Learning Replanning Policies With Direct Policy Search

Florian Brandherm, Jan Peters, Gerhard Neumann, and Riad Akrour

***Abstract*—Direct policy search has been successful in learning challenging real-world robotic motor skills by learning open-loop movement primitives with high sample efficiency. These primitives can be generalized to different contexts with varying initial configurations and goals. Current state-of-the-art contextual policy search algorithms can however not adapt to changing, noisy context measurements. Yet, these are common characteristics of real-world robotic tasks. Planning a trajectory ahead based on an inaccurate context that may change during the motion often results in poor accuracy, especially with highly dynamical tasks. To adapt to updated contexts, it is sensible to learn trajectory replanning strategies. We propose a framework to learn trajectory replanning policies via contextual policy search and demonstrate that they are safe for the robot, can be learned efficiently, and outperform non-replanning policies for problems with partially observable or perturbed context.**

***Index Terms*—Learning and adaptive systems, reactive and sensor-based planning, motion control, policy search, replaning.**

I. Introduction

A POPULAR approach to learn robotic skills is the combination of direct policy search with structured trajectory representations that can be easily initialized from demonstrations via imitation learning [1], [2]. Successful applications include the Ball-in-a-cup game [3], pancake flipping [4], dart-throwing [5] and table tennis [6]. To generalize over task variations, robotic tasks are often phrased as multitask problems where the motion trajectory is determined by the context. Such a context may include features of the initial and goal configuration. For example, in robot table tennis, the context could include the ball position at the start of the motion and a goal position on the opponent's side of the table. However, determining an entire trajectory from just the initial conditions can be difficult in real-world applications as they might only be partially observable or subject to later perturbations. In real world table tennis, the context is typically affected by noise, unobservable states (e.g., ball spin) and the ball trajectory might be perturbed by external forces. These issues require the online adaption of trajectories to such changes in context.

Formulating the problem of learning robot skills in terms of reinforcement learning and modeling it as a Markov decision process [7], [8] or partially observable Markov decision process [9] is a solution to online adaption. Value-based methods can yield a control policy that continuously produces motor actions from the current state of the task. However, it has become apparent that the resulting model complexity is often challenging for real-world tasks as the sample-complexity is often too large for real experiments to be feasible. Moreover, safety concerns may arise in case of discrete context changes due to updated measurements.

One approach to robot reinforcement learning is to learn the Q-function of a time-dependent control policy [10], [11]. Nevertheless, such a trajectory representation can be problematic. Although it is capable of instant context adaption, discrete jumps in context can result in dangerous movements [12]. This is an important issue since in reality, the frequency of context measurement can be orders of magnitude slower that the robot control frequency (e.g., optical tracking). Furthermore, such a time-dependent policy typically requires a large number of parameters to produce complex trajectories.

An alternative approach that requires to learn much fewer parameters is to learn a simple policy that determines the parameters of a low-dimensional structured trajectory representation [3]. For this purpose, a variety of direct policy search algorithms have been suggested [13]. Many policy search algorithms are based on gradient ascent [10], [14], [15]. An example for an information-theoretic approach is relative entropy policy search (REPS) [16]. Model-based extensions, such as GPREPS [17] (Gaussian process reward model) or Model-based relative entropy stochastic search (MORE) [18] (quadratic reward model) were introduced in order to improve data efficiency. For this paper we chose the contextual variant of MORE, but the principles could be applied to other algorithms as well.

Manuscript received September 10, 2018; accepted February 6, 2019. Date of publication February 25, 2019; date of current version March 11, 2019. This letter was recommended for publication by Associate Editor T. Inamura and Editor D. Lee upon evaluation of the reviewers' comments. This work was supported by the European Union's Horizon 2020 research and innovation program under Grant 640554 (SKILLS4ROBOTS). *(Corresponding author: Florian Brandherm.)*

F. Brandherm is with the Telecooperation Lab, Technical University Darmstadt, 64289 Darmstadt, Germany (e-mail: brandherm@tk.tu-darmstadt.de).

J. Peters is with the Intelligent Autonomous Systems Lab, Technical University Darmstadt, 64289 Darmstadt, Germany (e-mail: jan@robot-learning.de).

G. Neumann is with the School of Computer Science, University of Lincoln, Lincoln LN6 7TS, U.K. (e-mail: geri@robot-learning.de).

R. Akrour is with the Computational Learning for Autonomous Systems Lab, Technical University Darmstadt, 64289 Darmstadt, Germany (e-mail: riad@robot-learning.de).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The material consists of a video that shows the progression of learning a ball-tracking skill. The first part shows the initial policy, which explores around the demonstrated trajectory. There is no correlation between the ball trajectory and the robot. The second part shows a policy that was trained in simulation.

