Generalizing Movement Primitives
to New Situations

Jens Lundell[1], Murtaza Hazara[2(✉)], and Ville Kyrki[2]

[1] AASS Research Center, Örebro University, Örebro, Sweden
jens.lundell@oru.se
[2] School of Electrical Engineering, Aalto University, Espoo, Finland
{murtaza.hazara,ville.kyrki}@aalto.fi

Abstract. Although motor primitives (MPs) have been studied extensively, much less attention has been devoted to studying their generalization to new situations. To cope with varying conditions, a MP's policy encoding must support generalization over task parameters to avoid learning separate primitives for each condition. Local and linear parameterized models have been proposed to interpolate over task parameters to provide limited generalization.

In this paper, we present a global parametric motion primitive (GPDMP) which allows generalization beyond local or linear models. Primitives are modeled using a linear basis function model with global non-linear basis functions. The model is constructed from initial non-parametric primitives found using a single human demonstration and subsequent episodes of reinforcement learning to adapt the demonstrated skill to other task parameters. The initial models are then used to optimize the parameters of the global parametric model. Experiments with a ball-in-a-cup task with varying string lengths show that GPDMP allows greatly improved extrapolation compared to earlier local or linear models.

1 Introduction

Learning from demonstration (LfD) is a parametric supervised learning framework; the parameters of a model [1–5] are fine-tuned, thus fitting the model to the training data (demonstrations). Developing a model capable of adapting and generalizing to new unseen situations is one of the main challenges and objectives of supervised learning. In the context of motor primitives (MPs), a generalizable model can be translated into a model achieving a successful reproduction of an imitated task in a perturbed environment. For example, a generalizing model can reproduce a ball-in-a-cup task for a different string length unseen in the demonstrations; the model is interpolating when this new string length is in the range of demonstrations set; otherwise, it is extrapolating.

Recently, researchers have shown interest in generalizing MPs to new situations [6–9]. Such a generalization is achieved by parametrizing a policy encoding with respect to an evaluated environment condition. Although these parametric models are capable of interpolation, they are not guaranteed to extrapolate.
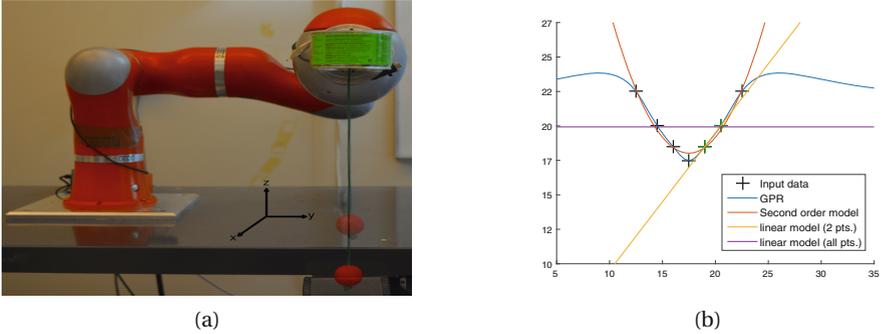
Fig. 1. (a) Ball-in-a-cup game with two different string lengths. (b) Models fit to 7 points of a non-linear function. Only the second order model captures the global pattern, while the GPR model tends to the mean when extrapolating, and the linear model is either (when trained with all samples) leaning toward the mean, or finds only the local pattern when fit to only two points marked by the green color. (Color figure online)

In fact, very few researchers have considered global policy encoding capable of extrapolation [10,11].

In this paper, we propose a new parametric LfD approach for generalizing an imitated task to new unseen situations. We selected the ball-in-a-cup task to assess how effective our method is in generalizing from initial demonstration with certain string length to changed lengths. The kinematics of the initial demonstration are encoded using a Dynamic Movement Primitive (DMP). Afterwards, the shape parameters of the DMP are optimized using PoWER [12] to adapt the skill to a few other string lengths. These training data are then used to build a global parametric model of the skill. Then, the global model can be used for generalizing model parameters (e.g. shape parameters of DMP) to new task parameters (e.g. string lengths) without re-learning the generalized model.

The main contribution of this paper is a novel global parametric MP model (GPDMP) based on DMPs which employs a linear basis function model with global non-linear basis functions. The representative power of the model can be controlled to avoid the over-fitting problem. We also show that the PDMP method proposed by Matsubara et al. [10] can be reduced to a linear special case of our parametric model. Furthermore, we propose a new mechanism for exploring the DMP's policy space. Experiments with a ball-in-a-cup task show that the proposed model greatly improves extrapolation capability over the existing local or linear models.

2   Related Work

To adapt LfD models to new environments, the model parameters need to be adjusted according to parameters characterizing the new environment or task.

Existing generalizable LfD models can be categorized as (i) generalization by design where the parameters are explicit in the model structure such as the goal of a DMP; (ii) generalization based on interpolation which uses a weighted combination of training models; and (iii) generalization by global linear models where the model parameters depend linearly on the environment/task parameters.

