Dimensionality Reduction for Dynamic Movement Primitives and Application to Bimanual Manipulation of Clothes

Adrià Colomé, *Member, IEEE*, and Carme Torras,*Senior Member, IEEE*

***Abstract*— Dynamic Movement Primitives (DMPs) are nowadays widely used as movement parametrization for learning robot trajectories, because of their linearity in the parameters, rescaling robustness and continuity. However, when learning a movement with DMPs, a very large number of Gaussian approximations needs to be performed. Adding them up for all joints yields too many parameters to be explored when using Reinforcement Learning (RL), thus requiring a prohibitive number of experiments/simulations to converge to a solution with a (locally or globally) optimal reward. In this paper we address the process of simultaneously learning a DMP-characterized robot motion and its underlying joint couplings through linear Dimensionality Reduction (DR), which will provide valuable qualitative information leading to a reduced and intuitive algebraic description of such motion. The results in the experimental section show that not only can we effectively perform DR on DMPs while learning, but we can also obtain better learning curves, as well as additional information about each motion: linear mappings relating joint values and some latent variables.**

I. Introduction

Motion learning by a robot may be implemented in a similar way to how humans learn to move. An initial coarse movement is learned from a demonstration and then rehearsed, performing some local exploration to adapt and possibly improve the motion.

We humans activate in a coordinated manner those muscles that we cannot control individually [1], generating coupled motions of our articulations that gracefully move our skeletons. Such muscle synergies lead to a drastic reduction in the number of degrees of freedom, which allows humans to learn and easily remember a wide variety of motions.

For most current robots, the relation between actuators and joints is more direct than in humans, usually linear, as in Barrett's WAM robot.

Learning robotic skills is a difficult problem that can be addressed in several ways. The most common approach is Learning from Demonstration (LfD), in which the robot is shown an initial way of solving a task, and then tries to reproduce, improve and/or adapt it to variable conditions. The learning of tasks is usually performed in the kinematic domain by learning trajectories [2], [3], but it can also be carried out in the force domain [4], [5], [6].

A training data set is often used in order to fit a relation between an input (experiment conditions) and an output (a good behavior of the robot). This fitting, which can use different regression models such as Gaussian Mixture Models (GMM) [7], is then adapted to the environmental conditions in order to modify the robot's behavior [8]. However, reproducing the demonstrated behavior and adapting it to new situations does not always solve a task optimally, thus Reinforcement Learning (RL) is also being used, where the solution learned from a demonstration improves through exploratory trial-and-error. RL is capable of finding better solutions than the one demonstrated to the robot.

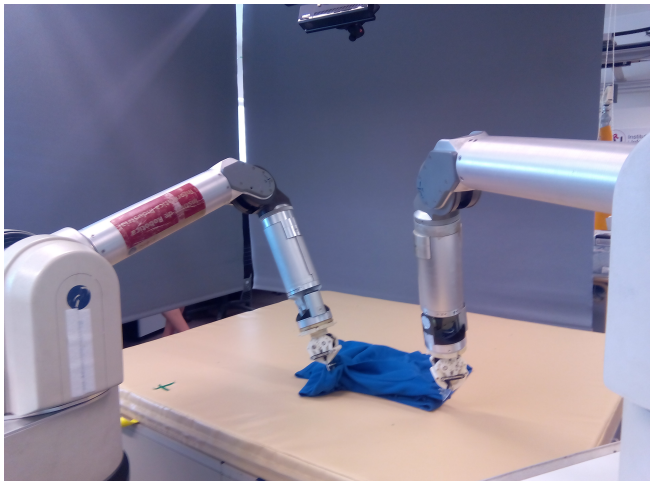


Fig. 1: Two robotic arms coordinating their motions to fold a polo shirt.

These motor/motion behaviors are usually represented with Movement Primitives (MPs), parameterized trajectories for a robot that can be expressed in different ways, such as splines, Gaussian mixtures [9], probability distributions [10] or others. A desired trajectory is represented by fitting certain parameters, which can then be used to improve or change it, while a proper control (a computed torque control [11], for example) tracks this reference signal.

Among all MPs, the most used ones are Dynamic Movement Primitives (DMPs) [12], [13], which characterize a movement or trajectory by means of a second-order dynamical system. The DMP representation of trajectories has good scaling properties wrt. trajectory time and initial/ending positions, has an intuitive behavior, does not have an explicit time dependence and is linear in the parameters, among other advantages [12]. For these reasons, DMPs are being widely

This work was partially developed in the context of the Advanced Grant CLOTHILDE ("CLOTH manIpulation Learning from DEmonstrations"), which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 741930). This work is also partially funded by CSIC projects MANIPlus (201350E102) and TextilRob (201550E028), and Chist-Era Project I-DRESS (PCIN-2015-147).

The authors are with the Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens Artigas 4-6, 08028 Barcelona, Spain. E-mails: [acolome,torras]@iri.upc.edu

used with Policy Search (PS) RL [14], [15], [16], where the problem of finding the best policy (i.e., MP parameters) becomes a case of stochastic optimization. Such PS methods can be gradient-based [16], based on expectation-maximization approaches [15], can also use information-theoretic approaches like Relative Entropy Policy Search (REPS) [17], [18] or be based on optimal control theory, as for the case of Policy Improvement with Path Integrals (PI2) [19], [20], [21]. All these types of PS try to optimize the policy parameters $\boldsymbol{\theta}$, which in our case will include the DMPs' weights, so that an expected reward $\mathbf{J}(\boldsymbol{\theta})$ is maximal, i.e., $\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}}\mathbf{J}(\boldsymbol{\theta})$. After each trajectory reproduction, namely rollout, the reward/cost function is evaluated and, after a certain number of rollouts, used to search for a set of parameters that improves the performance over the initial movement.

These ideas have resulted in algorithms that require several rollouts to find a proper policy update. In addition, to have a good fitting of the initial movement, many parameters are required, while we want to have few in order to reduce the dimensionality of the optimization problem. When applying learning algorithms using DMPs, several aspects must be taken into account:

- **Model availability**. RL can be performed through simulation or with a real robot. The first case is more practical when a good simulator of the robot and its environment is available. However, in the case of manipulation of non-rigid objects or, more generally, when accurate models are not available, reducing the number of parameters and rollouts is critical. Therefore, although model-free approaches like deep reinforcement learning [22] could be applied in this case, they require large resources to successfully learn motion.
- **Exploration constraints**. Certain exploration values might result in dangerous motion of the real robot, such as strong oscillations and abrupt acceleration changes. Moreover, certain tasks may not depend on all the Degrees of Freedom (DoF) of the robot, meaning that the RL algorithm used might be exploring motions that are irrelevant to the task, as we will see later.
- **Parameter dimensionality**. Complex robots still require many parameters for a proper trajectory representation. The number of parameters needed strongly depends on the trajectory length or speed. In a 7-DoF robot following a long 20-second trajectory, the use of more than 20 Gaussian kernels per joint might be necessary, thus having at least 140 parameters in total. A higher number of parameters will usually allow for a better fitting of the initial motion characterization, but performing exploration for learning with such a high dimensional space will result in a slower learning. Therefore, there is a tradeoff between better exploitation (many parameters) and efficient exploration (fewer parameters).

For these reasons, performing Dimensionality Reduction (DR) on the DMPs' DoF is an effective way of dealing with the tradeoff between exploitation and exploration in the parameter space to obtain a compact and descriptive projection matrix which helps the RL algorithm to converge faster to a (possibly) better solution. Additionally, Policy Search approaches in robotics usually have few sample experiments to update their policy. This results in policy updates where there are less samples than parameters, thus providing solutions with exploration covariance matrices that are rank-deficient (note that a covariance matrix obtained by linear combination of samples can't have a higher rank than the number of samples itself). These matrices are usually then regularized by adding a small value to the diagonal so the matrix remains invertible. However, this procedure is a greedy approach, since the unknown subspace of the parameter space is given a residual exploration value. Therefore, performing DR in the parameter space results in the elimination of unexplored space. On the contrary, if such DR is performed in the DoF space, the number of samples is larger than the DoF of the robot and, therefore, the elimination of one degree of freedom of the robot (or a linear combination of them) will not affect such unexplored space, but rather a subspace of the DoF of the robot that has a negligible impact on the outcome of the task.

Other works [23], [24], [25] proposed dimensionality reduction techniques for MP representations. In our previous work [26], we showed how an iterative dimensionality reduction applied to DMPs, using policy reward evaluations to weight such DR could improve the learning of a task, and [27] used weighted maximum likelihood estimations to fit a linear projection model for MPs.

In this paper, our previous work [26] is extended with a better reparametrization after DR, and generalized by segmenting a trajectory using more than a single projection matrix. The more systematic experimentation in three settings shows their clear benefits when used for reinforcement learning. After introducing some preliminaries in Section II, we will present the alternatives to reduce the parameter dimensionality of the DMP characterization in Section III, focusing on the robot's DoF. Then, experimental results with a simulated planar robot, a single 7-DoF WAM robot, and a bimanual task performed by two WAM robots will be discussed in Section IV, followed by conclusions and future work prospects in Section V.

II. Preliminaries

Throughout this work, we will be using DMPs as motion representation and REPS as PS algorithm. For clarity of presentation, we firstly introduce the basic concepts we will be using throughout this work.

A. Dynamic Movement Primitives

In order to encode robot trajectories, DMPs are widely used because of their adaptability. DMPs determine the robot commands in terms of acceleration with the following equation:

$$\begin{gathered}\ddot{\mathbf{y}}/\tau^2 = \alpha_z\left(\beta_z\left(\mathbf{G}-\mathbf{y}\right)-\dot{\mathbf{y}}/\tau\right)+\mathbf{f}(x)\\ \mathbf{f}(x) = \boldsymbol{\Psi}^T\boldsymbol{\omega},\end{gathered} \tag{1}$$

where $\mathbf{y}$ is the joint position vector, $\mathbf{G}$ the goal/ending joint position, τ a time constant, x is a transformation of time verifying $\dot{x} = -\alpha_x x/\tau$. In addition, $\boldsymbol{\omega}$ is the parameter vector of size dN_f, N_f being the number of Gaussian kernels used for each of the d DoF. The parameters $\boldsymbol{\omega}_j$, $j = 1..d$ fitting each joint behavior from an initial move are appended to obtain $\boldsymbol{\omega} = [\boldsymbol{\omega}_1; ...; \boldsymbol{\omega}_d]$, and then multiplied by a Gaussian weights matrix $\boldsymbol{\Psi} = \mathbf{I}_d \otimes \mathbf{g}(x)$, $\otimes$ being the Kronecker product, with the basis functions $\mathbf{g}(x)$ defined as:

$$g_i(x) = \frac{\phi_i(x)}{\sum_j \phi_j(x)} x, i = 1..N_f, \tag{2}$$

where $\phi_i(x) = \exp\left(-0.5(x - c_i)^2/d_i\right)$, and c_i, d_i represent the fixed center and width of the ith Gaussian.

With this motion representation, the robot can be taught a demonstration movement, to obtain the weights $\boldsymbol{\omega}$ of the motion by using least squares or maximum likelihood techniques on each joint j separately, with the values of $\mathbf{f}$ isolated from Eq. (1).

B. Learning an initial motion from demonstration

A robot can be taught an initial motion through kinesthetic teaching. However, in some cases, the robot might need to learn an initial trajectory distribution from a set of demonstrations. In that case, similar trajectories need to be aligned in time. In the case of a single-demonstration, the user has to provide an arbitrary initial covariance matrix $\boldsymbol{\Sigma}_\omega$ for the weights distribution with a magnitude providing as much exploration as possible while keeping robot behavior stable and safe. In the case of several demonstrations, we can sample from the parameter distribution, increasing the covariance values depending on how local we want our policy search to be.

Given a set of taught trajectories $\tau_1, ..., \tau_{N_k}$, we can obtain the DMP weights for each one and fit a normal distribution $\boldsymbol{\omega} \sim \mathcal{N}(\boldsymbol{\mu}_\omega, \boldsymbol{\Sigma}_\omega)$, where $\boldsymbol{\Sigma}_\omega$ encodes the time-dependent variability of the robot trajectory at the acceleration level. To reproduce one of the trajectories from the distribution, the parameter distribution can be sampled, or tracked with a proper controller that matches the joint time-varying variance, as in [28].