Digital Object Identifier 10.1109/LRA.2019.2901656

The main contributions of this article are as follows:

1) We propose a policy search algorithm that leverages the sample efficiency of contextual policy search (Sec. III) while incorporating a replanning step that takes into account new context information (Sec. IV), akin to step-based RL algorithms.
2) We provide a safety constraint to bound the maximum change in the generated trajectories.
3) The ability of our proposed approach to handle non-stationary and noisy context measurements is demonstrated and analyzed on three tasks in simulation and on a real robot ball tracking task (Sec. VII). This demonstrates that contextual direct policy search algorithms, when coupled with smooth parameterized trajectory generators are more powerful than they are given credit for in the literature.

II. Related Work

One approach to improve performance under uncertain or changing contexts is to delay the planning of a trajectory until enough information is collected or the motion has to be started because of timing constraints [2], However, this fails in cases where the context measurements are not accurate enough at the time a trajectory's execution must be started because of timing constraints.

Defining trajectories rigidly (e.g., based on splines [19]) has the disadvantage that trajectories are precomputed and cannot react online to perturbations. While advanced control methods like model predictive control [20] can handle perturbations, they are not capable of large modifications to the overall trajectory. Therefore, many state of the art approaches to adapt trajectories online use dynamical trajectory representations that are formed by attractor landscapes [2], [12], [21]–[25]. What all these approaches have in common is the utilization of dynamical systems to generate smooth trajectories with a low-dimensional parameterization.

Dynamical movement primitives (DMPs) [21], [22] are widely used for robotic tasks in conjunction with direct policy search [3], [6], [18], [26]. Our method also builds on DMPs (see Sec. V). Ude, Gams, Asfour, *et al.* [27] demonstrated how DMPs can be adapted online by continuously updating the goal position of a motion using a vision system. An alternative approach that was presented by Kober, Mohler, and Peters [23] is the introduction of a term that directly couples a DMP to an external state, demonstrating effective online adaptation to perturbed states. Similarly, Pastor, Righetti, Kalakrishnan, *et al.* demonstrated how DMPs can be used to adapt a grasping motion online by adding a feedback term in order to match prerecorded forces [24]. Khansari-Zadeh and Billard introduced autonomous dynamic systems [25] which is an alternative formulation of dynamical trajectories that models the dynamical system with Gaussian mixture models and has the ability to adapt to perturbations instantly.

State of the art contextual policy search typically assumes that the context is always completely specified ahead of the motion execution [13], allowing to pre-generate a trajectory. The present paper relaxes this assumption. We address both issues of perturbations in the state and uncertain state measurements by replanning trajectories after their execution has started. Replanning steps are introduced at which the trajectory parameters are updated in order to adapt to changes in the context.

We propose a combination of replanning policies with replannable DMPs and show how they can be learned with policy search by introducing some light assumptions. This is followed by an experimental evaluation on tasks that are difficult without rapid adaptation.

III. Preliminaries

The goal of episodic contextual policy search is to learn a contextual stochastic policy $\pi(\boldsymbol{\theta}|\boldsymbol{c})$ directly while treating the problem as a black box. The output parameters of the policy $\boldsymbol{\theta}$ could for example be the parameters of a dynamical movement primitive [21]. The parameters are sampled dependent on a context $\boldsymbol{c}$ which defines the configuration of the task at hand. Such a context can contain features of an observed system state or the definition of a desired goal state. The distribution of this context is considered unknown. The policy $\pi(\boldsymbol{\theta}|\boldsymbol{c})$ is learned by maximizing the expected reward $J(\pi) = \mathbb{E}_{\boldsymbol{\theta}\sim\pi(\boldsymbol{\theta}|\boldsymbol{c})}[\mathcal{R}(\boldsymbol{\theta}, \boldsymbol{c})]$. The reward function $\mathcal{R}(\boldsymbol{\theta}, \boldsymbol{c}) \mapsto \mathbb{R}$ maps parameters, chosen by the policy given the context to a real number that represents the quality of that choice. It is defined carefully as a goal definition of the problem at hand. In other words, if the parameters define a trajectory, $\mathcal{R}(\boldsymbol{\theta}, \boldsymbol{c})$ scores the quality of the generated trajectory.

Our choice of contextual policy search algorithm is Model-Based relative Entropy Stochastic Search (MORE) [18]. MORE samples a fixed number of parameters from the current policy $\pi^i(\boldsymbol{\theta}|\boldsymbol{c})$, executes them on the task and receives the resulting rewards $\mathcal{R}(\boldsymbol{\theta}, \boldsymbol{c})$. This set of parameter samples and their rewards is then used to improve the expected reward of the policy distribution, shifting it towards more successful parameters samples. The updated, improved policy $\pi^{i+1}(\boldsymbol{\theta}|\boldsymbol{c})$ is obtained by solving the optimization problem given by