Kober et al. [13] utilized a cost regularized kernel regression (based on Gaussian process regression) for learning the mapping of new situations to meta-parameters including the initial position, goal, amplitude and the duration of the MPs; they model the automatic adjustment of the meta-parameters as a RL problem. The approach allows then adjusting these designed aspects of motion based on the task but does not enable modifying other characteristics of the trajectory (policy), making the approach suitable for learning tasks where the DMPs are adapted spatially and temporally without changing the overall shape of the motion, but unsuitable for tasks where the dynamics during motion are changed such as in this paper.

Researchers have recently shown interest also in generalizing DMP shape parameters to new situations [7–9,14–16]. The approaches are primarily based on local regression methods. For example, Da Silva et al. [7] extract lower dimensional manifolds (latent space) from learned policies using ISOMAP algorithm; they achieve a generalizable policy by mapping the manifolds (representing task parameters) to DMPs shape parameters. Support Vector Machines with local Gaussian kernels are used for learning the mapping. Similarly, Forte et al. [8] utilize Gaussian process regression (GPR) to learn the mapping of task parameters to DMP shape parameters. Ude et al. [9] and Nemec et al. [15] utilize Locally Weighted Regression (LWR) and kernels with positive weights for learning the mapping. Stulp et al. [14] learn the original shape parameters and generalize it with one single regression using Gaussian kernels. Mülling et al. [16] propose a linear mixture of MPs for generalization and refine the generalized behavior using RL.

Although the local regression approaches might interpolate within the range of demonstrations, their extrapolation capability is not guaranteed, which is mainly because of the kernels learning the local structure; thus they typically tend towards the mean of training data when extrapolating. This problem is illustrated using a simple example in Fig. 1(b) which shows local (GPR) and global (linear, second order) models fit to a set of points. When a line is fit to two points close to each other, it learns the local pattern allowing interpolation and extrapolation in a small neighborhood. If the line fit is made using all points, the fit tends to become poor everywhere. A local regression such as GPR performs well in interpolation, but not in extrapolation. When a well fitting higher order global model can be found, it typically outperforms the others in extrapolation.

Carrera et al. [6] developed a parametric MPs model based on a mixture of several DMPs. First, they record multiple demonstrations. A DMP model is fit to each demonstration, and a parametric value is assigned to it representing the task environment in which the demonstration was recorded. Then, they calculate the influence of each model using a distance function between the model parameters

and the parametric value describing the current environment perturbation. Using the influence of each model as its mixing coefficient, the mixture of models is computed at the acceleration level. Since it is a linear combination with positive coefficients, the mixture model is not expected to be capable of extrapolation. In fact, we have observed in our experiments (see Sect. 4.3) that this model is incapable of extrapolation in a Ball-in-a-Cup task.

Calinon et al. [11] proposed an MPs model based on a Gaussian mixture model and generalize it to new situations using expectation maximization. Although their model is capable of linear extrapolation, it is only applicable when the task parameters can be represented in the form of coordinate systems.

All things considered, few researchers [10] have considered the extrapolation capability in generalizing task parameters to model parameters, which is the main focus of this paper. We show (see Sect. 3.3) that the model proposed by Matsubara et al. [10] is a linear special case of the proposed parametric model.

3   Method

In this section, we review dynamic movement primitives (DMPs). After that, we clarify our global parametric dynamic movement primitives (GPDMPs) method which incorporates both linear and non-linear parametric models. Besides that, we reformulate the parametric DMP method proposed by Matsubara et al. [10] and demonstrate how it can be reduced to a linear special case of the proposed approach.

3.1   Dynamic Movement Primitives

DMPs encode a policy for a one-dimensional system using two differential equations. The first differential equation

$$\dot{z} = -\tau \alpha_z z \tag{1}$$

formulates a canonical system where z denotes the phase of a movement; $\tau = \frac{1}{T}$ represents the time constant where T is the duration of a demonstrated motion, and α_z is a constant controlling the speed of the canonical system. This first order system resembles an adjustable clock driving the transform system

$$\frac{1}{\tau}\ddot{x} = \alpha_x(\beta_x(g - x) - \dot{x}) + f(z; \mathbf{w}) \tag{2}$$

consisting of a simple linear dynamical system acting like a spring damper perturbed by a non-linear component (forcing function) $f(z; \mathbf{w})$. x denotes the state of the system, and g represents the goal. The linear system is critically damped by setting the gains as $\alpha_x = \frac{1}{4}\beta_x$. The forcing function

$$f(z; \mathbf{w}) = \mathbf{w}^T \mathbf{g} \tag{3}$$

controls the trajectory of the system using a time-parameterized kernel vector $\mathbf{g}$ and a modifiable policy parameter vector (shape parameters) $\mathbf{w}$. Each element of the kernel vector

$$[\mathbf{g}]_n = \frac{\psi^n(z)z}{\sum_{n=1}^{N} \psi^n(z)}(g - x_0) \tag{4}$$

is determined by a normalized basis function $\psi^n(z)$ multiplied by the phase variable z and the scaling factor $(g - x_0)$ allowing for the spatial scaling of the resulting trajectory. Normally, a radial basis function (RBF) kernel

$$\psi^n(z) = \exp(-h_n(z - c_n)^2) \tag{5}$$

is selected as the basis function. The centres of kernels (c_n) are usually equi-spaced in time spanning the whole demonstrated trajectory. It is also a common practice to choose the same temporal width $(h_n = \frac{2}{3}|c_n - c_{n-1}|)$ for all kernels. Furthermore, the contribution of the non-linear component (3) decays exponentially by including the phase variable z in the kernels. Hence, the transform system (2) converges to the goal g.