This DMP representation of a demonstrated motion will then be used to initialize a RL process, so that after a certain number of reproductions of the trajectory (rollouts), a cost/reward function will be evaluated for each of those trajectories, and a suitable RL algorithm will provide new parameters that represent a similar trajectory, with a higher expected reward.

C. Policy search

Along this work, we will be using Relative Entropy Policy Search (REPS) as PS algorithm. REPS [17], [18] finds the policy that maximizes the expected reward of a robotic task, subject to the constraint of a bound on the Kullback-Leibler Divergence [29] of the new policy with respect to the previous policy $q(\boldsymbol{\omega})$, to avoid large changes in robotic tasks policies which could result in dangerous robot motion. Formally:

$$\begin{gathered} \pi^* = \text{argmax}_\pi \int \pi(\boldsymbol{\omega}) \mathbf{R}(\boldsymbol{\omega}) d\boldsymbol{\omega} \\ \text{s.t. } \epsilon_{KL} \geq \int \pi(\boldsymbol{\omega}) \log \frac{\pi(\boldsymbol{\omega})}{q(\boldsymbol{\omega})} d\boldsymbol{\omega} \\ 1 = \int \pi(\boldsymbol{\omega}) d\boldsymbol{\omega} \end{gathered} \tag{3}$$

where $\boldsymbol{\omega}$ are the parameters of a trajectory, $\mathbf{R}(\boldsymbol{\omega})$ their reward, and $\pi(\boldsymbol{\omega})$ is the probability, according to the policy π, of having such parameters. For DMPs, the policy π will be represented by $\boldsymbol{\mu}_\omega$ and $\boldsymbol{\Sigma}_\omega$, generating sample trajectories $\boldsymbol{\omega}$.

Solving this constrained optimization problem provides a solution of the form

$$\pi^* \propto q(\boldsymbol{\omega}) \exp(\mathbf{R}(\boldsymbol{\omega})/\eta), \tag{4}$$

where η is the solution of a dual function (see [17] for details on this derivation). Having the value of η and the rewards, the exponential part in Eq. (4) acts as a weight to use with the trajectory samples $\boldsymbol{\omega}_k$ in order to obtain the new policy, usually with a Gaussian weighted maximum likelihood estimation.

Table I shows a list of the parameters and variables used throughout this paper; those related to the coordination matrix will be introduced in the following section.

TABLE I: Parameters and variables

$\boldsymbol{\theta} = \{\boldsymbol{\Omega}, \boldsymbol{\mu}_\omega \boldsymbol{\Sigma}_\omega\}$	DMP parameters
$\boldsymbol{\Omega}$	Coordination matrix
$\boldsymbol{\omega} \sim \mathcal{N}(\boldsymbol{\mu}_\omega, \boldsymbol{\Sigma}_\omega)$	DMP weights with mean and covariance
α_z, β_z, α_x, τ	DMP fixed parameters
d, r	Robot's DOF and reduced dimensionality
N_f	Gaussian basis functions used per DoF
k, N_k	Rollout index and number of rollouts per policy update
t, N_t	Time index and number of timesteps
s, N_s	Coordination matrix index and number of coordination matrices
$\mathbf{y}$, $\mathbf{x}$	Robot's joint position vector and end effector's Cartesian pose

III. DMP COORDINATION

In this section, we will describe how to efficiently obtain the joint couplings associated to each task during the learning process, in order to both reduce the dimensionality of a problem, as well as obtaining a linear mapping describing a task. In [30], a coordination framework for DMPs was presented, where a robot's movement primitives were coupled through a coordination matrix, which was learned with an RL algorithm. Kormushev et al. [31] worked in a similar direction, using square matrices to couple d primitives represented as attractor points in the task space domain.

We now propose to use a not necessarily square coordination matrix in order to decrease the number of actuated DoF and thus reduce the number of parameters. To this purpose, in Eq. (1) we can take:

$$\mathbf{f}(x_t) = \boldsymbol{\Omega} \boldsymbol{\Psi}_t^T \boldsymbol{\omega}, \tag{5}$$

for each timestep t, $\boldsymbol{\Omega}$ being a $(d \times r)$ matrix, with $r \leq d$ a reduced dimensionality, $\boldsymbol{\Psi}_t^T = \mathbf{I}_r \otimes \mathbf{g}$, similarly as in the

previous section, and $\boldsymbol{\omega}$ is an (rN_f)-dimensional vector of motion parameters. Note that this representation is equivalent to having r movement primitives encoding the d-dimensional acceleration command vector $\mathbf{f}(x)$. Intuitively, the columns of $\boldsymbol{\Omega}$ represent the couplings between the robot's DoF.

The DR reduction in Eq. (5) is preferable to a DR on the DMP parameters themselves for numerical reasons. If such DR would be performed as $\mathbf{f}(x_t) = \boldsymbol{\Psi}_t^T \hat{\boldsymbol{\Omega}} \boldsymbol{\omega}$, then $\hat{\boldsymbol{\Omega}}$ would be a high-dimensional matrix but, more importantly, the number of rollouts per policy update performed in PS algorithms would determine the maximum dimension of the explored space as a subspace of the parameter space, leaving the rest of such parameter space with zero value or a small regularization value at most. In other words, performing DR in the parameter space requires N_f times more rollouts per update to provide full information than performing such DR in the joint space.

In order to learn the coordination matrix $\boldsymbol{\Omega}$, we need an initial guess and also an algorithm to update it and eliminate unnecessary degrees of freedom from the DMP, according to the reward/cost obtained. Within this representation, we can assume that the probability of having certain excitation values $\mathbf{f}_t = \mathbf{f}(x_t)$ at a timestep given the weights $\boldsymbol{\omega}$ is $p(\mathbf{f}_t|\boldsymbol{\omega}) \sim \mathcal{N}(\boldsymbol{\Omega}\boldsymbol{\Psi}_t^T\boldsymbol{\omega}, \boldsymbol{\Sigma}_\mathrm{f})$, $\boldsymbol{\Sigma}_\mathrm{f}$ being the system noise. Thus, if $\boldsymbol{\omega} \sim \mathcal{N}(\boldsymbol{\mu}_\omega, \boldsymbol{\Sigma}_\omega)$, the probability of $\mathbf{f}_t$ is:

$$p(\mathbf{f}_t) = \mathcal{N}(\boldsymbol{\Omega}\boldsymbol{\Psi}_t^T\boldsymbol{\mu}_\omega, \boldsymbol{\Sigma}_\mathrm{f} + \boldsymbol{\Omega}\boldsymbol{\Psi}_t^T\boldsymbol{\Sigma}_\omega\boldsymbol{\Psi}_t\boldsymbol{\Omega}^T). \quad (6)$$

Along this section we will firstly present the initialization of such coordination matrices in Section III-A, and how they can be updated with a reward-aware procedure in Section III-B. Additionally, Section III-C presents ways of eliminating robot DoF irrelevant for a certain task, and in Section III-E, we present a multiple coordination matrix framework to segment a trajectory so as to use more than one projection matrix. Finally, we consider some numerical issues in Section III-F and a summary in Section III-G.

A. Obtaining an initial coordination matrix with PCA

In this section, we will explain how to obtain the coordination motion matrices while learning a robotic task, and how to update them. A proper initialization for the coordination matrix $\boldsymbol{\Omega}$ is to perform a Principal Component Analysis (PCA) over the demonstrated values of $\mathbf{f}$ (see Eq(1)). Taking the matrix $\overline{\mathbf{F}}$ of all timesteps $\mathbf{f}_t$ in Eq. (5), of size $(d \times N_t)$, for the d degrees of freedom and N_t timesteps as:

$$\overline{\mathbf{F}} = \begin{bmatrix} f_\mathrm{de}^{(1)}(x_0) - \overline{f}_\mathrm{de}^{(1)} & \dots & f_\mathrm{de}^{(1)}(x_{N_t}) - \overline{f}_\mathrm{de}^{(1)} \\ \dots & & \dots \\ f_\mathrm{de}^{(d)}(x_0) - \overline{f}_\mathrm{de}^{(d)} & \dots & f_\mathrm{de}^{(d)}(x_{N_t}) - \overline{f}_\mathrm{de}^{(d)} \end{bmatrix}, \quad (7)$$

$\overline{f}_\mathrm{de}$ being the average over each joint component of the DMP excitation function, for the demonstrated motion (de subindex). Then we can perform Singular Value Decomposition (SVD), obtaining $\overline{\mathbf{F}} = U_{pca} \cdot \Sigma_{pca} \cdot V_{pca}^T$.

Now having set $r < d$ as a fixed value, we can take the r eigenvectors with the highest singular values, which will be the first r columns of $U_{pca} = [\mathbf{u}_1, ..., \mathbf{u}_r, ..., \mathbf{u}_d]$, with associated singular values $\sigma_1 > \sigma_2 > ... > \sigma_d$ and use

$$\boldsymbol{\Omega} = [\mathbf{u}_1, ..., \mathbf{u}_r] \quad (8)$$

as coordination matrix in Eq. (5), having a reduced set of DoF of dimension r, which activate the robot joints (dimension d), minimizing the error in the reprojection $e = \|\overline{\mathbf{F}} - \boldsymbol{\Omega} \cdot \overline{\Sigma} \cdot V_{pca}^T\|_{Frob}^2$, with $\overline{\Sigma}$ the part of Σ_{pca} corresponding to the first r singular values.

Note that this dimensionality reduction does not take any reward/cost function into consideration, so an alternative would be to start with a full-rank coordination matrix and progressively reduce its dimension, according to the costs or rewards of the rollouts. In the next section, we will explain the methodology to update such coordination matrix while also reducing its dimensionality, if necessary.

B. Reward-based Coordination Matrix Update (CMU)

In order to tune the coordination matrix once initialized as described in Section III-A, we assume we have performed N_k reproductions of motion, namely rollouts, obtaining an excitation function $\mathbf{f}_t^{(j),k}$, for each rollout $k = 1..N_k$, timestep $t = 1..N_t$, and DoF $j = 1..d$. Now having evaluated each of the trajectories performed with a cost/reward function, we can also associate a relative weight P_t^k to each rollout and timestep as it is done in policy search algorithms such as PI2 or REPS. We can then obtain a new $d \times N_t$ matrix $\mathbf{F}_\mathrm{co}$ with the excitation function on all timesteps defined as:

$$\mathbf{F}_\mathrm{co}^{new} = \begin{bmatrix} \sum_{k=1}^{N_k} \mathbf{f}_1^{(1),k} P_1^k & \dots & \sum_{k=1}^{N_k} \mathbf{f}_{N_t}^{(1),k} P_{N_t}^k \\ \dots & & \dots \\ \sum_{k=1}^{N_k} \mathbf{f}_1^{(d),k} P_1^k & \dots & \sum_{k=1}^{N_k} \mathbf{f}_{N_t}^{(d),k} P_{N_t}^k \end{bmatrix}, \quad (9)$$

which contains information of the excitation functions, weighted by their relative importance according to the rollout result. A new coordination matrix $\boldsymbol{\Omega}$ can be obtained by means of PCA. However, when changing the coordination matrix, we then need to reevaluate the parameters $\{\boldsymbol{\mu}_\omega, \boldsymbol{\Sigma}_\omega\}$ to make the trajectory representation fit the same trajectory. To this end, given the *old* distribution (represented with a hat) and the one with the new coordination matrix, the excitation functions distributions, excluding the system noise, are

$$\hat{\mathbf{f}}_t \sim \mathcal{N}(\hat{\boldsymbol{\Omega}}\hat{\boldsymbol{\Psi}}_t^T\hat{\boldsymbol{\mu}}_\omega, \hat{\boldsymbol{\Omega}}\hat{\boldsymbol{\Psi}}_t^T\hat{\boldsymbol{\Sigma}}_\omega\hat{\boldsymbol{\Psi}}_t\hat{\boldsymbol{\Omega}}^T) \quad (10)$$

$$\mathbf{f}_t \sim \mathcal{N}(\boldsymbol{\Omega}\boldsymbol{\Psi}_t^T\boldsymbol{\mu}_\omega, \boldsymbol{\Omega}\boldsymbol{\Psi}_t^T\boldsymbol{\Sigma}_\omega\boldsymbol{\Psi}_t\boldsymbol{\Omega}^T). \quad (11)$$

