skills by imitation. The use of HMM allowed us to get rid of the explicit time dependency that was considered in our previous work [12], by still encapsulating precedence information within the statistical representation. For the context of separated learning and reproduction processes, this novel formulation was systematically evaluated with respect to our previous approach, *Locally Weighted Regression* (LWR) [25], *Locally Weighted Projection Regression* (LWPR) [26], and *Dynamic Movement Primitives* (DMP) [13], [35]. We finally presented three applications to highlight the strengths of the proposed approach.

Appendix A
Reformulation as mixture of linear systems

By rewriting $\hat{\dot{x}}$ and $\hat{x}$ in (3) as a mixture of linear systems

$$\begin{array}{l}\hat{\dot{x}} = \sum_{i=1}^{K} h_i(M_i\, x + v_i)\\ \hat{x} = \sum_{i=1}^{K} h_i(M_i'\, \dot{x} + v_i')\end{array} \text{ with } \left| \begin{array}{l} M_i = \Sigma_i^{\dot{x}x}(\Sigma_i^{x})^{-1},\\ M_i' = \Sigma_i^{x\dot{x}}(\Sigma_i^{\dot{x}})^{-1},\\ v_i = \mu_i^{\dot{x}} - \Sigma_i^{\dot{x}x}(\Sigma_i^{x})^{-1}\mu_i^{x},\\ v_i' = \mu_i^{x} - \Sigma_i^{x\dot{x}}(\Sigma_i^{\dot{x}})^{-1}\mu_i^{\dot{x}}, \end{array}\right.$$

and knowing that $\sum_{i=1}^{K} h_i = 1$, (3) can be rewritten as

$$\ddot{x} = \sum_{i=1}^{K} h_i\left(C_i\, \dot{x} + C_i'\, x + C_i''\right) \text{with} \left| \begin{array}{l} C_i = \kappa^{\mathcal{P}} M_i' - \kappa^{\mathcal{V}} I,\\ C_i' = \kappa^{\mathcal{V}} M_i - \kappa^{\mathcal{P}} I,\\ C_i'' = \kappa^{\mathcal{V}} v_i + \kappa^{\mathcal{P}} v_i'. \end{array}\right.$$

The corresponding state-space representation for each subsystem i can then be written as

$$\overbrace{\frac{1}{dt}\begin{bmatrix} x \\ \dot{x} \end{bmatrix}}^{\dot{\mathcal{X}}} = \overbrace{\begin{bmatrix} 0 & I \\ C_i' & C_i \end{bmatrix}}^{A_i} \overbrace{\begin{bmatrix} x \\ \dot{x} \end{bmatrix}}^{\mathcal{X}} + \overbrace{\begin{bmatrix} 0 \\ C_i'' \end{bmatrix}}^{b_i},$$

where I is the identity matrix, and 0 represents a null matrix or vector.

Appendix B
DMP reformulation

By using a formulation similar to the one used in Sec. II (see also the reformulation in [35]), *Dynamic Movement Primitives* is computed as

$$\ddot{x} = (\hat{x} - x)\,\kappa^{\mathcal{P}} - \dot{x}\,\kappa^{\mathcal{V}}, \quad \text{with} \quad \hat{x} = \sum_{i=1}^{K} h_i \mu_i^{x}, \qquad (5)$$

where gains $\kappa^{\mathcal{P}}$ and $\kappa^{\mathcal{V}}$ have been fixed to obtain a critically damped system. The weights h_i are defined by Gaussian distributions

$$h_i(s) = \mathcal{N}(s;\ \mu_i^{s}, \Sigma_i^{s}),$$

and normalized such that $\sum_i^K h_i = 1$. $s \in [0, 1]$ is a decay term initialized with $s = 1$ and converging to zero through a canonical system[11] $\dot{s} = -\alpha s$. Centers μ_i^s are equally distributed between 1 and 0, and variance parameters Σ_i^s are set to a constant value depending on the number of kernels (here, $\alpha = 0.1$, $\kappa^{\mathcal{P}} = (\kappa^{\mathcal{V}})^2/4$ and $\Sigma_i^s = \frac{3}{2\kappa^{\mathcal{V}}}$).