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & \iint \mu^i(\boldsymbol{c})\pi(\boldsymbol{\theta}|\boldsymbol{c})\mathcal{R}(\boldsymbol{\theta}, \boldsymbol{c})\mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{c}, \\ \text{subject to} \quad & \mathbb{E}_{\boldsymbol{c}\sim\mu(\boldsymbol{c})}\left[\mathrm{KL}\pi(\boldsymbol{\theta}|\boldsymbol{c})\pi^i(\boldsymbol{\theta}|\boldsymbol{c})\right] \leq \epsilon, \\ & \mathbb{E}_{\boldsymbol{c}\sim\mu(\boldsymbol{c})}\left[\mathcal{H}\left(\pi(\boldsymbol{\theta}|\boldsymbol{c})\right)\right] \geq \beta. \end{aligned}$$

with $\mu^i(\boldsymbol{c})$ being the current empirical state distribution. The optimization determines the policy that maximizes the expected reward. Meanwhile, the Kullback-Leibler (KL) divergence of the policy update is bounded to limit the rate of convergence. The KL-divergence between distributions p and q is given by $\mathrm{KL}pq = \int p(x) \log \frac{p(x)}{q(x)}$. Furthermore, placing a lower bound on the entropy of the policy update limits the reduction of the covariance of the policy, which is needed for exploration. The entropy of a distribution p is given by $\mathcal{H}(p) = -\int p(x) \log p(x)$.

This optimization problem is solved using the method of Lagrange multipliers, yielding a closed solution for the policy update which is given by

$$\pi^{i+1}(\boldsymbol{\theta}|\boldsymbol{c}) \propto \pi^i(\boldsymbol{\theta}|\boldsymbol{c})^{\eta/(\eta+\omega)} \exp\left(\frac{\mathcal{R}(\boldsymbol{\theta}, \boldsymbol{c})}{\eta+\omega}\right), \quad (1)$$

where η and ω are Lagrange multipliers.

In order to enable an analytic solution for the corresponding dual function, a quadratic reward model

$$\mathcal{R}(\boldsymbol{\theta}, \boldsymbol{c}) \approx \begin{pmatrix} \boldsymbol{\theta} \\ \boldsymbol{c} \end{pmatrix}^T \underbrace{\begin{pmatrix} \boldsymbol{R}_{\theta\theta} & \boldsymbol{R}_{\theta c}/2 \\ \boldsymbol{R}_{\theta c}{}^T/2 & \boldsymbol{R}_{cc} \end{pmatrix}}_{\text{symmetric}} \begin{pmatrix} \boldsymbol{\theta} \\ \boldsymbol{c} \end{pmatrix} + \begin{pmatrix} \boldsymbol{\theta} \\ \boldsymbol{c} \end{pmatrix}^T \begin{pmatrix} \mathbf{r}_\theta \\ \mathbf{r}_c \end{pmatrix} + r_0 \qquad (2)$$

is learned from the samples. Akrour, Neumann, Abdulsamad *et al.* [11] showed that with this quadratic model and assuming a linear Gaussian policy $\pi(\boldsymbol{\theta}|\boldsymbol{c}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{K}\boldsymbol{c} + \mathbf{b}, \boldsymbol{\Sigma})$, the policy update simplifies to

$$\pi^{i+1}(\boldsymbol{\theta}|\boldsymbol{c}) = \mathcal{N}(\boldsymbol{\theta}| \underbrace{\boldsymbol{F}\boldsymbol{L}}_{\boldsymbol{K}^{(i+1)}} \boldsymbol{c} + \underbrace{\boldsymbol{F}\mathbf{f}}_{\mathbf{b}^{(i+1)}}, \underbrace{\boldsymbol{F}(\eta+\omega)}_{\boldsymbol{\Sigma}^{i+1}})$$

with $\boldsymbol{F} = (\eta\boldsymbol{\Sigma}^{-1} - 2\boldsymbol{R}_{\theta\theta})^{-1}$, $\boldsymbol{L} = \eta\boldsymbol{\Sigma}^{-1}\boldsymbol{K} + \boldsymbol{R}_{\theta c}$ and $\mathbf{f} = \eta\boldsymbol{\Sigma}^{-1}\mathbf{b} + \mathbf{r}_\theta$.

IV. Replanning Policies

Contextual learning often assumes a Gaussian policy $\pi(\boldsymbol{\theta}|\boldsymbol{c}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{K}\boldsymbol{c} + \mathbf{b}, \boldsymbol{\Sigma})$ that is linear in context features. Commonly, such a policy is used to pre-generate a trajectory. In case there are multiple planning steps $k \in 1 \ldots \rho$ such a policy can be divided into one independent linear Gaussian sub-policy $\pi_k(\boldsymbol{\theta}|\boldsymbol{c}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{K}_k\boldsymbol{c} + \mathbf{b}_k, \boldsymbol{\Sigma}_k)$ for each planning step. These sub-policies can be evaluated successively with the current context at their respective planning step. Because of the simple linear relationship of $\pi_k(\boldsymbol{\theta}|\boldsymbol{c})$, replanning can happen quasi-instantaneously in the robot controller whenever a replanning step is triggered. While the initial planning step must remain at the start of a trajectory, the following *replanning* steps can be spaced arbitrarily along the trajectory's time frame. This allows the adaption of trajectory parameters to a changing context.