We then represent the trajectories as a single probability distribution over $\mathbf{f}$ using (6):

$$\mathcal{F} = \begin{bmatrix} \mathbf{f}_1 \\ \dots \\ \mathbf{f}_{N_t} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{O}\boldsymbol{\Psi}^T\boldsymbol{\mu}_\omega, \mathbf{O}\boldsymbol{\Psi}^T\boldsymbol{\Sigma}_\omega\boldsymbol{\Psi}\mathbf{O}^T\right), \quad (12)$$

where $\mathbf{O} = \mathbf{I}_{N_t} \otimes \boldsymbol{\Omega}$, and

$$\boldsymbol{\Psi} = \begin{bmatrix} \mathbf{I}_r \otimes \mathbf{g}_1^T \\ \dots \\ \mathbf{I}_r \otimes \mathbf{g}_{N_t}^T \end{bmatrix}, \quad (13)$$

Algorithm 1 Coordination Matrix Update (CMU)

Input:
Rollout and timestep probabilities P_t^k, $k = 1..N_k$, $t = 1..N_t$.
Excitation function $\mathbf{f}_i^{(j),k}$, $j = 1..d$.
Previous update (or initial) excitation function $\mathbf{F}_{\text{co}}$.
Current $\mathbf{\Omega}$ of dimension $d \times r$.
DoF discarding threshold η.
Current DMP parameters $\boldsymbol{\theta} = \{\mathbf{\Omega}, \boldsymbol{\mu}_\omega, \mathbf{\Sigma}_\omega\}$.

1: Compute $\mathbf{F}_{\text{co}}^{\text{new}}$ as in Eq. (9)
2: Filter excitation matrix: $\mathbf{F}_{\text{co}}^{\text{new}} = \alpha \mathbf{F}_{\text{co}}^{\text{new}} + (1-\alpha)\mathbf{F}_{\text{co}}$
3: Subtract average as in Eq. (7)
4: Perform PCA and obtain $U_{pca} = [\mathbf{u}_1, ..., \mathbf{u}_r, ..., \mathbf{u}_d]$ (as detailed in Section III-A)
5: **if** $\sigma_1/\sigma_r > \eta$ **then**
6: $\quad r = r - 1$
7: **end if**
8: $\mathbf{\Omega}^{new} = [\mathbf{u}_1, ..., \mathbf{u}_r]$
9: Recompute: $\{\boldsymbol{\mu}_\omega, \mathbf{\Sigma}_\omega\}$ as in Eqs. (16)-(17)

while

$$\hat{\mathcal{F}} = \begin{bmatrix} \hat{\mathbf{f}}_1 \\ ... \\ \hat{\mathbf{f}}_{N_t} \end{bmatrix} \sim \mathcal{N}\left(\hat{\mathbf{O}}\hat{\mathbf{\Psi}}^T\hat{\boldsymbol{\mu}}_\omega, \hat{\mathbf{O}}\hat{\mathbf{\Psi}}^T\hat{\mathbf{\Sigma}}_\omega\hat{\mathbf{\Psi}}\hat{\mathbf{O}}^T\right), \quad (14)$$

where $\hat{\mathbf{O}} = \mathbf{I}_{N_t} \otimes \hat{\mathbf{\Omega}}$, and $\hat{\mathbf{\Psi}}$ is built in accordance to the value of r in case the dimension has changed, as it will be seen later.

To minimize the loss of information when updating the distribution parameters $\boldsymbol{\mu}_\omega$ and $\mathbf{\Sigma}_\omega$, given a new coordination matrix, we can minimize the Kullbach-Leibler (KL) divergence between $\hat{p} \sim \mathcal{N}(\hat{\boldsymbol{\mu}_\omega}, \hat{\mathbf{\Sigma}_\omega})$ and $p \sim \mathcal{N}(\mathbf{M}\boldsymbol{\mu}_\omega, \mathbf{M}\mathbf{\Sigma}_\omega\mathbf{M}^T)$, being $\mathbf{M} = (\hat{\mathbf{O}}\hat{\mathbf{\Psi}}^T)^\dagger\mathbf{O}\mathbf{\Psi}^T$, and $\dagger$ representing the Moore-Penrose pseudoinverse operator. This reformulation is done so that we have two probability distributions with the same dimensions, and taking into account that the KL divergence is not symmetric, using $\mathbf{f}_t$ as an approximation of $\hat{\mathbf{f}}_t$.

As the KL divergence for two normal distributions is known [32], we have

$$\begin{aligned}\text{KL}(\hat{p}\|p) = \log\frac{|\mathbf{M}\mathbf{\Sigma}_\omega\mathbf{M}^T|}{|\hat{\mathbf{\Sigma}}_\omega|} + \text{tr}\left((\mathbf{M}\mathbf{\Sigma}_\omega\mathbf{M}^T)^{-1}\hat{\mathbf{\Sigma}}_\omega\right) \\ +(\mathbf{M}\boldsymbol{\mu}_\omega - \hat{\boldsymbol{\mu}}_\omega)^T(\mathbf{M}\mathbf{\Sigma}_\omega\mathbf{M}^T)^{-1}(\mathbf{M}\boldsymbol{\mu}_\omega - \hat{\boldsymbol{\mu}}_\omega) - d\end{aligned} \quad (15)$$

Now, differentiating wrt. $\boldsymbol{\mu}_\omega$ and wrt. $(\mathbf{M}\mathbf{\Sigma}_\omega\mathbf{M}^T)^{-1}$, and setting the derivative to zero to obtain the minimum, we obtain:

$$\boldsymbol{\mu}_\omega = \mathbf{M}^\dagger\hat{\boldsymbol{\mu}}_\omega \quad (16)$$

$$\mathbf{\Sigma}_\omega = \mathbf{M}^\dagger\left[\hat{\mathbf{\Sigma}}_\omega + (\mathbf{M}\boldsymbol{\mu}_\omega - \hat{\boldsymbol{\mu}}_\omega)(\mathbf{M}\boldsymbol{\mu}_\omega - \hat{\boldsymbol{\mu}}_\omega)^T\right](\mathbf{M}^T)^\dagger. \quad (17)$$

Minimizing the KL divergence provides the solution with the least loss of information, in terms of probability distribution on the excitation function.

C. Eliminating irrelevant degrees of freedom

In RL, the task the robot tries to learn does not always necessarily depend on all the degrees of freedom of the robot. For example, if we want to track a Cartesian xyz position with a 7-DoF robot, it is likely that some degrees of freedom, which mainly alter the end-effector's orientation, may not affect the outcome of the task. However, these DoF are still considered all through the learning process, causing unnecessary motions which may slow down the learning process or generate a final solution in which a part of the motion was not necessary.

For this reason, the authors claim that the main use of a coordination matrix should be to remove those unnecessary degrees of freedom, and the coordination matrix, as built in Section III-B, can easily provide such result. Given a threshold η for the ratio of the maximum and minimum singular values of $\mathbf{F}_{\text{co}}^{\text{new}}$ defined in Eq.(9), we can discard the last column of the coordination matrix if those singular values verify $\sigma_1/\sigma_r > \eta$.

In Algorithm 1, we show the process of updating and reducing the coordination matrix, where the parameter α is a filtering term, in order to keep information from previous updates.

D. Dimensionality reduction in the parameter space (pDR-DMP)

While most approaches found in literature perform DR in the joint space [27], [25], [26], for comparison purposes we also derived DR in the parameter space. To do so, the procedure is equivalent to that of the previous subsections, with the exception that now the parameter r disappears and we introduce the parameter $M_f \leq dN_f$, indicating the total number of Gaussian parameters used. Then, (5) becomes:

$$\mathbf{f}(x_t) = \mathbf{\Psi}_t^T\mathbf{\Omega}\boldsymbol{\omega}, \quad (18)$$

with $\mathbf{\Psi}_t^T$ being a $(d \times dN_f)$ matrix of Gaussian basis functions, as detailed in Section II and $\mathbf{\Omega}$ being a $(dN_f \times M_f)$ matrix with the mappings from a parameter space of dimension M_f to the whole DMP parameter space. The DMP weight vector $\boldsymbol{\omega}$ now has dimension M_f. In order to initialize the projection matrix $\mathbf{\Omega}$, we have to take the data matrix in Eq.(7), $\overline{\mathbf{F}}$, knowing that:

$$\overline{\mathbf{F}} = [\mathbf{f}_1, ..., \mathbf{f}_{N_t}] = [\mathbf{\Psi}_1^T\mathbf{\Omega}\boldsymbol{\omega}, ..., \mathbf{\Psi}_{N_t}^T\mathbf{\Omega}\boldsymbol{\omega}], \quad (19)$$

which can be expressed as a least-squares minimization problem as:

$$[\mathbf{\Psi}_1^{\dagger,T}\mathbf{f}_1, ..., \mathbf{\Psi}_{N_t}^{\dagger,T}\mathbf{f}_{N_t}] \simeq \mathbf{\Omega}\boldsymbol{\omega}, \quad (20)$$

where $\dagger$ represents the pseudoinverse matrix. We can then obtain the projection matrix $\mathbf{\Omega}$ by performing PCA in Eq. (20). Note that, in order to fit the matrix $\mathbf{\Omega}$ $(dN_f \times M_f)$, we need at least a number of timesteps $N_t > M_f$.

E. Multiple Coordination Matrix Update (MCMU)

Using a coordination matrix to translate the robot degrees of freedom into others more relevant to task performance may result in a too strong linearization. For this reason, multiple coordination matrices can be built in order to perform a coordination framework that uses different mappings throughout the trajectory. In order to do so, we will use a second layer of N_s Gaussians and build a coordination matrix $\mathbf{\Omega}_s$ for each Gaussian $s = 1..N_s$, so that at each timestep the coordination matrix $\mathbf{\Omega}^t$ will be an interpolation between such constant coordination matrices $\mathbf{\Omega}_s$. To compute such an approximation, linear interpolation of projection matrices does not necessarily yield robust result. For that reason, given the time t and the constant matrices $\mathbf{\Omega}_s$, we compute

$$\mathbf{\Omega}^t = \text{argmin}_{\mathbf{X}} \sum_{s=1}^{N_s} \varphi_s^t \left[\text{tr}(\mathbf{\Omega}_s^\dagger \mathbf{X}) - d \log \left(\frac{\|\mathbf{X}\|_F}{\|\mathbf{\Omega}_s\|_F} \right) \right] \tag{21}$$

with

$$\varphi_s^t = \varphi_s(x_t) = \frac{\phi_s(x_t)}{\sum_{p=1}^{N_s} \phi_p(x_t)}, \tag{22}$$

where ϕ_s, $s = 1..N_s$ are equally distributed Gaussians in the time domain, and $\|.\|_F$ is the Frobenius norm. A new Gaussian basis function set is used in order to independently choose the number of coordination matrices, as the number of Gaussian kernels for DMPs is usually much larger than the number needed for linearizing the trajectory in the robot's DoF space. Such number N_s can then be freely set, according to the needs and variability of the trajectory. The optimization cost is chosen for its similarity with the covariance terms of the Kullback-Leibler divergence, and if we use the number of DoF of the robot, d, as a factor in the equation and the matrices $\mathbf{\Omega}_s$ are all orthonormal, then the optimal solution is a linear combination of such matrices:

$$\mathbf{\Omega}^t = \sum_{s=1}^{N_s} \varphi_s^t \mathbf{\Omega}_s. \tag{23}$$

Note that φ_s^t acts as an activator for the different matrices, but it is also used to distribute the responsibility for the acceleration commands to different coordination matrices in the newly-computed matrix $\mathbf{F}_{co}^s$. Then we can proceed as in the previous section, with the exception that we will compute each $\mathbf{\Omega}_s$ independently by using the following data for fitting:

$$\mathbf{F}_{co}^s = \begin{bmatrix} \sum_{k=1}^{N_k} \mathbf{f}_1^{(1),k} \varphi_s^1 P_1^k & \dots & \sum_{k=1}^{N_k} \mathbf{f}_{N_t}^{(1),k} \varphi_1^s P_{N_t}^k \\ \dots & & \dots \\ \sum_{k=1}^{N_k} \mathbf{f}_1^{(d),k} \varphi_s^{N_t} P_1^k & \dots & \sum_{k=1}^{N_k} \mathbf{f}_{N_t}^{(d),k} \varphi_s^{N_t} P_{N_t}^k \end{bmatrix}, \tag{24}$$

and use the following excitation function in Eq. (1):

$$\mathbf{f}(x) = \mathbf{\Omega}_t \mathbf{\Psi}_t^T \boldsymbol{\mu}_\omega, = \left(\sum_{s=1}^{N_s} \varphi_s^t \mathbf{\Omega}_s \right) \mathbf{\Psi}_t^T \boldsymbol{\mu}_\omega. \tag{25}$$