The shape parameter vector $\mathbf{w}$ can be learned using weighted linear regression (LWR) [17]; firstly, the nominal forcing function f^{ref} is retrieved by integrating the transform system (2) with respect a human demonstration x^{demo}; next, the shape parameter for every kernel is estimated using

$$[\mathbf{w}]_n = (\mathbf{Z}^T \Psi \mathbf{Z})^{-1} \mathbf{Z}^T \Psi \mathbf{f}^{ref} \tag{6}$$

where $[\mathbf{f}^{ref}]_t = f_t^{ref}$, $[\mathbf{Z}]_t = z_t$, and $\Psi = \text{diag}(\psi_1^n, ..., \psi_t^n, ..., \psi_T^n)$.

3.2   Global Parametric Dynamic Movement Primitives

Using DMPs, a task can be imitated from a human demonstration; however, the reproduced task cannot be adapted to different environment conditions. To overcome this limitation, we have integrated a parametric model to DMPs capturing the variability of a task from multiple demonstrations. We transform the basic forcing function (3) into a parametric forcing function

$$f(z, \mathbf{l}; \mathbf{w}) = \mathbf{w}^T(\mathbf{l})\mathbf{g} \tag{7}$$

where the kernel weight vector $\mathbf{w}$ is parametrized using a parameter vector $\mathbf{l}$ of measurable environment factors.

We model the dependency of the weights with respect to parameters as a linear combination of J basis vectors $\mathbf{v}_i$ with coefficients depending on parameters in a non-linear fashion,

$$\mathbf{w}(\mathbf{l}) = \sum_{i=0}^{J} \phi_i(\mathbf{l})\mathbf{v}_i \tag{8}$$

where $\phi_i(\mathbf{l})$ is a function describing the coefficient of the ith basis vector $\mathbf{v}_i$.

For a chosen non-linear basis (known functions ϕ_i), the basis vectors can be chosen by minimizing the difference between modeled and initial non-parametric DMP shape parameters,

$$\arg \min_{\mathbf{v}_{0,\ldots,J}} \sum_{k=1}^{K} \left\| \mathbf{w}(\mathbf{l}_k) - \mathbf{w}^k \right\|_2 \tag{9}$$

where $\mathbf{w}^k$ denotes the weight vector of a non-parametric DMP optimized for parameter values $\mathbf{l}_k$. The initial weights can be merely imitated from a human demonstration using (6) or improved using a policy search method [18]. In either case, reproducing an imitated task using $\mathbf{w}^k$ should lead to a successful performance in an environment parametrized by $\mathbf{l}_k$.

The formulation captures linear models such as [10] as a special case. Considering a single parameter l for presentational simplicity, the linear model can be written

$$\mathbf{w}(l) = l\mathbf{v}_1 + \mathbf{v}_0. \tag{10}$$

In the next section we show the equivalence of (10) to the mode presented in [10].

To optimize the linear model using DMPs for two parameter values, each element of the weight vectors $[\hat{\mathbf{v}}_1]_i$ and $[\hat{\mathbf{v}}_0]_i$ can be estimated independently using

$$[\hat{\mathbf{v}}_1]_i = \frac{[\mathbf{w}^1]_i - [\mathbf{w}^2]_i}{l_1 - l_2} \tag{11}$$

$$[\hat{\mathbf{v}}_0]_i = [\mathbf{w}^1]_i - l_i[\hat{\mathbf{v}}_1]_i \tag{12}$$

The linear model requires thus $2N$ parameters where N refers to the number of kernels $\mathbf{g}$.

For a general polynomial model in one parameter, the non-linear basis is

$$\phi(l) = (1 \; l \; l^2 \ldots \; l^J). \tag{13}$$

The number of initial DMPs must then be at least equal to or greater than the order of the model to avoid unconstrained optimization problems. In the experimental part of the paper, we consider a second order model in one parameter, so that

$$\mathbf{w}(l) = l^2\mathbf{v}_2 + l\mathbf{v}_1 + \mathbf{v}_0. \tag{14}$$

The model choice for a particular application is a compromise between complexity of the attractor landscape that can be modeled and overfitting due to insufficient data.