If the Markov property holds, the sub-policies can be assumed to be independent because of causality. The Markov property can be achieved by either including the robot state in the context or by including the entire history of contexts and parameters of previous planning steps. However, we make the simplifying assumption that the sub-policies are always independent. Thus, replanning policies can be described by a single multivariate Gaussian distribution

$$\pi(\boldsymbol{\theta}|\boldsymbol{c}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{K}\boldsymbol{c} + \mathbf{b}, \boldsymbol{\Sigma}) \qquad (3)$$

that is linear in context features with stacked vectors for the output parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^T, \boldsymbol{\theta}_2^T, \ldots, \boldsymbol{\theta}_\rho^T)^T$, context $\boldsymbol{c} = (\boldsymbol{c}_1^T, \boldsymbol{c}_2^T, \ldots, \boldsymbol{c}_\rho^T)^T$ and bias $\mathbf{b} = (\mathbf{b}_1^T, \mathbf{b}_2^T, \ldots, \mathbf{b}_\rho^T)^T$ along with block matrices for the gain and covariance

$$\boldsymbol{K} = \begin{pmatrix} \boldsymbol{K}_1 & & \cdots & 0 \\ & \boldsymbol{K}_2 & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & \boldsymbol{K}_\rho \end{pmatrix} \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_1 & & \cdots & 0 \\ & \boldsymbol{\Sigma}_2 & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & \boldsymbol{\Sigma}_\rho \end{pmatrix}.$$

Clearly, the number of parameters of the replanning policy $\pi(\boldsymbol{\theta}|\boldsymbol{c})$ grows linearly with the number of planning steps ρ. Retaining this block shape during the policy update requires further constraints on the reward model in equation (2). It is necessary to assume that $\boldsymbol{R}_{\theta\theta}$, $\boldsymbol{R}_{\theta c}$ and $\boldsymbol{R}_{cc}$ are block-diagonal. With this assumption, the policy update of MORE will trivially preserve the independence of the policies by preserving the block-diagonality of $\boldsymbol{K}^{(i+1)}$ and $\boldsymbol{\Sigma}^{(i+1)}$ which is imperative to preserving the independence of the sub-policies.

In fact, any policy search algorithm with linear gaussian policy can be adapted to replanning if the set of learned parameters and covariances can be restricted to the block-diagonal while fixing the remaining parameters of the gain matrix and covariance matrix at zero.

The most powerful replanning policy model is a set of distinct sub-policies for every possible measurement step. Yet, increasing the number of parameters increases the search space and will slow down learning (more on this tradeoff in Sec. VI-A). We present two approaches to limit the number of parameters. The first approach uses the same sub-policy for all planning steps, allowing the number of replanning steps to be independent of the number of parameters. The second approach uses one sub-policy per planning step but limits the number of replanning steps.

A. Stationary Replanning Policy

If replanning has to occur constantly during a trajectory, the number of parameters for independent sub-policies becomes very large. In this case, a desirable assumption is $\hat{\pi}(\boldsymbol{\theta}|\boldsymbol{c}) := \pi_1(\boldsymbol{\theta}|\boldsymbol{c}) = \pi_2(\boldsymbol{\theta}|\boldsymbol{c}) = \cdots = \pi_\rho(\boldsymbol{\theta}|\boldsymbol{c})$, keeping the same number of parameters as the equivalent non-replanning policy. This means that the same policy is evaluated multiple times during a trajectory and it enables trajectories with a varying number of replanning steps. However, this requires that the context of all planning steps is stationary. This requirement arises from the fact that we are using a linear model for generalization. For example, if the context of a table tennis setup is defined as the current ball position, it violates this condition. In some cases, this limitation might be overcome by using a model of the state progression and defining the context as features of the model (e.g., a prediction at some point in the future). However, this is only feasible if 1) a sufficiently good model of the system state is available, and 2) unpredictable system state perturbations (external or by the robot itself) can be ruled out until after the last replanning step. We denote such a policy with equal sub-policies a *stationary policy*.

The reward model for the policy update can be learned from a set of policy samples using linear ridge-regression, given by

$$\boldsymbol{w} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{Y}\,, \qquad \text{with}$$

$$\boldsymbol{Y} = \left(\mathcal{R}^{(1)}, \mathcal{R}^{(2)}, ..., \mathcal{R}^{(n)}\right)^T$$

$$\boldsymbol{\Phi} = \left(\phi(\boldsymbol{\theta}^{(1)}, \boldsymbol{c}^{(1)}), \phi(\boldsymbol{\theta}^{(2)}, \boldsymbol{c}^{(2)}), ..., \phi(\boldsymbol{\theta}^{(n)}, \boldsymbol{c}^{(n)})\right)^T$$

where n is the number of samples, $\mathcal{R}^{(i)}$, $\boldsymbol{c}^{(i)}$ and $\boldsymbol{\theta}^{(i)}$ are the reward, context and corresponding parameters of sample i respectively and λ is a regularization constant. The regression solves for the vector $\boldsymbol{w}$ that contains all parameters of the reward model in equation (2). $\phi(\boldsymbol{\theta}, \boldsymbol{c})$ are the features of the policy sample $(\boldsymbol{\theta}, \boldsymbol{c})$.