Note that, in Eq. (24), $\mathbf{F}_{co}^s$ will probably have columns entirely filled with zeros. We filtered those columns out before performing PCA, while an alternative is to use φ as weights in a weighted PCA. Both approaches have been implemented and show a similar behavior in real-case scenarios. Now, changing the linear relation between the d robot's DoF and the r variables encoding them within a probability distribution (see Eq. (6)) requires to update the covariance matrix in order to keep it consistent with the different coordination matrices. In this case, as $\mathbf{\Omega}$ is varying, we can reproject the weights similarly as in Eqs. (10)-(17), by using:

$$\boldsymbol{O} = \begin{bmatrix} \mathbf{\Omega}^1 & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & \mathbf{\Omega}^{N_t} \end{bmatrix}, \tag{26}$$

for the new values of r, $\mathbf{\Omega}_s$, $\forall s$, compared to the previous values (now denoted with a hat). We can then use Eqs. (16) and (17) to recalculate $\boldsymbol{\mu}_\omega$ and $\mathbf{\Sigma}_\omega$.

F. Numerical issues of a sum of two coordination matrices

1) Orthogonality of components and locality: When using Eq. (23) to define the coordination matrix, we are in fact doing a weighted sum of different coordination matrices $\mathbf{\Omega}_s$, obtaining a matrix whose *j*-th column is the weighted sum of the *j*-ths columns of the N_s coordination matrices. This operation would not necessarily provide a matrix with its columns pairwise orthonormal, despite all the $\mathbf{\Omega}_s$ having that property. Nevertheless, such orthonormality property is not necessary, other than to have an easier-to-compute inverse matrix. The smaller the differences between consecutive coupling matrices, the closer to an orthonormal column-wise matrix we will obtain at each timestep. From this fact, we conclude that the number of coupling matrices has to be fitted to the implicit variability of the task, so as to keep consecutive coordination matrices relatively similar.

2) Eigenvector matching and sign: Another issue that may arise is that, when computing the singular value decomposition, some algorithms provide ambiguous representations in terms of the signs of the columns of the matrix U_{pca} in Section III-A. This means that it can be the case of two coordination matrices, $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$, having similar columns with opposite signs, the resulting vector being a weighted difference between them, which will then translate into a computed coupling matrix $\mathbf{\Omega}^t$ obtained through Eq. (23) with a column vector that only represents *noise*, instead of a joint coupling.

It can also happen that consecutive coordination matrices $\mathbf{\Omega}_1$, $\mathbf{\Omega}_2$ have similar column vectors but, due to similar eigenvalues coming from the singular value decomposition, their column order becomes different.

Because of these two facts, a reordering of the coupling matrices $\mathbf{\Omega}_s$ has to be carried out, as shown in Algorithm 2. In such algorithm, we use the first coordination matrix $\mathbf{\Omega}_1$ as a reference and, for each other $s = 2..N_s$, we compute the pairwise column dot product of the reference $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_s$. We

Algorithm 2 Reordering of PCA results

Input:

$\mathbf{\Omega}_s$,$\forall s = 1..N_s$, computed with PCA

1: **for** $is = 2..N_s$ **do**
2: Initialize $\mathbf{K} = \mathbf{0}_{r\times r}$, the pair-wise dot product matrix
3: Initialize PCAROT $= \mathbf{0}_{r\times r}$, the rotation matrix
4: **for** $i1 = 1..r$, $i2 = 1..r$ **do**
5: $\mathbf{K}(i1, i2) = \text{dot}(\mathbf{\Omega}_1(:, i1), \mathbf{\Omega}_s(:, i2))$
6: **end for**
7: **for** $j = 1..r$ **do**
8: $v_{max} = \max(|\mathbf{K}(:, j)|)$
9: $i_{max} = \text{argmax}(|\mathbf{K}(:, j)|)$
10: **if** $v_{max} = \max(|\mathbf{K}(i_{max}, :)|)$ **then**
11: PCAROT$(i_{max}, j) = \text{sign}(\mathbf{K}(i_{max}, j))$
12: **end if**
13: **end for**
14: **if** rank(PCAROT) $< r$ **then**
15: Return to line 7
16: **end if**
17: **end for**

then reorder the eigenvectors in $\mathbf{\Omega}_s$ and change their signs according to the dot products matrices.

G. Variants of the DR-DMP method

To sum up the proposed dimensionality reduction methods for DMPs (DR-DMP) described in this section, we list their names and initializations in Tables II and III, which show their descriptions and usages.

TABLE II: Methods description

DR-DMP$^0(r)$	Fixed $\mathbf{\Omega}$ of dimension $(d \times r)$
DR-DMP$^0(N_s, r)$	Fixed multiple $\mathbf{\Omega}_s$ of dimension $(d \times r)$
DR-DMP$_{\text{CMU}}(r)$	Recalculated $\mathbf{\Omega}$ of dimension $(d \times r)$
DR-DMP$_{\text{MCMU}}(N_s, r)$	Recalculated multiple $\mathbf{\Omega}_s$ of dimension $(d \times r)$
IDR-DMP$_{\text{CMU}}$	Iterative DR while recalculating $\mathbf{\Omega}$
IDR-DMP$_{\text{MCMU}}(N_s)$	Iterative DR while recalculating multiple $\mathbf{\Omega}_s$
EM DR-DMP(r)	DR with Expectation Maximization as in [27]
pDR-DMP(M_f)	DR in the parameter space as in Sec. III-D

TABLE III: Methods initialization and usage

Method	Initialization of $\mathbf{\Omega}$	$\boldsymbol{\theta}$ update
DR-DMP$^0(r)$	PCA(r)	REPS
DR-DMP$^0(N_s, r)$	N_s-PCA(r)	REPS
DR-DMP$_{CMU}(r)$	PCA(r)	REPS+CMU, $\eta = \infty$
DR-DMP$_{\text{MCMU}}(N_s, r)$	N_s-PCA(r)	REPS+MCMU, $\eta = \infty$
IDR-DMP$_{CMU}(r)$	PCA(d)	REPS+CMU, $\eta < \infty$
IDR-DMP$_{\text{MCMU}}(N_s, r)$	N_s-PCA(d)	REPS+MCMU, $\eta < \infty$
EM DR-DMP(r)	EM(r)	REPS
pDR-DMP(M_f)	(param) PCA(M_f)	REPS+CMU, $\eta = \infty$

In Table III, PCA(r) represents Principal Component Analysis (PCA) keeping the r eigenvectors with the largest singular values (see. Section III-A). N_s-PCA(r) is used to represent the computation of N_s PCA approximations and coordination using equally-initialized weights in Eq. (24). The CMU algorithm is defined in Algorithm 1, and its MCMU variant in Section III-E. EM DR-DMP(r) represents the adaptation of the work in [27] to the DMPs. Finally, IDR-DMP is used to denote the iterative dimensionality reduction as described in Section III-C, either with one coordination matrix, IDR-DMP$_{CMU}(r)$, or N_s coordination matrices, IDR-DMP$_{\text{MCMU}}(N_s, r)$.

IV. Experimentation

To assess the performance of the different algorithms presented throughout this work, we performed three experiments. An initial one consisting of a fully-simulated 10-DoF planar robot (Experiment 1 in Section IV-A), a simulated experiment with 7-DoF real robot data initialization (Experiment 2 in Section IV-B) and a real-robot experiment with two coordinated 7-DoF robots (Experiment 3 in Section IV-C). We set a different reward function for each task, according to the nature of the problem and based on similar examples in literature [14]. Different variants of the proposed latent space DMP representation have been tested, as well as an EM-based approach [27] adapted to the DMPs. We used episodic REPS in all the experiments and, therefore, time-step learning methods like [25] were not included in the experimentation. The application of the proposed methods does not depend on the REPS algorithm, as they can be implemented with any PS procedure using Gaussian weighted maximum likelihood estimation for reevaluating the policy parameters, such as for example PI2 [19], [20], [21].

A. 10-DoF planar robot arm experiment

As an initial learning problem for testing, we take the planar arm task used as a benchmark in [20], where a d-dimensional planar arm robot learns to adapt an initial trajectory to go through some via-points.

1) Initialization and reward function: Taking $d = 10$, we generated a minimum jerk trajectory from an initial position to a goal position. As a cost function, we used the Cartesian positioning error on two via-points. The initial motion was a min-jerk-trajectory for each of the 10 joints of the planar arm robot, with each link of length $1m$, from an initial position $q_j(t = 0) = 0\ \forall j$, to the position $q_j(t = 1) = 2\pi/d$ (see Fig. 2(a)). Then, to initialize the trajectory variability, we generated several trajectories for each joint by adding

$$q_j(t) = q_{j,minjerk}(t) + \sum_{a=1}^{2} A_a \text{exp}\left(-(t - c_a)^2/d_a^2\right),$$

where $A_a \sim \mathcal{N}(0, \frac{1}{4d})$, and obtained trajectories from a distribution as those shown for one joint in Fig. 2(b). We used those trajectories to initialize $\boldsymbol{\mu}_\omega$ and $\mathbf{\Sigma}_\omega$.

The task to learn is to modify the trajectory so as to go through $N_v = 2$ via points along the trajectory. As a reward function for the experiments, we used $R = \sum_t r_t$, where r_t is the reward at time-step t defined as:

$$r_t = -\sum_{v=1}^{N_v} \delta(t = t_v)(\mathbf{x}_t - \mathbf{x}_v)^T \mathbf{C}_x (\mathbf{x}_t - \mathbf{x}_v) - \ddot{\mathbf{x}}_t^T \mathbf{C}_u \ddot{\mathbf{x}}_t, \quad (27)$$

which is a weighted sum of an acceleration command and a via-points error; $\mathbf{x}_t, \mathbf{x}_v$ being the Cartesian trajectory point

and via-point coordinates for each of the $1..N_v$ via-points. This cost function penalizes accelerations in the first joints, which move the whole robot. As algorithmic parameters, we used a bound on the KL-divergence of 0.5 for REPS, and a threshold $\eta = 50$ for the ratio of the maximum and minimum singular values for dimensionality reduction in Algorithm 1.

We used REPS for the learning experiment for a fixed dimension (initially set to a value $r = 1..10$), and starting with $r = 10$ and letting the algorithm reduce the dimensionality by itself. We also allowed for an exploration outside the linear subspace represented by the coordination matrix (noise added to the d-dimensional acceleration commands in simulation) following $\epsilon_{noise} \sim \mathcal{N}(0, 0.1)$.

2) Results and discussion: After running the simulations, we obtained the results detailed in Table IV, where the mean and its 95% confidence interval variability are shown (through 20 runs for each case). An example of solution found can be seen in Fig. 2(c), where the initial trajectory has been adapted so as to go through the marked via-points. The learning curves for those DR-DMP variants considered of most interest in Table IV are also shown in Fig. 2(d).

In Table IV we can observe that:

- Using two coordination matrices yields better results than using one; except for the case of a fixed dimension set to 10, where the coupling matrices would not make sense as they would have full rank.
- Among all the fixed dimensions, the one showing the best results is $r = 2$, which is indeed the dimension of the implicit task in the Cartesian space.
- The variable-dimension iterative method produces the best results.