3.3   Relationship to Matsubara's PDMP

Matsubara et al. [10] have proposed a PDMP method for learning parametric attractor landscape by extracting a small number of common factors from M

demonstrations. Firstly, these M demonstrations are aligned so that they have the same size. Then, the nominal forcing function matrix is generated using

$$\mathbf{F}_{all}^{ref} = \left(\mathbf{f}_1^{ref}\ \mathbf{f}_2^{ref}\ ...\ \mathbf{f}_M^{ref}\right) \tag{15}$$

where $\mathbf{f}_m^{ref}$ represents the reference forcing function calculated for the m-th demonstration $\mathbf{x}_m^{demo}$. Next, using the singular value decomposition of the nominal forcing functions matrix

$$\mathbf{F}_{all}^{ref} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \tag{16}$$

the matrix of desired basis $\mathbf{F}^{basis}$ function is created from the first J columns of $\mathbf{V}$. They estimate the forcing function using

$$\hat{f}(z, s; \mathbf{w}) = \Sigma_{j=1}^J s_j b_j(z; \mathbf{w}) \tag{17}$$

where the basis function

$$b_j = \mathbf{w}_j^T \mathbf{g} \tag{18}$$

is a weighted sum of kernels $[\mathbf{g}]_n$ (4). The weight vector $\mathbf{w}$ can be calculated using least square fitting or LWR as

$$[\mathbf{w}_j]_n = (\mathbf{Z}^T\boldsymbol{\Psi}\mathbf{Z})^{-1}\mathbf{Z}^T\boldsymbol{\Psi}\mathbf{F}^{basis} \tag{19}$$

In addition, the hyper parameter is calculated using

$$s_j = \beta_{j,1}l + \beta_{j,2} \tag{20}$$

where l represents an environment parameter. The structure of the basis function (18) allows for further simplification of the forcing function (17). In fact, by substituting (20) into (17), we get

$$\begin{aligned}
\hat{f}(z, s; \mathbf{w}) &= \Sigma_{j=1}^J (\beta_{j,1}l + \beta_{j,2})b_j(z; \mathbf{w}) \\
&= \Sigma_{j=1}^J (\beta_{j,1}l\mathbf{w}_j^T\mathbf{g}) + \Sigma_{j=1}^J (\beta_{j,2}\mathbf{w}_j^T\mathbf{g}) \\
&= (\Sigma_{j=1}^J \beta_{j,1}\mathbf{w}_j^T)l\mathbf{g} + (\Sigma_{j=1}^J \beta_{j,2}\mathbf{w}_j^T)\mathbf{g} \\
&= \mathbf{v}_1^T l\mathbf{g} + \mathbf{v}_0^T\mathbf{g} \\
&= (\mathbf{v}_1 l + \mathbf{v}_0)^T\mathbf{g}
\end{aligned} \tag{21}$$

which is equivalent to our linear parametric model (10). However, Matsubara's approach is computationally more complex. In fact, their method involves three main processes: an SVD decomposition of the matrix F^{basis} of desired basis functions, calculating J kernel vectors $\mathbf{w}$ for basis functions b_j, and computing J style parameters $\beta_{j,1}$ and $\beta_{j,1}$. The complexity of the SVD (16) is $O(s_d^2 M + M^3)$ where s_d refers to the size of each one of M demonstrations. Moreover, $J \times N$ basis function parameters and $J \times 2$ style parameters need to be estimated using least square fitting. The number of desired basis functions $J \geq 2$ should be at least two; otherwise, the learned model will be too general failing to capture

the variability of a task. On the other hand, our linear parametric model (8) requires only the estimation of $2 \times N$ parameters. Hence, our approach is simpler, less computationally complex, and at least as representative as the Mastubara's method. Furthermore, our GPDMP approach accommodates higher order parametric models (8), thus allowing the generalization of skills with more complex dependencies.

3.4 Reinforcement Learning

Executing a DMP with imitated shape parameters might not lead to a successful reproduction of a task. One way to refine the shape parameters is to learn them through trial-and-error using policy search reinforcement learning (RL). Next, we briefly review the state-of-the-art policy search method PoWER [12] which was used in this work to optimize individual primitives.

PoWER (see Algorithm 1) updates the DMP shape parameters $\boldsymbol{\theta} \equiv \mathbf{w}$ iteratively. In each iteration, (several) stochastic roll-out(s) of the task is performed, each of which is achieved by adding random (Gaussian) noise to the DMPs shape parameters. Each noisy vector is weighted by the returned accumulated reward. Hence, the higher the returned reward, the more the noisy vector contribute to the updated policy parameters. This exploration process continues until the algorithm converges to the optimal policy.

Algorithm 1. Pseudocode of the PoWER [12] algorithm for a one-dimensional policy.

Input: The initial policy parameters $\boldsymbol{\theta}$, the exploration variance $\boldsymbol{\Sigma}$

1: **repeat**
2:     *Sample*: Perform rollout(s) using $\mathbf{a} = (\theta + \epsilon_\mathbf{t})^\mathbf{T} \phi(\mathbf{s}, \mathbf{t})$ with
        $\epsilon_\mathbf{t}^\mathbf{T} \phi(\mathbf{s}, \mathbf{t}) \sim \mathcal{N}(\mathbf{0}, \phi(\mathbf{s}, \mathbf{t})^\mathbf{T} \hat{\boldsymbol{\Sigma}} \phi(\mathbf{s}, \mathbf{t}))$ as stochastic
        policy and collect all $(\mathbf{t}, \mathbf{s_t^h}, \mathbf{a_t^h}, \mathbf{s_{t+1}^h}, \epsilon_\mathbf{t}^\mathbf{h}, \mathbf{r_{t+1}^h})$ for
        $\mathbf{t} = \{\mathbf{1}, \mathbf{2}, \ldots, \mathbf{T+1}\}$.
3:     *Estimate*: Use unbiased estimate

$$\hat{Q}^\pi(s, a, t) = \sum_{\widetilde{t}=t}^{T} r(s_{\widetilde{t}}, a_{\widetilde{t}}, s_{\widetilde{t}+1}, \widetilde{t}).$$