For our stationary policy, solving the reward model for all model parameters yields the parameter vector

$$\boldsymbol{w} = (r_0, \boldsymbol{w}_{\text{linear}}, \boldsymbol{w}_{\text{quadratic}})^T$$

with the parameters for the linear part $\boldsymbol{w}_{\text{linear}} = (\hat{\mathbf{r}_\theta}^T, \hat{\mathbf{r}_c}^T)^T$ and the parameters for the quadratic part $\boldsymbol{w}_{\text{quadratic}} = (\text{vech}(\hat{\boldsymbol{R}_{\theta\theta}})^T, \text{vec}(\hat{\boldsymbol{R}_{\theta c}})^T, \text{vech}(\hat{\boldsymbol{R}_{cc}})^T)^T$. $\text{vec}(\cdot)$ is the vectorization function, reordering all elements of a matrix into a column vector. Due to the symmetry of $\boldsymbol{R}_{\theta\theta}$ and $\boldsymbol{R}_{cc}$, we also utilize the half-vectorization function $\text{vech}(\cdot)$, which reorders all elements of a lower triangular matrix into a column vector.

The corresponding features for the regression are defined as

$$\phi(\boldsymbol{\theta}, \boldsymbol{c}) = (1, \phi_{\text{linear}}, \phi_{\text{quadratic}})^T$$

with linear features $\phi_{\text{linear}} = (\sum_{i=1}^{\rho} \boldsymbol{\theta}_i^T, \ \sum_{i=1}^{\rho} \boldsymbol{c}_i^T)^T$ and quadratic features $\phi_{\text{quadratic}} = (\text{vech}(\sum_{i=1}^{\rho} \boldsymbol{\theta}_i \cdot \boldsymbol{\theta}_i^T)^T, \text{vec}(\sum_{i=1}^{\rho} \boldsymbol{\theta}_i \cdot \boldsymbol{c}_i^T)^T, \text{vech}(\sum_{i=1}^{\rho} \boldsymbol{c}_i \cdot \boldsymbol{c}_i^T)^T)^T$.

B. Non-Stationary Replanning Policy

The second, more general variant of replanning policy can utilize differently distributed contexts for different planning steps, exploiting the independence of the sub-policies. This implies that even the dimensionality of the context can differ between planning steps. Because of its capability to handle non-stationary contexts, we denote this policy variant as *non-stationary*. An example for such a task is robot table tennis, where the ball state distribution is different at different points in time due to the progressing motion of the ball. As a result of non-stationarity, the number of parameters for the regression increases significantly. The vector of linear parameters for this case is given by $\boldsymbol{w}_{\text{linear}} = (\mathbf{r}_{\theta 1}^T, \ldots, \mathbf{r}_{\theta \rho}^T, \mathbf{r}_{c1}^T, \ldots, \mathbf{r}_{c\rho}^T)^T$. Likewise, the vector of quadratic parameters $\boldsymbol{w}_{\text{quadratic}} = (\text{vech}(\boldsymbol{R}_{\theta\theta 1})^T, \ldots, \text{vech}(\boldsymbol{R}_{\theta\theta \rho})^T, \text{vec}(\boldsymbol{R}_{\theta c 1})^T, \ldots, \text{vec}(\boldsymbol{R}_{\theta c \rho})^T, \text{vech}(\boldsymbol{R}_{cc1})^T, \ldots, \text{vech}(\boldsymbol{R}_{cc\rho})^T)^T$ contains all elements of the block-diagonal matrices $\boldsymbol{R}_{\theta\theta}$, $\boldsymbol{R}_{\theta c}$ and $\boldsymbol{R}_{cc}$.

V. Replanning Control

To react to unforeseen state changes after the initial plan for the trajectory, additional replanning steps are introduced. A common approach to robot reinforcement learning is to exploit a structured trajectory representation and train a policy to generalize only a small subset of parameters (e.g., goal position) that adjust a demonstrated trajectory. This allows very low-dimensional policies since all relevant trajectory parameters $\boldsymbol{\theta}$ are determined by a small set of initial conditions $\boldsymbol{c}$.

For our replanning framework, we build on the formulation of dynamical movement primitives (DMP) [21], [22]. The trajectory is produced by the differential equation

$$\ddot{\boldsymbol{x}}(t) = \alpha(\beta(\boldsymbol{g} - \boldsymbol{x}(t)) - \dot{\boldsymbol{x}}(t)) + \boldsymbol{f}(t)\,, \tag{4}$$

where $\boldsymbol{x}$ is the vector of joint positions and α, β are constants. As time progresses, the goal attractor becomes dominant and the trajectory converges at the goal position $\boldsymbol{g}$. This representation as a differential equation enables on-the-fly trajectory generation, which can be exploited to smoothly adapt online to updated parameters.