B. 7-DoF WAM robot circle-drawing experiment

As a second experiment, we kinesthetically taught a real robot - a 7-DoF Barrett's WAM robot - to perform a 3-D circle motion in space.

1) Initialization and reward function: We stored the real robot data obtained through such kinesthetic teaching and a plot of the end-effector's trajectory together with the closest circle can be seen in Fig. 3(a).

Using REPS again as PS algorithm with $\epsilon_{\text{KL}} = 0.5$, 10 simulated experiments of 200 policy updates consisting of 20 rollouts each were performed, reusing the data of up to the previous 4 epochs. Using the same REPS parameters, we ran the learning experiment with $N_f = 15$ Gaussian basis per DoF, and one constant coordination matrix $\mathbf{\Omega}$ and different dimensions $r = 1..7$, namely DR-DMP$^0(r)$. We also ran the experiment updating the coordination matrix of constant dimension after each epoch, DR-DMP$_{CMU}(r)$. Similarly, we ran the learning experiments with $N_s = 3$ coordination matrices: With constant coordination matrices initialized at the first iteration, DR-DMP$^0_{MCMU}(3, r)$, and updating the coordination matrices at every policy update, DR-DMP$_{MCMU}(3, r)$. We also ran the iterative dimensionality reduction with $N_s = 1$ coordination matrix, IDR-DMP$_{CMU}$, and with $N_s = 3$ matrices, IDR-DMP$_{MCMU}(3)$. We then implemented and tested a weighted expectation-maximization approach for linear DR with an expression equivalent to that found in [27], where DR was applied to the forcing term $\mathbf{f}$, with one coordination matrix and a fixed number for the reduced dimension r, EM DR-DMP(r). Last, we added to the comparison the pDR-DMP(M_f) variant described in Section III-D, using $M_f = 15 \cdot 1, ..., 6$, an equivalent number of Gaussian kernels as for the DR-DMP(r) method with $r = 1, ..., 6$.

As a reward function, we used:

$$R = -\left(\sum_{t=1}^{N_t} r^t_{\text{circle}} + \alpha \|\ddot{\mathbf{q}}\|^2\right), \tag{28}$$

where r_{circle} is the minimum distance between the circle and each trajectory point, $\|\ddot{\mathbf{q}}\|^2$ is the squared norm of the acceleration at each trajectory point, and $\alpha = \frac{1}{5 \cdot 10^6}$ is a constant so as to keep the relative weight of both terms in a way the acceleration represents a value between 10% and 20% of the cost function.

2) Results and discussion: The results shown in Table V have the mean values throughout the 10 experiments, and their confidence intervals with 95% confidence. Figure 3(b) shows the learning curves for some selected methods. Using the standard DMP representation as the benchmark for comparison, with $r = 7$ as fixed dimension (see first row in Table V), we can say that:

- Using $N_s = 3$ coordination matrices yields significantly better results than using only one. $N_s = 3$ with a single dimension results in a final reward of -0.010 ± 0.008, the best obtained throughout all experiments.
- It is indeed better to use a coordination matrix update with automatic dimensionality reduction than to use the standard DMP representation. Additionally, it provides information on the true underlying dimensionality of the task itself. In the considered case, there is a significant improvement from $r = 4$ to $r = 3$, given that the reward doesn't take orientation into account and, therefore, the task itself lies in the $3-$dimensional Cartesian space. Moreover, the results indicate that a 1-dimensional representation can be enough for the task.
- Fixing the dimension to 1 leads to the best performance results overall, clearly showing that smaller parameter dimensionality yields better learning curves. It is to be expected that, given a 1-dimensional manifold of the Cartesian space, i.e., a trajectory, there exists a 1-dimensional representation of such trajectory. Our approach seems to be approaching such representation, as seen in black in Fig. 3(a).
- Both DR-DMP$_{CMU}(r)$ and DR-DMP$_{MCMU}(3, r)$ provide a significant improvement over the standard DMP representation, DR-DMP$^0(r)$. This is specially noticeable for $r \leq 3$, where the final reward values are much better. Additionally, the convergence speed is also significantly faster for such dimensions, as the 10 updates column shows.
- EM DR-DMP(r) shows a very fast convergence to high-

TABLE IV: Results for the 10-DoF planar arm experiment displaying $(-\log_{10}(-R))$, R being the reward. DR-DMP variants for several reduced dimensions after the indicated number of updates, using 1 or 2 coordination matrices, were tested.

Dimension	1 update	10 updates	25 updates	50 updates	100 updates	200 updates
DR-DMP$_{CMU}(10)$	0.698 ± 0.070	1.244 ± 0.101	1.713 ± 0.102	1.897 ± 0.038	1.934 ± 0.030	1.949 ± 0.026
DR-DMP$_{CMU}(8)$	0.724 ± 0.073	1.265 ± 0.155	1.617 ± 0.135	1.849 ± 0.079	1.905 ± 0.072	1.923 ± 0.075
DR-DMP$_{CMU}(5)$	$\mathbf{0.752 \pm 0.117}$	$\mathbf{1.304 \pm 0.098}$	1.730 ± 0.108	1.910 ± 0.076	1.954 ± 0.063	1.968 ± 0.064
DR-DMP$_{CMU}(2)$	0.677 ± 0.063	1.211 ± 0.093	$\mathbf{1.786 \pm 0.073}$	$\mathbf{1.977 \pm 0.040}$	$\mathbf{1.993 \pm 0.039}$	$\mathbf{1.997 \pm 0.037}$
DR-DMP$_{CMU}(1)$	0.612 ± 0.057	1.161 ± 0.103	1.586 ± 0.071	1.860 ± 0.056	1.931 ± 0.041	1.951 ± 0.042
DR-DMP$_{MCMU}(2,10)$	$\mathbf{0.738 \pm 0.094}$	$\mathbf{1.304 \pm 0.074}$	1.666 ± 0.159	1.848 ± 0.117	1.893 ± 0.080	1.919 ± 0.054
DR-DMP$_{MCMU}(2,8)$	0.676 ± 0.123	1.270 ± 0.168	1.681 ± 0.148	1.883 ± 0.058	1.927 ± 0.038	1.939 ± 0.036
DR-DMP$_{MCMU}(2,5)$	0.687 ± 0.052	1.264 ± 0.113	1.684 ± 0.130	1.897 ± 0.091	1.950 ± 0.056	1.962 ± 0.054
DR-DMP$_{MCMU}(2,2)$	0.704 ± 0.055	1.258 ± 0.130	$\mathbf{1.749 \pm 0.162}$	$\mathbf{1.976 \pm 0.029}$	$\mathbf{2.000 \pm 0.016}$	$\mathbf{2.006 \pm 0.017}$
DR-DMP$_{MCMU}(2,1)$	0.579 ± 0.076	1.103 ± 0.125	1.607 ± 0.131	1.885 ± 0.107	1.959 ± 0.055	1.972 ± 0.054
IDR-DMP$_{CMU}$	0.715 ± 0.140	1.195 ± 0.091	1.672 ± 0.121	1.952 ± 0.040	1.997 ± 0.034	2.004 ± 0.030
IDR-DMP$_{MCMU}(2)$	0.656 ± 0.058	1.174 ± 0.158	1.683 ± 0.169	1.937 ± 0.067	$\mathbf{2.013 \pm 0.030}$	$\mathbf{2.019 \pm 0.028}$

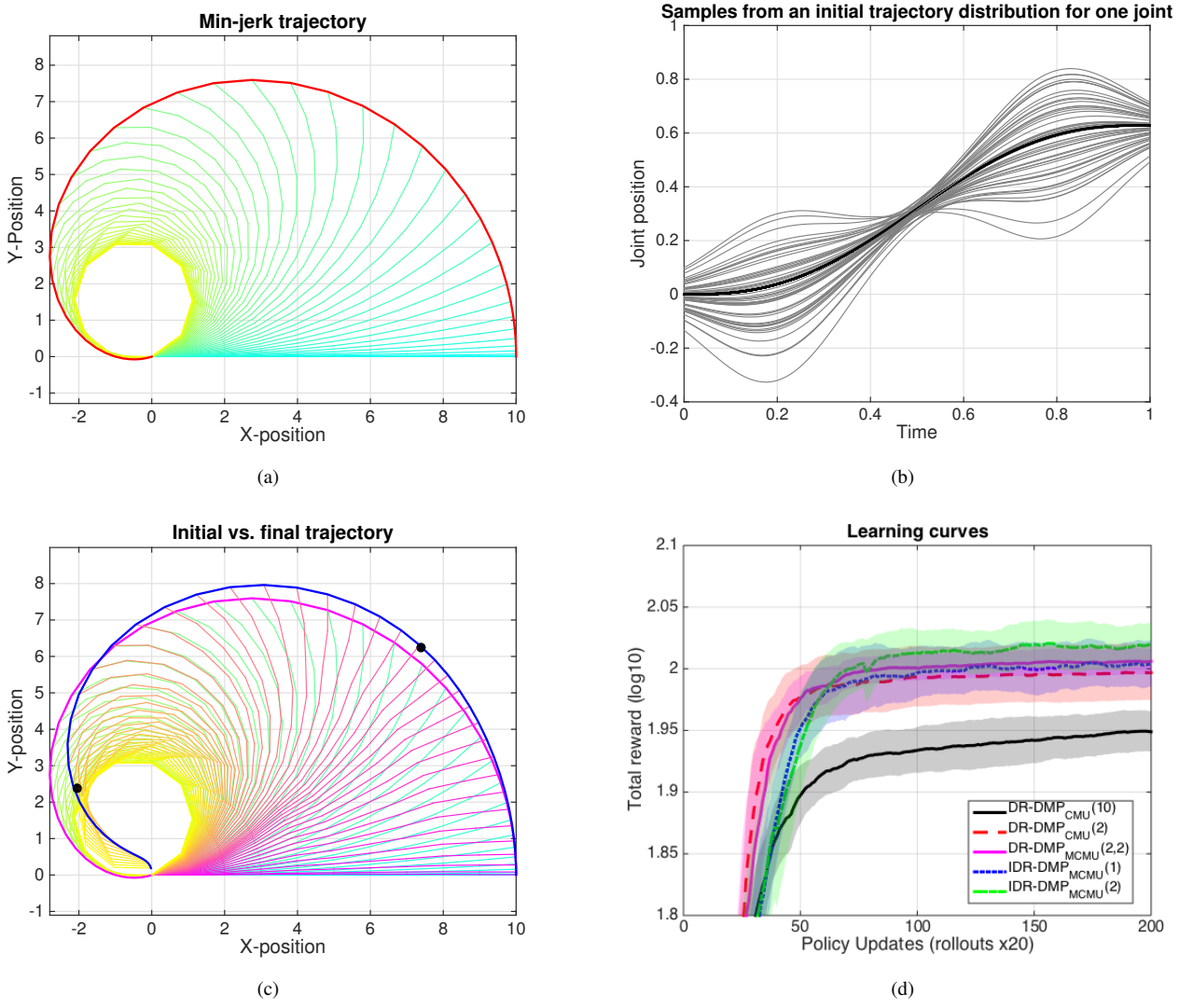
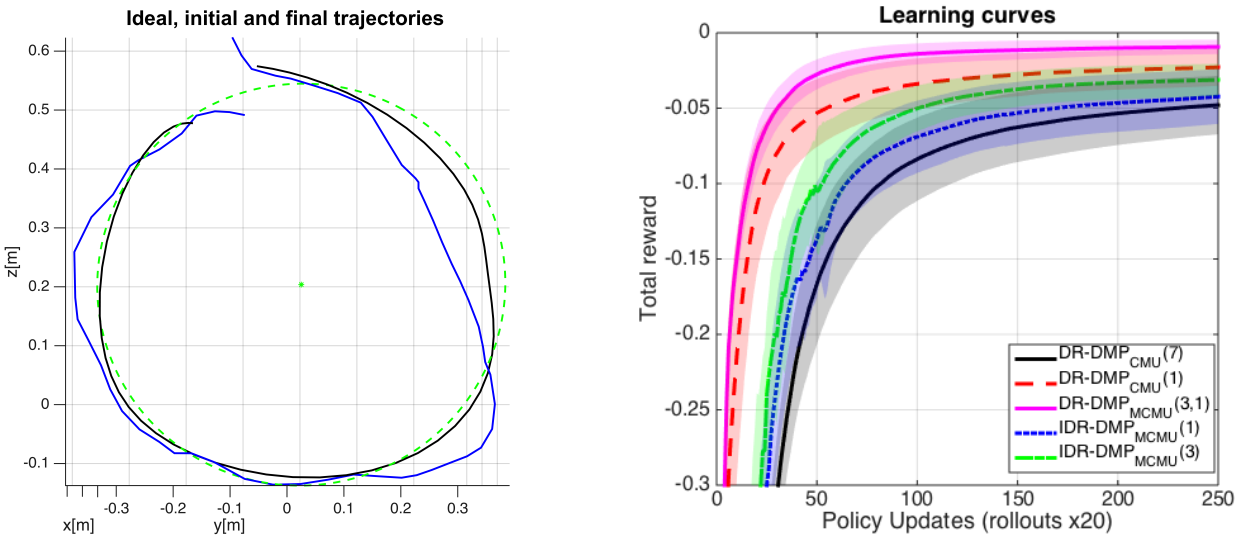


Fig. 2: 10-DoF planar robot arm experiment. (a) Joints min-jerk trajectory in the Cartesian XY space. The robot links move from the initial position (cyan color) to the end position (yellow), while the end-effector's trajectory is plotted in red. (b) Data generated to initialize the DMP for a single joint. (c) Initial trajectory (magenta) vs. final trajectory (blue) obtained with the IDR-DMP algorithm; via points plotted in black. (d) Learning curves showing mean and 95% confidence interval.