4:     *Reweight*: Compute importance weights and reweigh rollouts, discard low-importance rollouts.
5:     *Update* policy using

$$\theta_{k+1} = \theta_k + \frac{\langle \sum_{t=1}^{T} \epsilon_t Q^\pi(\mathbf{s}, \mathbf{a}, t) \rangle_{w(\tau)}}{\langle \sum_{t=1}^{T} Q^\pi(\mathbf{s}, \mathbf{a}, t) \rangle_{w(\tau)}}$$

6: **until** convergance $\theta_{k+1} \approx \theta_k$

The structure of the noise is a key element influencing the convergence speed of a policy search method but the choice is a trade-off. In the case of DMP shape parameters and uncorrelated noise, high noise variance causes large accelerations of the system, causing a safety hazard and possibly surpassing the physical capabilities of the robot. In contrast, low noise variance makes the learning process slow.

To address this trade-off, we propose to use correlated noise instead of the earlier works employing uncorrelated noise. Since the elements of a DMP parameter vector correspond to temporally ordered perturbing forces, we want to control their temporal statistics. To achieve this, an intuitive structure for the covariance matrix $\Sigma = \mathbf{R}^{-1}$ can be used where the quadratic control cost matrix

$$\mathbf{R} = \sum_{k=1}^{K} w_k \mathbf{A}_k^T \mathbf{A}_k \qquad (22)$$

is a weighted combination of quadratic costs related to finite difference matrices $\mathbf{A}_1 \cdots \mathbf{A}_K$. w_k denotes the weight of the k-th finite difference matrix, and k is the order of differentiation. This structure allows us then to control statistics of any order. In experiments, we consider variation only in acceleration (second order). Thus, $w_2 = 1$ and all other weights $w_k = 0, k \neq 2$, and the second order finite difference matrix $\mathbf{A}_2$ can be written

$$\mathbf{A}_2 = \begin{bmatrix} 1 & 0 & 0 & & 0 & 0 & 0 \\ -2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -2 & 1 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & 1 & -2 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -2 \\ 0 & 0 & 0 & & 0 & 0 & 1 \end{bmatrix}. \qquad (23)$$

With this covariance matrix, the noise signal is smooth (see Fig. 2(a) due to limited acceleration and it has small magnitude in the beginning and at the end of the trajectory. Hence, safe exploration is provided. It is worth mentioning that a similar covariance matrix has been applied in [19,20] for direct trajectory encoding.

In order to control the magnitude of noise, we used a further modified covariance matrix $\Sigma = \gamma \beta \mathbf{R}^{-1}$ where γ is a constant controlling the initial magnitude and convergence factor

$$\beta = \frac{1}{\sum_{i=1}^{I} r_i^2}, \qquad (24)$$

reduces the magnitude of noise as the policy search algorithm is converging to the optimal policy.

4   Experimental Evaluation

We studied experimentally the generalization performance of the proposed model using a Ball-in-a-Cup task taught to KUKA LBR 4+ initially using
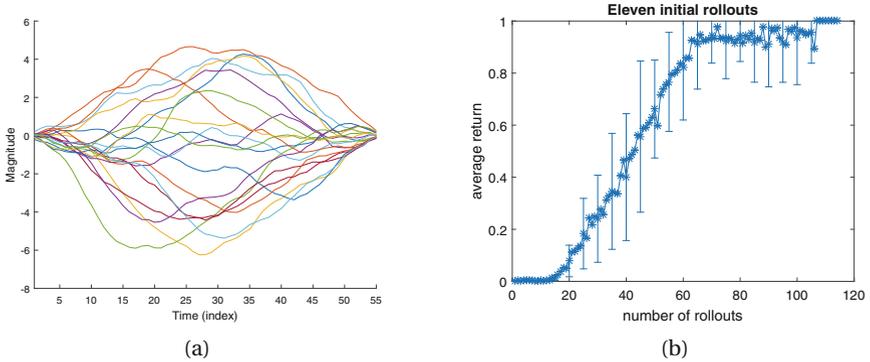
Fig. 2. (a) Noises ϵ sampled from a zero mean multivariate Gaussian distribution $\mathcal{N}(0, \boldsymbol{\Sigma})$ with $\gamma = 1$ and $\beta = 0.01$. (b) Mean and variance of returns over 12 trials.

kinesthetic teaching. In this section, we explain the scenario, compare the proposed noise (proposal) generation to standard uncorrelated noise in terms of convergence speed, and study the extrapolation capability of the model.

4.1 Ball-in-a-Cup Task

The Ball-in-a-Cup game consists of a cup, a string, and a ball; the ball is attached to the cup by the string (see Fig. 1(a)). The objective of the game is to get the ball in the cup by moving the cup in a suitable fashion. In practice, the cup needs to be moved back and forth at first; then, a movement is induced on the cup, thus pulling the ball up and catching it with the cup.