Replanning can be achieved by updating the goal position $\boldsymbol{g}$ or the weights $\boldsymbol{w}$ of the forcing function $\boldsymbol{f}$ that defines the shape of the motion. At every planning step, the relevant subset of these trajectory parameters is initialized/updated by the (re)planned parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_\rho$ However, in the original formulation, such an update results in noticeable jerky trajectories, because it causes discrete jumps in acceleration. Such trajectories can cause damage to real robots. We mitigate jerky behavior by updating the desired goal position $\boldsymbol{g}_d$ and weights $\boldsymbol{w}_d$ and smoothly transitioning $\boldsymbol{g}$ and $\boldsymbol{w}$ with the differential equations $\dot{\boldsymbol{g}} = c_g(\boldsymbol{g} - \boldsymbol{g}_d)$ and $\dot{\boldsymbol{w}} = c_w(\boldsymbol{w} - \boldsymbol{w}_d)$, where c_g and c_w are constants that bound the jerk that is introduced by parameter updates.

A. Safety of Replanning

We show in this section, that a trajectory that results from an update of the goal position can be guaranteed to be bounded. For this, we assume that the exploratory noise of the policy is small and we can express an updated goal position as $\boldsymbol{g_{i+1}} = \boldsymbol{K}\boldsymbol{c_{i+1}} + \boldsymbol{b}$. The DMP equation for this upated goal position then becomes $\ddot{\boldsymbol{x}}_{i+1} = \alpha(\beta(\boldsymbol{K}\boldsymbol{c_{i+1}} + \boldsymbol{b} - \boldsymbol{x}_{i+1}) - \dot{\boldsymbol{x}}_{\boldsymbol{i+1}}) + \boldsymbol{f}$. This new trajectory can be expressed as $\ddot{\boldsymbol{x}}_{i+1}(t) = \ddot{\boldsymbol{x}}_i(t) + \ddot{\boldsymbol{h}}(t)K\Delta_c$, where $\Delta_c = \boldsymbol{c_{i+1}} - \boldsymbol{c_i}$ is the change in context and $\boldsymbol{h}(t)$ is an unknown function of time. It follows from the definition of the DMP that

$$\ddot{\boldsymbol{x}}_{i+1}(t) = \ddot{\boldsymbol{x}}_i(t) + \underbrace{[\alpha\beta(1 - \boldsymbol{h}(t)) - \beta\dot{\boldsymbol{h}}(t)]}_{\ddot{\boldsymbol{h}}(t)} K\Delta_c.$$

Solving the differential equation

$$\ddot{\boldsymbol{h}}(t) = \alpha\beta(1 - \boldsymbol{h}(t)) - \beta\dot{\boldsymbol{h}}(t),$$

with initial condition $\boldsymbol{h}(0) = 0$ (both plans start from the same position) and with standard setting $\beta = 4\alpha$ we obtain $\boldsymbol{h}(t) = 1 - \exp(-2\alpha t)$. Since $|\boldsymbol{h}(t)| = |1 - \exp(-2\alpha t)| \leq 1$, the maximum joint position deviation $\Delta_{\max} = \max_t |\boldsymbol{x}_{i+1}(t) - \boldsymbol{x}_i(t)|$ is bounded and we obtain $\Delta_{\max} \leq |K\Delta_c|$. In other words, provided that the maximum change Δ_c in context is bounded, we can ensure the safety of the re-planning by bounding a norm on the policy parameters. In practice one can perform a line-search, starting from the η returned by the dual function minimization (see Sec. III) to ensure that $|K| < \xi$, where ξ is some hand defined hyper-parameter of the algorithm. Note that such a value of η always exists since as η goes to infinity, the limit of the updated policy is the old policy ($\text{KL}\pi^{i+1}\pi^i = 0$) and the linear term of the previous policy has norm smaller than ξ.

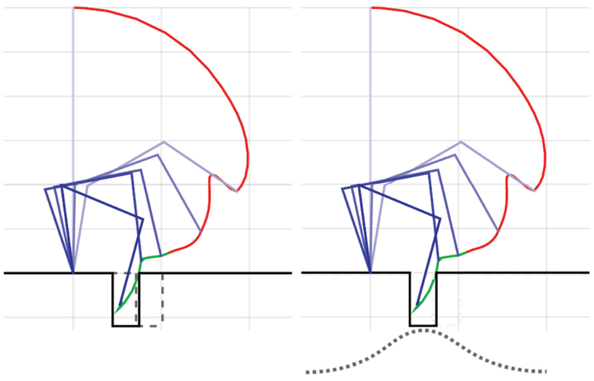

Fig. 1. Hole Reaching Task. Two variants of the Hole Reaching task with moving hole position (left) or hole position with noisy measurement (right). Left: Initial hole position (gray, dashed) and current hole position (black, solid). Right: Noisy measurement of the hole position. The amount of noise is inversely proportional to the distance between end-effector and hole. An end-effector trajectory that was replanned by a non-stationary policy is shown in red before the hole position change and in green after the change.