(a) (b)

Fig. 3: 7-DoF WAM robot circle-drawing experiment. (a) End-effector's trajectory (blue) resulting from a human kinesthetically teaching a WAM robot to track a circle and closest circle in the three-dimensional Cartesian space (green), which is the ideal trajectory; in black, one of the best solutions obtained through learning using DR-DMP. (b) Learning curves showing mean and 95% confidence intervals for some of the described methods.

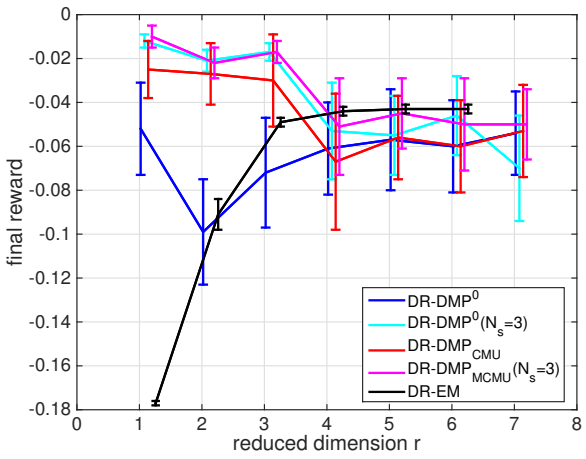


Fig. 4: Final rewards of different methods depending on chosen latent dimension.

reward values for $r = 4, 5, 6$ but the results for smaller dimensions show a premature convergence too far from optimal reward values.

- All pDR-DMP(M_f) methods display a very similar performance, regardless of the number M_f of parameters taken. They also present a greedier behavior, in the sense of reducing the exploration covariance $\mathbf{\Sigma}_\omega$ faster given the difficulties in performing dimensionality reduction with limited data on a high-dimensional space. In order for this alternative to work, M_f must verify that $M_f < N_t$ and M_f should also be smaller than the number of samples used for the policy update (note that a reuse of the previous samples is also performed).

C. 14-DoF dual-arm clothes-folding experiment

As a third experiment, we implemented the same framework on a dual-arm setting consisting of two Barrett's WAM robots, aiming to fold a polo shirt as seen in Fig. 1.

1) Initialization and reward function: We kinesthetically taught both robots to joinly fold a polo shirt, with a relatively wrong initial attempt as shown in Fig. 5(a). Then we performed 3 runs consisting of 15 policy updates of 12 rollouts each (totaling 180 real robot dual-arm motion executions for each run), with a reuse of the previous 12 rollouts for the PS update. We ran the standard method and the automatic dimensionality reduction for both $N_s = 1$ and $N_s = 3$. This added up to a total of 540 real robot executions with the dual-arm setup.

The trajectories were stored in Cartesian coordinates, using 3 variables for position and 3 more for orientation, totaling 12 DoF and, with 20 DMP weights per DoF, adding up to 240 DMP parameters. Additionally, an inverse kinematics algorithm was used to convert Cartesian position trajectories to joint trajectories, which then a computed-torque controller [33] would compliantly track.

As reward, we used a function that would penalize large joint accelerations, together with an indicator of how well the polo shirt was folded. To do so, we used a rooftop-placed *Kinect* camera to generate a depth map with which to evaluate the resulting wrinkleness of the polo. Additionally, we checked its rectangleness by color-segmenting the polo on the table and fitting a rectangle with the obtained result

TABLE V: Results for the circle-drawing experiment using real data from a 7-DoF WAM robot. Mean rewards and 95% confidence intervals for most variants of the DR-DMP methods are reported.

Method	1 update	10 updates	25 updates	50 updates	100 updates	200 updates
DR-DMP$^0(7)$	-1.812 ± 0.076	-0.834 ± 0.078	-0.355 ± 0.071	-0.165 ± 0.034	-0.085 ± 0.024	-0.054 ± 0.019
DR-DMP$^0(6)$	-1.801 ± 0.053	-0.836 ± 0.058	-0.359 ± 0.058	-0.169 ± 0.040	-0.093 ± 0.027	-0.060 ± 0.021
DR-DMP$^0(5)$	-1.810 ± 0.047	-0.834 ± 0.058	-0.372 ± 0.048	-0.164 ± 0.037	-0.086 ± 0.028	-0.057 ± 0.023
DR-DMP$^0(4)$	-1.879 ± 0.054	-0.901 ± 0.043	-0.405 ± 0.059	-0.189 ± 0.037	-0.093 ± 0.025	-0.061 ± 0.021
DR-DMP$^0(3)$	-1.523 ± 0.057	-0.765 ± 0.051	-0.340 ± 0.052	-0.162 ± 0.038	-0.096 ± 0.030	-0.072 ± 0.025
DR-DMP$^0(2)$	-1.833 ± 0.065	-0.906 ± 0.074	-0.422 ± 0.063	-0.230 ± 0.043	-0.140 ± 0.029	-0.099 ± 0.024
DR-DMP$^0(1)$	$\mathbf{-0.860 \pm 0.030}$	$\mathbf{-0.447 \pm 0.031}$	$\mathbf{-0.195 \pm 0.031}$	$\mathbf{-0.103 \pm 0.022}$	$\mathbf{-0.068 \pm 0.022}$	$\mathbf{-0.052 \pm 0.021}$
DR-DMP$_{CMU}(7)$	-1.970 ± 0.080	-0.907 ± 0.072	-0.386 ± 0.045	-0.167 ± 0.040	-0.084 ± 0.028	-0.053 ± 0.021
DR-DMP$_{CMU}(6)$	-1.949 ± 0.071	-0.878 ± 0.078	-0.367 ± 0.066	-0.175 ± 0.046	-0.090 ± 0.028	-0.060 ± 0.021
DR-DMP$_{CMU}(5)$	-1.925 ± 0.073	-0.897 ± 0.078	-0.410 ± 0.076	-0.192 ± 0.042	-0.092 ± 0.025	-0.056 ± 0.019
DR-DMP$_{CMU}(4)$	-2.146 ± 0.085	-1.108 ± 0.150	-0.386 ± 0.113	-0.174 ± 0.055	-0.094 ± 0.036	-0.067 ± 0.031
DR-DMP$_{CMU}(3)$	-0.678 ± 0.274	-0.330 ± 0.122	-0.141 ± 0.058	-0.073 ± 0.033	-0.044 ± 0.025	-0.030 ± 0.021
DR-DMP$_{CMU}(2)$	-0.758 ± 0.329	-0.401 ± 0.155	-0.173 ± 0.074	-0.085 ± 0.045	-0.043 ± 0.024	-0.027 ± 0.014
DR-DMP$_{CMU}(1)$	$\mathbf{-0.563 \pm 0.108}$	$\mathbf{-0.216 \pm 0.081}$	$\mathbf{-0.094 \pm 0.036}$	$\mathbf{-0.053 \pm 0.022}$	$\mathbf{-0.034 \pm 0.017}$	$\mathbf{-0.025 \pm 0.013}$
DR-DMP$_{CMU}(7)$	-1.970 ± 0.080	-0.907 ± 0.072	-0.386 ± 0.045	-0.167 ± 0.040	-0.084 ± 0.028	-0.053 ± 0.021
DR-DMP$_{CMU}(6)$	-1.949 ± 0.071	-0.878 ± 0.078	-0.367 ± 0.066	-0.175 ± 0.046	-0.090 ± 0.028	-0.060 ± 0.021
DR-DMP$_{CMU}(5)$	-1.925 ± 0.073	-0.897 ± 0.078	-0.410 ± 0.076	-0.192 ± 0.042	-0.092 ± 0.025	-0.056 ± 0.019
DR-DMP$_{CMU}(4)$	-2.146 ± 0.085	-1.108 ± 0.150	-0.386 ± 0.113	-0.174 ± 0.055	-0.094 ± 0.036	-0.067 ± 0.031
DR-DMP$_{CMU}(3)$	-0.678 ± 0.274	-0.330 ± 0.122	-0.141 ± 0.058	-0.073 ± 0.033	-0.044 ± 0.025	-0.030 ± 0.021
DR-DMP$_{CMU}(2)$	-0.758 ± 0.329	-0.401 ± 0.155	-0.173 ± 0.074	-0.085 ± 0.045	-0.043 ± 0.024	-0.027 ± 0.014
DR-DMP$_{CMU}(1)$	$\mathbf{-0.563 \pm 0.108}$	$\mathbf{-0.216 \pm 0.081}$	$\mathbf{-0.094 \pm 0.036}$	$\mathbf{-0.053 \pm 0.022}$	$\mathbf{-0.034 \pm 0.017}$	$\mathbf{-0.025 \pm 0.013}$
DR-DMP$^0_{MCMU}(3,7)$	-1.880 ± 0.080	-0.854 ± 0.061	-0.384 ± 0.047	-0.196 ± 0.041	-0.107 ± 0.032	-0.070 ± 0.024
DR-DMP$^0_{MCMU}(3,6)$	-1.937 ± 0.064	-0.901 ± 0.098	-0.371 ± 0.063	-0.164 ± 0.041	-0.075 ± 0.024	-0.046 ± 0.018
DR-DMP$^0_{MCMU}(3,5)$	-1.995 ± 0.087	-0.916 ± 0.072	-0.363 ± 0.067	-0.165 ± 0.039	-0.083 ± 0.027	-0.055 ± 0.019
DR-DMP$^0_{MCMU}(3,4)$	-2.169 ± 0.054	-1.108 ± 0.115	-0.355 ± 0.103	-0.154 ± 0.058	-0.081 ± 0.035	-0.053 ± 0.022
DR-DMP$^0_{MCMU}(3,3)$	-0.468 ± 0.016	-0.239 ± 0.020	-0.104 ± 0.016	-0.054 ± 0.011	-0.029 ± 0.006	-0.017 ± 0.004
DR-DMP$^0_{MCMU}(3,2)$	-0.499 ± 0.016	-0.272 ± 0.025	-0.117 ± 0.016	-0.057 ± 0.010	-0.031 ± 0.006	-0.021 ± 0.005
DR-DMP$^0_{MCMU}(3,1)$	$\mathbf{-0.462 \pm 0.033}$	$\mathbf{-0.146 \pm 0.014}$	$\mathbf{-0.061 \pm 0.012}$	$\mathbf{-0.031 \pm 0.008}$	$\mathbf{-0.019 \pm 0.006}$	$\mathbf{-0.012 \pm 0.003}$
DR-DMP$_{MCMU}(3,7)$	-1.970 ± 0.074	-0.872 ± 0.059	-0.390 ± 0.045	-0.181 ± 0.026	-0.086 ± 0.021	-0.050 ± 0.016
DR-DMP$_{MCMU}(3,6)$	-1.981 ± 0.053	-0.929 ± 0.117	-0.376 ± 0.086	-0.181 ± 0.060	-0.084 ± 0.036	-0.050 ± 0.021
DR-DMP$_{MCMU}(3,5)$	-1.951 ± 0.058	-0.851 ± 0.075	-0.320 ± 0.050	-0.133 ± 0.025	-0.068 ± 0.019	-0.045 ± 0.016
DR-DMP$_{MCMU}(3,4)$	-2.165 ± 0.053	-1.090 ± 0.133	-0.322 ± 0.107	-0.142 ± 0.059	-0.075 ± 0.034	-0.051 ± 0.022
DR-DMP$_{MCMU}(3,3)$	-0.456 ± 0.014	-0.244 ± 0.021	-0.112 ± 0.021	-0.057 ± 0.011	-0.030 ± 0.009	-0.017 ± 0.005
DR-DMP$_{MCMU}(3,2)$	-0.500 ± 0.019	-0.285 ± 0.025	-0.134 ± 0.022	-0.070 ± 0.016	-0.037 ± 0.009	-0.022 ± 0.007
DR-DMP$_{MCMU}(3,1)$	$\mathbf{-0.492 \pm 0.032}$	$\mathbf{-0.150 \pm 0.016}$	$\mathbf{-0.062 \pm 0.015}$	$\mathbf{-0.028 \pm 0.010}$	$\mathbf{-0.014 \pm 0.007}$	$\mathbf{-0.010 \pm 0.005}$
IDR-DMP$_{CMU}$	-1.826 ± 0.093	-0.815 ± 0.089	-0.300 ± 0.063	-0.137 ± 0.038	-0.069 ± 0.024	-0.047 ± 0.019
IDR-DMP$_{MCMU}(3)$	-1.955 ± 0.085	-0.807 ± 0.127	-0.240 ± 0.054	-0.105 ± 0.040	-0.050 ± 0.015	-0.033 ± 0.011
EM DR-DMP(6)	-2.288 ± 0.021	-0.884 ± 0.105	-0.180 ± 0.027	-0.086 ± 0.012	-0.051 ± 0.003	-0.043 ± 0.002
EM DR-DMP(5)	-2.274 ± 0.018	-0.975 ± 0.175	-0.217 ± 0.034	-0.094 ± 0.015	-0.056 ± 0.006	-0.043 ± 0.002
EM DR-DMP(4)	-2.363 ± 0.026	-0.926 ± 0.144	-0.194 ± 0.026	-0.100 ± 0.017	-0.059 ± 0.006	-0.044 ± 0.002
EM DR-DMP(3)	-2.050 ± 0.027	-1.075 ± 0.155	-0.231 ± 0.064	-0.103 ± 0.018	-0.063 ± 0.003	-0.049 ± 0.002
EM DR-DMP(2)	-2.271 ± 0.015	-1.417 ± 0.100	-0.335 ± 0.049	-0.203 ± 0.014	-0.145 ± 0.009	-0.091 ± 0.007
EM DR-DMP(1)	-1.111 ± 0.015	-0.653 ± 0.072	-0.283 ± 0.028	-0.196 ± 0.006	-0.182 ± 0.001	-0.177 ± 0.001
pDR-DMP(90)	-0.826 ± 0.000	-0.496 ± 0.029	-0.149 ± 0.025	-0.048 ± 0.009	-0.034 ± 0.003	-0.028 ± 0.002
pDR-DMP(75)	-0.825 ± 0.000	-0.487 ± 0.025	-0.132 ± 0.030	-0.044 ± 0.010	-0.032 ± 0.003	-0.026 ± 0.001
pDR-DMP(60)	-0.822 ± 0.000	-0.501 ± 0.031	-0.148 ± 0.026	-0.041 ± 0.005	-0.031 ± 0.002	-0.026 ± 0.002
pDR-DMP(45)	-0.972 ± 0.000	-0.550 ± 0.029	-0.163 ± 0.028	-0.053 ± 0.006	-0.035 ± 0.003	-0.028 ± 0.002
pDR-DMP(30)	-1.124 ± 0.000	-0.820 ± 0.025	-0.305 ± 0.042	-0.073 ± 0.010	-0.040 ± 0.004	-0.028 ± 0.002
pDR-DMP(15)	-0.791 ± 0.000	-0.587 ± 0.022	-0.196 ± 0.044	-0.076 ± 0.018	-0.048 ± 0.009	-0.033 ± 0.003