Centers μ_i^x are learned through regression from the observed data.[12] For each datapoint $\{x_i, \dot{x}_i, \ddot{x}_i\}_{i=1}^N$ of the training set ($N = MT$), and following (5), a set of attractors $\hat{x}_i$ are defined as

$$\hat{x}_i = \ddot{x}_i/\kappa^{\mathcal{P}} + \dot{x}_i\kappa^{\mathcal{V}}/\kappa^{\mathcal{P}} + x_i \quad \forall i \in \{1, \ldots, N\}.$$

By rewriting (5) in a matrix form, we define $\hat{X} = H\Phi$ with $\hat{X} = (\hat{x}_1, \ldots, \hat{x}_N)$, $H = (\hat{h}_0, \ldots, \hat{h}_N)$ and $\Phi = (\mu_1^x, \ldots, \mu_K^x)$ ($\hat{X} \in \mathbb{R}^{N\times D}$, $H \in \mathbb{R}^{N\times K}$ and $\Phi \in \mathbb{R}^{K\times D}$). The set of $\hat{h}_i$ is determined for each datapoint by numerically integrating the canonical system $\dot{s} = -\alpha s$. Centers μ_i^x are then estimated through least-square regression

$$\Phi = (H^T H)^{-1} H^T \hat{X},$$

where $(H^T H)^{-1} H^T$ is the pseudoinverse of H.

DMP can be also used to model periodic motion. In this case, the canonical system is defined as $\dot{s} = 2\pi/T$ where T is the period of the motion. The weights of the Gaussian kernels are then defined as

$$h_i(s) = \mathcal{N}\big(s_{[2\pi]};\ \mu_i^s, \Sigma_i^s\big),$$

and normalized such that $\sum_i^K h_i = 1$. $s_{[2\pi]}$ is the value of s modulus 2π. Centers μ_i^s are distributed equally around the circle.

References

[1] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, “Learning from demonstration and adaptation of biped locomotion,” *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 79–91, 2004.

[2] F. Guenter, M. Hersch, S. Calinon, and A. Billard, “Reinforcement learning for imitating constrained reaching movements,” *Advanced Robotics*, vol. 21, no. 13, pp. 1521–1544, 2007.

[3] M. Nicolescu and M. Mataric, “Natural methods for robot task learning: Instructive demonstrations, generalization and practice,” in *Proc. Intl Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 2003, pp. 241–248.

[4] J. Saunders, C. Nehaniv, and K. Dautenhahn, “Teaching robots by moulding behavior and scaffolding the environment,” in *Proc. ACM SIGCHI/SIGART Conf. on Human-Robot Interaction (HRI)*, March 2006, pp. 118–125.

[5] M. Pardowitz, R. Zoellner, S. Knoop, and R. Dillmann, “Incremental learning of tasks from user demonstrations, past experiences and vocal comments,” *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 37, no. 2, pp. 322–332, 2007.

[11]Note that this secondary term creates an implicit time-dependency, whereby each step of motion is incremented following the clock of the canonical system.

[12]Here, learning has been performed in a batch mode by taking the least-square solution of the set of linear equations defined by the training data. Note however that the original approach also allows incremental learning of the centers based on *Locally Weighted Regression* (LWR).

[6] S. Ekvall and D. Kragic, "Learning task models from multiple human demonstrations," in *Proc. IEEE Intl Symposium on Robot and Human Interactive Communication (RO-MAN)*, September 2006, pp. 358–363.

[7] A. Alissandrakis, C. Nehaniv, and K. Dautenhahn, "Correspondence mapping induced state and action metrics for robotic imitation," *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 37, no. 2, pp. 299–307, 2007.

[8] B. Argall, B. Browning, and M. Veloso, "Learning robot motion control with demonstration and advice-operators," in *IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, September 2008, pp. 399–404.