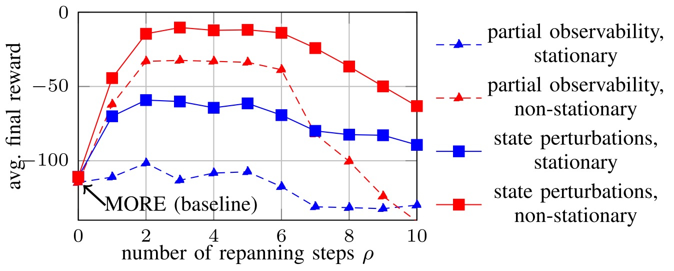

Fig. 2. Comparing Different Numbers of Replanning Steps. Average final rewards after 1000 episodes for different numbers of replanning steps ρ. While the performance initially rises du to increased model power, it eventually degrades because it can't converge within 1000 episodes.

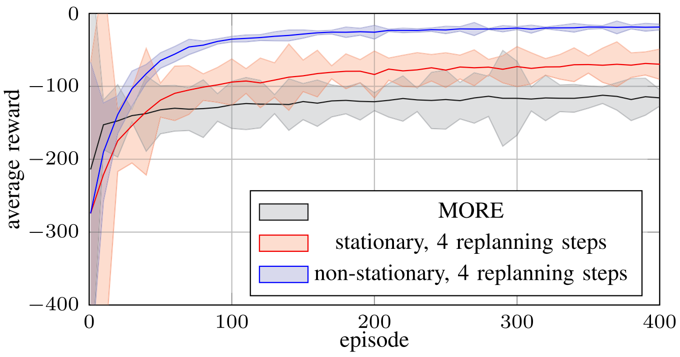

Fig. 3. Hole Reaching Task with State Perturbations. Average rewards and variances per episode of hole-reaching with state perturbations for MORE and replanning.

VI. Simulation Results

To evaluate the replanning policies, we conducted experiments that compare the performance of stationary and non-stationary policies, as well as MORE on two variants of a planar hole reaching task and a simulated hole reaching task.

A. *Planar Hole Reaching With State Perturbations*

Our first experiment is a planar hole reaching task with perturbations in the state. A simulated robot arm is tasked with reaching into a hole in the ground at a randomized distance in front of it. At a random time during the motion, the hole position changes a small random amount. This is an example where replanning is necessary to adapt the motion to a changing context.

As pictured in Figure 1 we simulated a 3-link robot arm. It follows the trajectory of a set of DMPs with 3 basis functions per joint that define the trajectory for each joint angle, starting from an upright position. The task is to move the end-effector into the bottom of the hole while avoiding contact with the floor and walls (black). Therefore, shortly before the end of the simulation, a reward is given proportional to the negative squared distance of the end-effector to the center of the bottom of the hole. Meanwhile, every step of contact with the floor or hole walls is punished. Additionally, the problem is regularized by punishing joint accelerations. After a random time which is drawn from a uniform distribution, the hole position changes position by a small random distance. The context is defined as a set of 3 radial basis function features of the hole position, while the output parameters of the policy are the weights of the DMP forcing functions and the DMP goal positions (in total 12 parameters per independent planning step).

For the experiments, 10 trials of 1000 episodes were evaluated. In each episode, 50 samples were generated while keeping the last 1000 for the policy update.

1) No Replanning (MORE): The black curve in Figure 3 shows the performance of the hole reaching task without replanning. Unable to adapt to a change in the hole position, it fails to reach adequate performance.

2) Replanning With Stationary Policy: We spaced the replanning steps uniformly within the simulation interval for all experiments. The red graph shows that replanning with a stationary policy improves the performance radically, although the number of parameters (12) is the same as for the non-replanning policy. It has 4 replanning steps. The average rewards are affected by a similar level of noise as the non-replanning version. This is because the summation of noisy contexts increases the total noise of the reward model estimation.

3) Replanning With Non-Stationary Policy: The best performance was reached with a non-stationary replanning policy (blue). This boost in performance over the stationary policy can be explained by the more powerful model. The policies for the different replanning steps differ greatly, allowing advanced strategies like waiting until the hole change has occurred before moving down into the hole.

We also analyzed the effect of different numbers of replanning steps. As shown in Fig. 2, more replanning steps ρ are not always better and there exists a task-specific optimum value for ρ. While increasing ρ yields more powerful models, the problem also becomes more difficult. Thus, the available budget for sample generation is an important factor for the choice of ρ. In the non-stationary case, increasing the number of planning steps results in a higher number of parameters that have to be learned, rendering the problem more difficult. Similarly, increasing the number of planning steps for the stationary case leads to

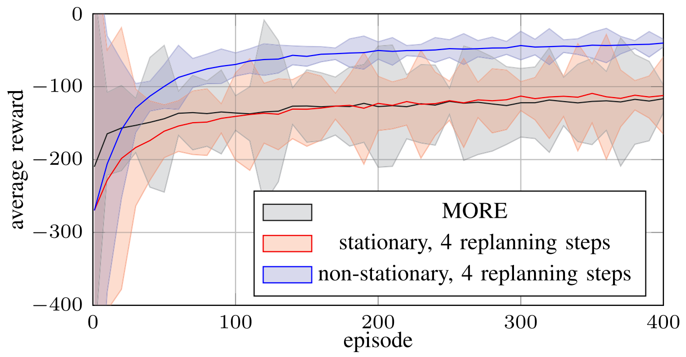

Fig. 4. Hole Reaching Task with Partial Observability. Average rewards and variances per episode of hole-reaching with partial observablity for MORE and replanning.