We chose the Ball-in-a-Cup game because variation in the environment can be generated simply by changing the string length. The string length is observable and easy to evaluate, thus providing a suitable parameter representing the environment variation. Nevertheless, changing the length requires a complex change in the motion to succeed in the game. Hence, the generalization capability of a parametric LfD model can be easily assessed using this game.

The state of the robot is defined in a seven dimensional space $\mathbf{X} = \{x, y, z, q_x, q_y, q_z, q_w\}$, where $\mathbf{X}_p = \{x, y, z\}$ represents the position of the robot end effector (cup), and $\mathbf{X}_q = \{q_x, q_y, q_z, q_w\}$ formulates its orientation using a quaternion. The ball-in-a-cup is essentially a two-dimensional game and thus only motion along two axes, y and z was used. In the demonstration phase, the robot was set compliant along y and z, while it was set stiff rotationally and along x, which were considered as constant states. The plane spanning y and z was orthogonal with respect to the table upon which the robot was mounted (see Fig. 1(a)).

The trajectories along y and z were encoded using separate DMPs with same number of parameters. We found experimentally that 55 kernels (shape parameters) were required so that the reproduced movement was able to put the ball above the rim of the cup in the execution phase. In total, 110 shape

parameters were then learned using LWR (6). However, using these initial shape parameters, the reproduced movement did not put the ball back into the cup. Hence, the shape parameters were optimized in a trial-and-error fashion using RL as described in Sect. 3.4.

Reward function is the most fundamental ingredient of RL. We formulated the reward function similar to [12] as

$$r(t) = \begin{cases} e^{-\alpha d^2}, & \text{if } t = t_c, \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

where t_c denotes the time instant when the ball crosses the rim of the cup with a downward motion; d represents the horizontal distance between the rim of the cup and the centre of the ball; and α is a scaling parameter set to 100 in our experiments. The closer the ball is to the rim of the cup, the higher the reward will be. As the shape parameters are fine-tuned in a trial-and-error approach, the ball would get closer to the cup. Furthermore, the reward is zero if the ball does not reach above the rim of the cup. Without such a constraint, the RL algorithm might converge to a policy where the ball is tossed to the bottom of the cup.

4.2   RL Convergence Rate

Figure 2(b) depicts the convergence rate for the RL algorithm with the proposed method starting from an imitated trajectory. Figure 2(b) shows both mean and variance of returns over 12 trials. On average a total of 80 rollouts (including 11 initial rollouts) are required for the policy to converge to an optimal one where the robot repeatedly succeeds at bringing the ball into the cup. After the ball went into the cup for the first time, on average 11 additional rollouts were required for the policy to converge. The convergence rate is similar to [12] where 75 rollouts were typically required for convergence. However, the results are not directly comparable due to differences in hardware realizations and human demonstration quality.

Figure 3 shows a comparison between the proposed correlated (blue) and earlier uncorrelated (red) exploration noises. The graph on left (Fig. 3(a)) shows that the proposed correlated noise improves the convergence rate significantly. The slow learning rate of uncorrelated noise is partially due to the small variance of the sampled noise in comparison to the correlated noise as shown in Fig. 3(b). However, larger noise variance was not feasible in the uncorrelated case because of the required accelerations were not physically realizable. This is demonstrated in Fig. 3(c) which shows accelerations in three cases: original demonstration, correlated noise and uncorrelated noise. Figure 3 shows that although the magnitude of the noise is smaller in the uncorrelated case, the learned policy requires much larger accelerations. After 36 iterations, the learned policy with uncorrelated exploration became infeasible to be executed on the real robot as it required more acceleration than the robot was physically able to realize.
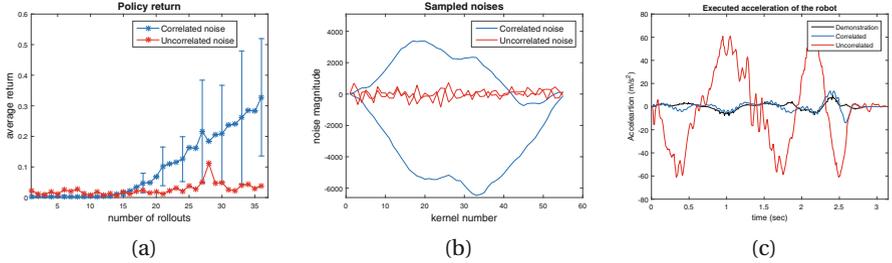
Fig. 3. (a) Returns for uncorrelated (red) and correlated (blue) exploration noise. (b) Two samples of correlated and uncorrelated noise. (c) Acceleration of end-effector in y-direction for demonstration (black), and policies after 36 roll-outs with uncorrelated (red) and correlated (blue) exploration noise. (Color figure online)

4.3 Generalization Capability

We evaluated both the proposed linear 10 and second order (14) GPDMP models for generalizing the DMP policy (shape) parameters. As a comparison, we used the parametric model by Carrera et al. [6] as a recent example of a data-driven local regression model.

All models require training data with varying string lengths. The training data was collected from a single demonstration with a string length of 40 cm. Starting from the shape parameters derived from this demonstration, the parameters were then learned using PoWER for string lengths of $32, 35, 38, 40$ and 41 cm. It should be noted that during the RL, executing the task with shape parameters learned for a specific length did not lead to a successful reproduction for another string length in the training set.