[9] H. Friedrich, S. Muench, R. Dillmann, S. Bocionek, and M. Sassin, "Robot programming by demonstration (RPD): Supporting the induction by human interaction," *Machine Learning*, vol. 23, no. 2, pp. 163–189, 1996.

[10] 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.

[11] 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.

[12] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 37, no. 2, pp. 286–298, 2007.

[13] 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.

[14] T. Inamura, I. Toshima, and Y. Nakamura, "Acquiring motion elements for bidirectional computation of motion recognition and generation," in *Experimental Robotics VIII*, B. Siciliano and P. Dario, Eds. Springer-Verlag, 2003, vol. 5, pp. 372–381.

[15] K. Dixon and P. Khosla, "Trajectory representation using sequenced linear dynamical systems," in *Proc. of the IEEE Intl Conf. on Robotics and Automation (ICRA)*, vol. 4, April–May 2004, pp. 3925–3930.

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

[17] D. Lee and Y. Nakamura, "Mimesis scheme using a monocular vision system on a humanoid robot," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, April 2007, pp. 2162–2168.

[18] 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.

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

[20] O. Jenkins and M. Mataric, "A spatio-temporal extension to isomap nonlinear dimension reduction," in *Proc. Intl Conf. on Machine Learning (ICML)*, 2004, pp. 56–63.

[21] 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.

[22] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *Proc. ACM/IEEE Intl Conf. on Human-Robot Interaction (HRI)*, March 2007, pp. 255–262.

[23] ——, "A probabilistic programming by demonstration framework handling constraints in joint space and task space," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, September 2008, pp. 367–372.

[24] 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)*, May 2009.

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

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

[27] 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)*, September 2008, pp. 380–385.

[28] 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.

[29] 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.

[30] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.

[31] S. Calinon, "Robot programming by demonstration: A probabilistic approach," `http://programming-by-demonstration.org`, April 2009.

[32] 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.

[33] H. Wold, "Estimation of principal components and related models by iterative least squares," in *Multivariate Analysis*, P. Krishnaiaah, Ed. New York, USA: Academic Press, 1966, pp. 391–420.

[34] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control a unifying view," *Progress in Brain Research*, vol. 165, pp. 425–445, 2007.

[35] 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)*, May 2009, pp. 2587–2592.

[36] C. Nehaniv and K. Dautenhahn, "Like me? - measures of correspondence and imitation," *Cybernetics and Systems*, vol. 32, no. 1-2, pp. 11–51, 2001.

[37] N. Hogan, "Adaptive control of mechanical impedance by coactivation of antagonist muscles," *IEEE Trans. on Automatic Control*, vol. 29, no. 8, pp. 681–690, 1984.

[38] K. Perlin, "Real time responsive animation with personality," *IEEE Trans. on Visualization and Computer Graphics*, vol. 1, no. 1, pp. 5–15, 1995.

[39] S. Snibbe, M. Scheeff, and K. Rahardja, "A layered architecture for lifelike robotic motion," in *Proc. of the Intl Conference on Advanced Robotics (ICAR)*, 1999.

[40] N. Tsagarakis, G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. Santos-Victor, A. Ijspeert, M. Carrozza, and D. Caldwell, "iCub: The design and realization of an open humanoid platform for cognitive and neuroscience research," *Advanced Robotics*, vol. 21, no. 10, pp. 1151–1175, 2007.

[41] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, pp. 461–464, 1978.

[42] A. Billard, "Robota: Clever toy and educational tool," *Robotics and Autonomous Systems*, vol. 42, pp. 259–269, 2003.

[43] A. Gams, A. Ijspeert, S. Schaal, and J. Lenarcic, "On-line learning and modulation of periodic movements with nonlinear dynamical systems," *Autonomous Robots*, vol. 27, no. 1, pp. 3–23, 2009.

[44] J. Buchli, L. Righetti, and A. Ijspeert, "Frequency analysis with coupled nonlinear oscillators," *Physica D: Nonlinear Phenomena*, vol. 237, pp. 1705–1718, August 2008.