(see Fig. 5(a)). Therefore, the reward function used was:

$$R = -R_{\text{acceleration}} - R_{\text{color}} - R_{\text{depth}}, \tag{29}$$

where $R_{\text{acceleration}}$ is a term penalizing large acceleration commands at the joint level, R_{color} has a large penalizing value if the result after the motion does not have a rectangular projection on the table (see Fig. 5(a)), expressed as:

$$R_{\text{color}} = \frac{\#\text{pix. rectangle}}{\#\text{blue pix. rectangle}} \left[(a - a_{ref})^2 + (b - b_{ref})^2\right], \tag{30}$$

where a, b are the measured side lengths of the bounding rectangle in Fig. 5(a), and a_{ref}, b_{ref} are their reference values, given the polo dimensions. R_{depth} penalizes the outcome if the polo shirt presents a too wrinkled configuration after the motion (see Fig. 5(b)) and it is computed using the code

available from [34].

2) Results and discussion: Despite the efforts of the authors to reduce the variability of results wrt. environmental uncertainties, a slightest variation of the setup would change the outcome. The initial exact configuration of a garment hanging from two grasping points at the moment of starting the motion is hard to repeat with precision. We ran the initial attempt with the same DMP parameters (shown in Figs. 5(a) and 5(b)) 20 times with no exploration, yielding a mean and a standard deviation for the vision terms of the reward function of $R_{\text{depth}} = 0.473 \pm 0.029$ and $R_{\text{color}} = 0.799 \pm 0.088$. This uncertainty increased with exploration, resulting in slower learning curves than initially expected. Nonetheless, the resulting learning curves obtained from the experiments and displayed in Fig. 5(c) show:

- The standard representation of DMPs, with 240 parameters, leads to a long transient period of very small improvement, specially between epochs 4 and 10. This is due to the large number of parameters wrt. the number of rollouts performed.
- The automatic dimensionality reduction with $N_s = 1$ algorithm presents a better and more stable improvement behavior. Ending with a reduced dimension of $r = 6$, reduces the dimensionality in the parameter space down to 120 parameters which, specially in the early stages, allows to keep improving over epochs.
- The automatic dimensionality reduction with $N_s = 3$ algorithm, IDR-DMP(3,12) ending with a dimensionality of $r = 4$, meaning 80 DMP weights - a third of those in the standard method- has a quicker learning at the early epochs, thanks to being able to quickly eliminate unnecessary exploration. After a certain number of iterations, the IDR-DMP(3) algorithm ends with a very similar result to that of IDR-DMP(1).

A video showing some snapshots of this experiment is provided in the supplementary material. In such video, we can also see the complexity of the task itself as some humans struggle to correctly fold a polo shirt. Our method allows a dual-arm robot to improve its folding skill from an initial faulty demonstration.

Additionally, Fig. 5(d) shows a graphical interpretation of the 3 coordination matrices obtained by the IDR-DMP(3) method. Darker areas indicate a higher correlation than lighter ones. Knowing that within each 12×4 matrix the columns on the left are more relevant, some symmetries can be readily observed. For example, the z component appears very similar for both robot arms. The x and y components for the two arms show a barely symmetric pattern, which could presumably had been stronger if the arms had been placed in a perfectly symmetric configuration, which was not the case (see Fig. 1). We see this as a promising avenue for future research, namely to analyse ways of imposing symmetry constraints on either the motion of the two arms or the coordination matrices themselves so as to both speed up learning and improve solution quality.

V. Conclusions

Using DMPs as motion characterization for robot learning leads to a kind of exploration vs. exploitation tradeoff (i.e., learning speed vs. solution quality). Such tradeoff meaning that a good fitting of the initial trajectory yields too many parameters to effectively perform PS to improve the robot behavior, while too few parameters allows for faster improvements, but limit the optimality of the solution found after learning.

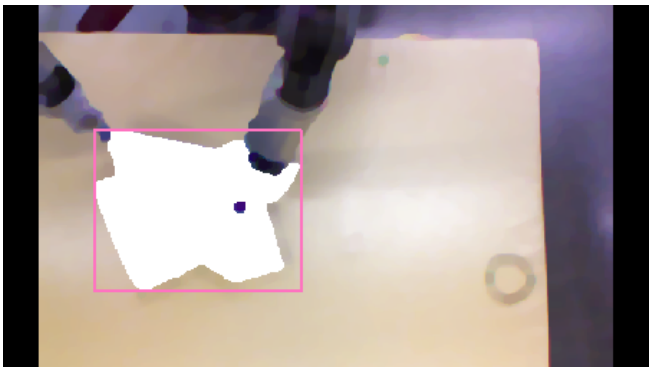
Throughout this paper, we proposed different ways to perform task-oriented linear dimensionality reduction of DMPs characterizations of motion. Such approaches help reduce the parameter dimensionality, allowing for faster convergence, while still keeping good optimality conditions.

We presented an algorithm to update the linear dimensionality reduction projection matrix with a task-related weighting in Section III-B, so that it better adapts to the directions expected to provide the most gain in performance in the following steps. In Section III-C, we showed how to remove unnecessary parameters from the trajectory representation, by discovering couplings between the robot's degrees of freedom and removing the redundant ones. Both these approaches were combined and extended to use several projection matrices in Section III-E, yielding improved behavior.

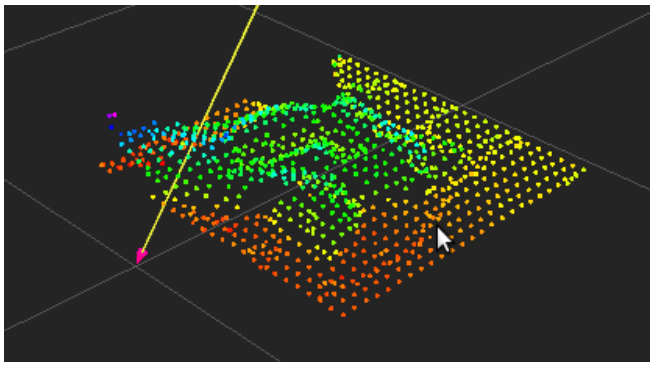
The results of the experiments performed (the fully-simulated, the hybrid real-data simulated, and the dual-arm real-robot experiment) clearly show the advantages of using dimensionality reduction for improving PS results with DMP motion characterizations.

In general, when fitting a robot motion with a certain parametrized movement primitive, it is common to have some overfitting that might result in meaningless exploration when learning. Such overfitting might be useful to have a wider range of exploration in early stages, but quickly eliminating it shows a significant improvement in the learning process of robotic skills.

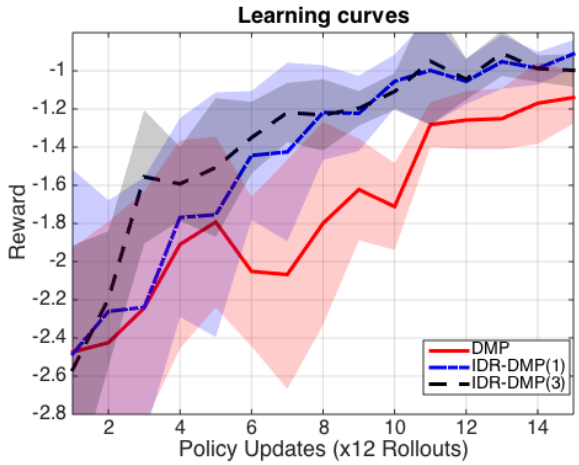
Additional Expectation-Maximization (EM) derivations [27] were tested for such linear dimensionality reduction, but those showed a more greedy behavior in the policy updates, resulting in premature convergence. Moreover, hand-crafting a reward function may not always be possible, it is often unsatisfactory and might lead to unexpected solutions. To overcome this shortcoming, inverse reinforcement learning may be used to infer a reward function for a certain task under some expertise assumptions on the demonstrated motions to the robot [35], and future developments of this work will go in this direction. Another direction of future work is to automatically decide the number of coordination matrices N_s, defined in Section III-E, which has been arbitrarily set along this work. A study of the complexity of trajectories and their piecewise linearity might lead to an accurate estimation of the number of such matrices. Finally, the analysis of the obtained coordination matrices $\mathbf{\Omega}$ for the dual-arm experiment unraveled the possibility of exploiting task symmetries to both speed up learning and improve solution quality, which seems also a promising idea to explore.