We performed two experiments, both studying the range of interpolation and extrapolation obtained by the models, with varying number of training data. In both experiments, a reproduction was considered to be successful if 5 consecutive replications of the Ball-in-a-Cup task with the same shape parameters put the ball into the cup.

Generalization over Minor Variations. In the first experiment, we studied generalization over minor variations by extracting parametric models from two training samples with string lengths of 38 and 41 cm. As there were only two training samples, only the linear variant 10 and the locally interpolating model of Carrera et al. [6] were used. The range of validity for these is shown in Fig. 4, the red line showing the range of validity, and (a) and (b) denoting Carrera et al. and the linear GPDMP, correspondingly. The training samples are shown with crosses. Both models were capable of interpolating successfully within the range of the training samples. In addition, the proposed linear model was able to extrapolate within ± 2 cm from the training samples. The result demonstrates that models using a positive linear combination of training models, such as Carrera's and others in the literature, are not well suited for extrapolation as they
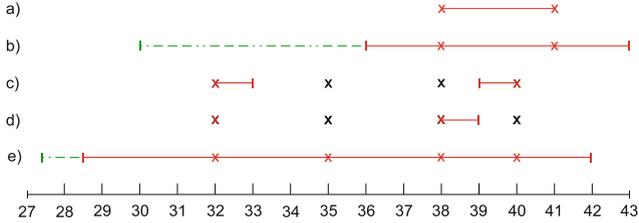
Fig. 4. Validity ranges for different models (red lines). Input data indicated as X. (a)–(b) small task variation: (a) Carrera et al. [6]; (b) linear GPDMP; (c)-(e) larger task variation: (c) Carrera et al.; (d) linear GPDMP; (e) second order GPDMP. (Color figure online)

tend towards the mean of training data when extrapolating as discussed in Sec. 2. However, a simple linear model is global and capable of extrapolation when the variation in the task is minor.

Generalization over Larger Variations. In the second experiment, we studied larger variations by extracting Carrera's PDMP [6], the linear GPDMP model 10, and the second order GPDMP (14) using a dataset of four samples with string lengths of 32, 35, 38, and 40 cm. The range of validity for each of these is shown in Fig. 4, (c) denoting Carrera et al., (d) the linear GPDMP, and (e) the second-order GPDMP.

Both Carrera's and the linear GPDMP show poor performance, capable of limited interpolation and missing also some of the training samples. The reasons for the failures appear to be different: Carrera's model uses a positive linear combination of the training data weighted inversely proportional to a distance metric in the task parameter space. With an optimal distance metric, the model should at least be able to replicate the training samples. We used the metric proposed in the original paper in our experiment but believe that with a more suitable metric the model would be likely to be able to interpolate successfully in this experiment. Nevertheless, success in extrapolation would be unlikely as explained earlier. Similarly, models employing Gaussian process regression or support vector machines would be unlikely to perform better in extrapolation. The failure of the linear GPDMP is likely due to the fact that the linear model is simply incapable of representing the complex relationship between the task and policy parameters. The linear approach by Matsubara et al. [10] would be likely to suffer from the same problem.

In contrast to the above, the proposed second order GPDMP (e in Fig. 4) is capable of both interpolation over the whole range and a surprising range of extrapolation within the range of $[-3.5\,\mathrm{cm}, +2\,\mathrm{cm}]$. It thus greatly outperforms the others. This demonstrates that global non-linear models are clearly beneficial for representing parametric policies. The choice of a model complexity for a particular application is not trivial, but model selection criteria based on e.g. information theoretic metrics could be used. In our experiments, the achieved

range of extrapolation already approached the performance limits of the physical system and therefore we did not study how higher order models could have increased the extrapolation capability even further.

Extrapolation Using a One-Dimensional Basis for Policy. To further study the complexity of policies needed for extrapolation, we performed a separate experiment with a linear GPDMP fit to demonstrations with lengths 38 and 41 cm, identical to the first experiment. Instead of using the model as such, we searched experimentally, if a task parameter value other than the real one would lead to success. Thus, in effect we studied if non-linear coefficients for the one-dimensional basis 10 would allow generalization, and found that this is indeed the case. Figure 5 shows the found task parameter values (string lengths) that lead to success versus their actual values. The two training points used to determine the basis using (11–12) are shown in red. As seen in Fig. 5, in the neighborhood of the training points, a line fit would have small residual error, showing that in that neighborhood a linear model is valid, as also apparent from the earlier Fig. 4. However, the one-dimensional basis ($\mathbf{v}_1$) is sufficient for significant further generalization (up to -8 cm) if non-linear coefficients are used. This extended range of validity is shown with green dashed line in Fig. 4 and is only slightly less than that of the second order model. It should be noted, however, that this is not a free lunch; the linear space coefficients were found by trial and error and the two training points do not allow to determine the non-linearity. Nevertheless, low dimensional vector spaces appear surprisingly powerful in policy representation, but simple linear transforms of task variables are not sufficient for coefficients.
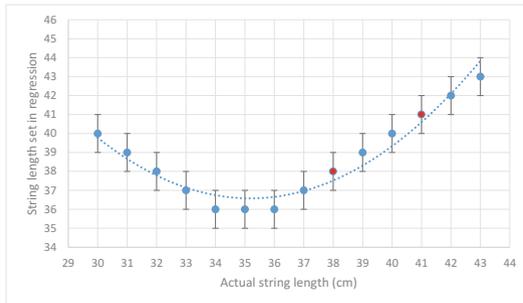

Fig. 5. One-dimensional linear space coefficients found by trial and error.

5   Conclusion and Future Work

In this paper, we proposed a global model for mapping a task parameter to policy parameters. The training examples for constructing the global model were obtained from a single human demonstration and optimized using

reinforcement learning. The trained global model is capable of both inter- and extrapolating policy parameters from new task parameters unseen in the examples. In fact, policy parameters are generalized without re-learning. The global model is simple and can easily be scaled to accommodate for non-linearities in the task space. Experiments showed a significant improvement in extrapolation capabilities over current state-of-the-art. This is due to inherent structures of existing methods which are based either on linear or local regression type relationships between task and policy parameters. Studying the extension of the other available models towards more global regression would open interesting research venues. For example, Gaussian process regression models are capable of representing global relationships, however, the typically used covariance structures (kernels) are local.

Our experiments were limited to a single task parameter and future work should address generalization over more parameters. In addition, we will consider the choice of basis function and model complexity using multiple tasks. Both of these research questions are addressed in model selection methods based on either cross validation or information criteria such as BIC or AIC. Nevertheless, experimental evidence in this paper indicates that the current local and linear models have limited extrapolation powers which needs to be addressed by models able to capture more global relationships despite the number of task parameters.

References

1. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robot. Autonom. Syst. **57**(5), 469–483 (2009)
2. Chernova, S., Veloso, M.: Confidence-based policy learning from demonstration using Gaussian mixture models. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, p. 233. ACM (2007)
3. Sammut, C., Hurst, S., Kedzier, D., Michie, D., et al.: Learning to fly. In: Proceedings of the Ninth International Workshop on Machine Learning, pp. 385–393 (1992)
4. Hovland, G.E., Sikka, P., McCarragher, B.J.: Skill acquisition from human demonstration using a hidden Markov model. In: IEEE International Conference on Robotics and Automation, Proceedings, vol. 3, pp. 2706–2711. IEEE (1996)
5. Calinon, S., D'halluin, F., Sauser, E.L., Caldwell, D.G., Billard, A.G.: Learning and reproduction of gestures by imitation. IEEE Robot. Autom. Mag. **17**(2), 44–54 (2010)
6. Carrera, A., Palomeras, N., Hurtós, N., Kormushev, P., Carreras, M.: Learning multiple strategies to perform a valve turning with underwater currents using an I-AUV. In: OCEANS 2015-Genova, pp. 1–8. IEEE (2015)
7. Da Silva, B., Konidaris, G., Barto, A.: Learning parameterized skills. arXiv preprint arXiv:1206.6398 (2012)
8. Forte, D., Gams, A., Morimoto, J., Ude, A.: On-line motion synthesis and adaptation using a trajectory database. Robot. Autonom. Syst. **60**(10), 1327–1339 (2012)
9. Ude, A., Gams, A., Asfour, T., Morimoto, J.: Task-specific generalization of discrete and periodic dynamic movement primitives. IEEE Trans. Robot. **26**(5), 800–815 (2010)

10. Matsubara, T., Hyon, S.-H., Morimoto, J.: Learning parametric dynamic movement primitives from multiple demonstrations. Neural Netw. **24**(5), 493–500 (2011)
11. Calinon, S., Alizadeh, T., Caldwell, D.G.: On improving the extrapolation capability of task-parameterized movement models. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 610–616. IEEE (2013)
12. Kober, J., Peters, J.R.: Policy search for motor primitives in robotics. In: Advances in Neural Information Processing Systems, pp. 849–856 (2009)
13. Kober, J., Oztop, E., Peters, J.: Reinforcement learning to adjust robot movements to new situations. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence, vol. 22, p. 2650 (2011)
14. Stulp, F., Raiola, G., Hoarau, A., Ivaldi, S., Sigaud, O.: Learning compact parameterized skills with a single regression. In: 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 417–422. IEEE (2013)
15. Nemec, B., Vuga, R., Ude, A.: Efficient sensorimotor learning from multiple demonstrations. Adv. Robot. **27**(13), 1023–1031 (2013)
16. Mülling, K., Kober, J., Kroemer, O., Peters, J.: Learning to select and generalize striking movements in robot table tennis. Int. J. Robot. Res. **32**(3), 263–279 (2013)
17. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: IEEE International Conference on Robotics and Automation, Proceedings, ICRA 2002, vol. 2, pp. 1398–1403. IEEE (2002)
18. Peters, J.R.: Machine learning of motor skills for robotics. Ph.D. thesis, University of Southern California (2007)
19. Kalakrishnan, M., Righetti, L., Pastor, P., Schaal, S.: Learning force control policies for compliant manipulation. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4639–4644. IEEE (2011)
20. Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., Schaal, S.: STOMP: stochastic trajectory optimization for motion planning. In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 4569–4574. IEEE (2011)

Springer