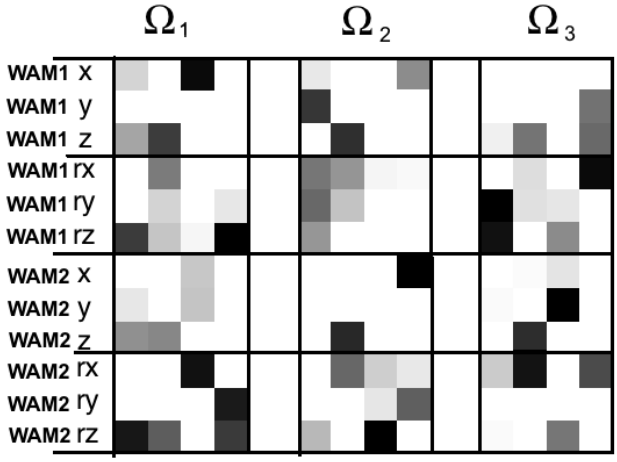
(a)



(b)



(c)



(d)

Fig. 5: 14-DoF dual-arm clothes-folding experiment. (a) Color segmentation of the polo shirt after an initial folding attempt. The blue color (as the polo color) was segmented and the number of blue pixels within the smallest rectangle containing it was counted. Then the ratio of blue pixels wrt. the total number of pixels within the rectangle was used for the reward function. (b) Depth image visualization from which the R_{depth} component of the reward function was computed. The mean gradient of the depth was used as a wrinkleness indicator. (c) Learning curves showing mean and standard deviation over the rollouts for each epoch, for the standard setting and two variants of the DR-DMP method. (d) Graphical representation of the synergies obtained for the IDR-DMP(3) method; black areas indicate a higher influence (in absolute value), while white areas represent a small influence.

TABLE VI: Results for the dual-arm real-robot experiment of folding clothes. The rewards for the three tested methods are shown with the average over rollouts and their standard deviation at each epoch.

method	**1 update**	**2 update**	**5 update**	**10 update**	**15 update**
DR-DMP$^0(12)$	-2.474 ± 0.546	-2.424 ± 0.634	-1.792 ± 0.447	-1.713 ± 0.228	-1.140 ± 0.136
IDR-DMP$_{CMU}$	-2.484 ± 0.966	-2.261 ± 0.582	-1.754 ± 0.641	-1.055 ± 0.141	-0.911 ± 0.074
IDR-DMP$_{MCMU}(3)$	-2.565 ± 0.650	-2.195 ± 0.354	-1.507 ± 0.364	-1.108 ± 0.095	-0.996 ± 0.086

REFERENCES

[1] N. A. Bernstein, "The co-ordination and regulation of movements". Oxford: Pergamon Press, 1967.

[2] A. J. Ijspeert, J. Nakanishi and S. Schaal, "Movement Imitation with Nonlinear Dyamical Systems in Humanoid Robots". *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 1398-1403, 2002.

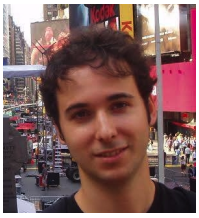
[3] J. Kober, K. Mülling, O. Krömer, C. H. Lampert and B. Schölkopf, "Movement Templates for Learning of Hitting and Batting". *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 853 - 858, 2010.

[4] L.Rozo, P. Jiménez and C. Torras, "Robot Learning from Demonstration of Force-based Tasks with Multiple Solution Trajectories", *15th IEEE Int. Conf. on Advanced Robotics*, pp. 124-129, 2011.

[5] L. Rozo, P. Jiménez and C. Torras, "A robot learning from demonstration framework to perform force-based manipulation tasks", *Intelligent Service Robotics*, vol. 6, no 1, pp. 33-51, 2013.

[6] L. Rozo, S. Calinon, D. Caldwell, P. Jimenez and C. Torras, "Learning Physical Collaborative Robot Behaviors from Human Demonstra-

tions", *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 513-527, 2016.
[7] S. M. Khansari-Zadeh and A. Billard, "Learning Stable Nonlinear Dynamical Systems with Gaussian Mixture Models". *IEEE Transactions on Robotics* , vol. 27, no 5, pp. 943-957, 2011.
[8] A. Billard, S. Calinon, R. Dillmann and S. Schaal, "Robot Programming by Demonstration." *Springer Handbook of Robotics*, part G, chapter 59.
[9] S. Calinon, F. D'halluin, E.L. Sauser, D.G. Caldwell and A.G. Billard, "Learning and reproduction of gestures by imitation. An approach based on Hidden Markov Model and Gaussian Mixture Regression". *IEEE Robotics and Automation Magazine*, vol 17, no. 2, pp. 44-54, 2010.
[10] A. Paraschos, G Neumann, C. Daniel, and J. Peters, "Probabilistic movement primitives". *In Proc. Neural Information Processing Systems (NIPS)*, Cambridge, MA: MIT Press., 2013.
[11] D. Nguyen-Tuong and J. Peters, "Learning Robot Dynamics for Computed Torque Control Using Local Gaussian Processes Regression". *ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems*, pp. 59-64, 2008.
[12] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, "Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors". *Neural Computation*, vol. 25, no. 2, pp. 328-373, 2013.
[13] A. J. Ijspeert, J. Nakanishi and S. Schaal, "Movement imitation with nonlinear dyamical systems in humanoid robots." *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 1398-1403, 2002.
[14] M. Deisenroth, G. Neumann and J. Peters, "A Survey on Policy Search for Robotics". *Foundations and Trends in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2011.
[15] J. Peters, S. Schaal, "Policy gradient methods for robotics ." *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots (IROS)*, pp. 2219-2225, 2006.
[16] J. Peters and S. Schaal, "Reinforcement Learning of Motor Skills with Policy Gradients". *Journal of Neural Networks*, vol. 21, no. 4, pp. 682-697, 2008.
[17] J. Peters, K. Mülling and Y. Altün, "Relative Entropy Policy Search". *24th National Conf. on Articial Intelligence*, pp. 182-189, 2011.
[18] C. Daniel, G. Neumann and J. Peters, "Hierarchical Relative Entropy Policy Search". *Journal of Machine Learning Research*, vol. 17, no. 93, pp. 1-50, 2012.
[19] E. Theodorou, J. Buchli and S. Schaal, "Reinforcement Learning of Motor Skills in High Dimensions: A Path Integral Approach". *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 2397 - 2403, 2010.
[20] E. Theodorou, J. Buchli and S. Schaal, "A Generalized Path Integral Control Approach to Reinforcement Learning". *Journal of Machine Learning Research*, vol. 11, pp. 3137-3181, 2010.
[21] F. Stulp, E. A. Theodorou, S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation" *IEEE Transactions on robotics*, vol. 28, no. 6, 2012.
[22] S. Levine, P. Pastor, A. Krizhevsky, D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection", *Int. Symposium on Experimental Robotics (ISER)*, 2016.
[23] S. Bitzer and S. Vijayakumar, "Latent spaces for dynamic movement primitives." *IEEE-RAS Int. Conf. on Humanoid Robots*, pp. 574 - 581, 2009.
[24] H. Ben Amor, O. Kroemer, U. Hillenbrand, G. Neumann, and J. Peters, "Generalization of human grasping for multi-fingered robot hands," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots (IROS)*, pp. 2043-2050, 2012.
[25] K. S. Luck, G. Neumann, E. Berger, J. Peters, and H. Ben Amor, "Latent space policy search for robotics." *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots (IROS)*, pp. 1434-1440, 2014.
[26] A. Colomé and C. Torras, Dimensionality reduction and motion coordination in learning trajectories with dynamic movement primitives, *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots (IROS)*, pp. 1414-1420, 2014.
[27] A. Colomé, G. Neumann, J. Peters and C. Torras, "Dimensionality reduction for probabilistic movement primitives", *Proc. IEEE-RAS Humanoid Robots*, pp. 794-800, 2014.
[28] N. Torres Alberto, M. Mistry and F. Stulp, "Computed Torque Control with Variable Gains through Gaussian Process Regression". *Proc. IEEE-RAS Humanoid Robots*, pp. 212 - 217, 2014.
[29] S. Kullback and R.A. Leibler, "On information and sufficiency". *Annals of Mathematical Statistics*, vol. 22 no. 1, pp. 79–86, 1951.
[30] D. Pardo, "Learning rest-to-rest Motor Coordination in Articulated Mobile Robots", *Ph.D. Dissertation*, 2009.
[31] P. Kormushev, S. Calinon and G. Caldwell, "Robot Motor Skill Coordination with EM-based Reinforcement Learning", *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots (IROS)*, pp. 3232 - 3237, 2010.
[32] M. Toussaint, "Lecture Notes: Gaussian Identites", available on `http://ipvs.informatik.uni-stuttgart.de/mlr/marc/notes/gaussians.pdf`, 2011.
[33] A. Colomé, A. Planells and C. Torras, "A friction-model-based framework for reinforcement learning of robotic tasks in non-rigid environments", *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 5649-5654, 2015.
[34] A. Ramisa, G. Alenyà, F. Moreno-Noguer, and C. Torras. "A 3D descriptor to detect task-oriented grasping points in clothing", *Pattern Recognition*, vol. 60, pp. 936-948, 2016.
[35] S. Zhifei and E. Meng Joo, "A survey of inverse reinforcement learning techniques", *Intelligent Computing and Cybernetics*, vol. 5, no. 3, pp. 293-311, 2012.



Adrià Colomé is a Postdoctoral researcher at the Robotics Institute in Barcelona. He obtained two B.Sc. degrees in Mathematics and Industrial Engineering in 2009, a M.Sc. degree (2011) and a PhD (2017) in Automatic Control, from the Technical University of Catalonia (UPC). Dr. Colomé has published papers on robot kinematics and dynamics, and also on movement primitives and direct policy search reinforcement learning. His current interests are now on efficient policy representations and context-dependency of robotic tasks. For more information: http://www.iri.upc.edu/people/acolome

Carme Torras (M'07, SM'11) is Research Professor at the Spanish Scientific Research Council (CSIC), and Head of the Perception and Manipulation group at the Robotics Institute in Barcelona. She holds M.Sc. degrees in Mathematics and Computer Science from the University of Barcelona and the University of Massachusetts, Amherst, respectively, and a Ph.D. degree in Computer Science from the Technical University of Catalonia (UPC). Prof. Torras has published five books and about three hundred papers in the areas of robot kinematics, computer vision, geometric reasoning, machine learning and manipulation planning. She has supervised 18 PhD theses and led 15 European projects, the latest being the Chist-Era project I-DRESS and the H2020 project IMAGINE. She has been Associate Vice-President for Publications of the IEEE Robotics and Automation Society (RAS), Editor of the IEEE Transactions on Robotics, and is currently an elected member of the Administrative Committee of IEEE RAS. She was awarded the Narcís Monturiol Medal of the Generalitat de Catalunya in 2000, and she became ECCAI Fellow in 2007, member of Academia Europaea in 2010, and member of the Royal Academy of Sciences and Arts of Barcelona in 2013. She has recently been awarded an ERC Advanced Grant with for the project "CLOTHILDE - CLOTH manIpulation Learning from DEmonstrations". For more information: http://www.iri.upc.edu/people/torras