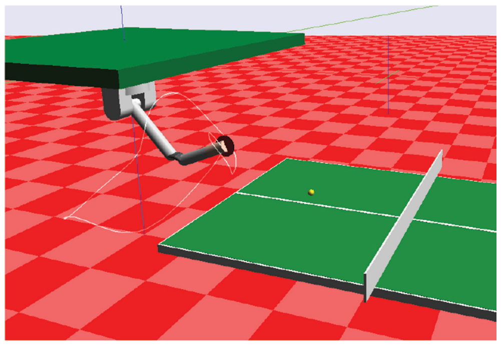

Fig. 5. Table Tennis Task. Depiction of the simulated table tennis setup. A table tennis racket is attached to the end-effector of a ceiling-mounted barret robot.

increased noise in the estimation of the reward model, likewise making the problem more difficult. This demonstrates that the number of planning steps ρ is an important hyper-parameter that needs to be chosen carefully.

A further consequence of this quasi-bound on ρ (and therefore the number of learned parameters) is that the computational cost of learning is bounded as well. It was observed that the cost of sample generation is in practice much higher than the cost of computing policy updates.

B. *Hole Reaching With Partial Observability*

We performed an additional experiment with a partially observable variant of the hole reaching task. Instead of changing the hole position, the hole position remains fixed. However, for this experiment the context features of the hole position are affected by additive uniform noise. The noise level is proportional to the distance of the end-effector to the entry of the hole. The reasoning behind this is to simulate a camera that is mounted to the end-effector.

Again, 10 trials of 1000 episodes were evaluated. In each episode, 50 samples were generated, keeping the last 1000 for the policy update.

1) No Replanning (MORE): The black curve in Figure 4 shows the performance without replanning. The noise level of the initial context is significant. Thus, it is unable to locate the hole reliably enough.

2) Replanning With Stationary Policy: The performance of a replanning policy with a stationary policy is displayed in red. The performance cannot be improved compared to the non-replanning policy. Although the mean of the context feature are the same for each replanning step, variance differs drastically, which violates the stationarity requirement. Policies needs to be aware of the higher accuracy of updated measurements to plan more precisely, requiring a non-stationary policy.

3) Replanning With Non-Stationary Policy: As before, the best performance was achieved with a non-stationary replanning policy (blue). This demonstrates that simple replanning policies can be used to learn problems that suffer from partial observable states with non-trivial noise models.

C. *Simulated Table Tennis*

Another task that we evaluated in this letter is simulated robot table tennis. In our setup, a ceiling-mounted robot arm is tasked to return incoming balls to a point on the other side of the table. This is a difficult problem in the real world because of imperfect ball tracking and tight timing constraints. For this reasons it has been used frequently in reinforcement learning research ([2], [16], [11], [6], [28], [26]). Ball state measurements tend to increase in accuracy the closer it gets to the hitting moment due to filtering. However, the hitting motion must be initiated before a perfect estimate of the ball state is available, leading to decreased performance of trajectories that are planned ahead. These challenges make the table tennis task a good candidate for the application of replanning.

The setup consists of a table tennis table and a ceiling-mounted barret robot arm which has table tennis paddle attached to its end-effector (see Fig. 5). A simulated ball cannon shoots a table tennis ball in a fixed direction such that it bounces once on the robot's side of the table. However, the trajectories of the ball cannon are affected by noise, similar to a real ball cannon, requiring the adaption of each hitting motion to the given ball context. The robot is tasked to perform a forehand stroke that returns the ball to a specified point on the opposite side of the table. It is controlled by a PD-controller that follows a DMP trajectory which is initialized by imitation-learning. The policy has to adapt the DMP goal position to the variations in the ball trajectories. The issue of noisy ball measurements is simulated by adding Gaussian noise to the simulated ball position. This ball measurement is then filtered by an extended Kalman filter to maximize the performance of policy search without replanning, resulting in the black curve in Figure 6. Despite the filtering, the ball state remains noisy and can never be determined exactly. Moreover, the uncertainty of the ball state measurement decreases with time, due to the extended Kalman filtering. Because the robot's acceleration is limited, the trajectory must be initiated before the measurement becomes accurate enough.

We simulated 7 trials table tennis experiments with 1000 episodes for the non-replanning policy and the non-stationary replanning policy. Each episode, 100 samples are generated. For the policy update, the last 500 samples are used.

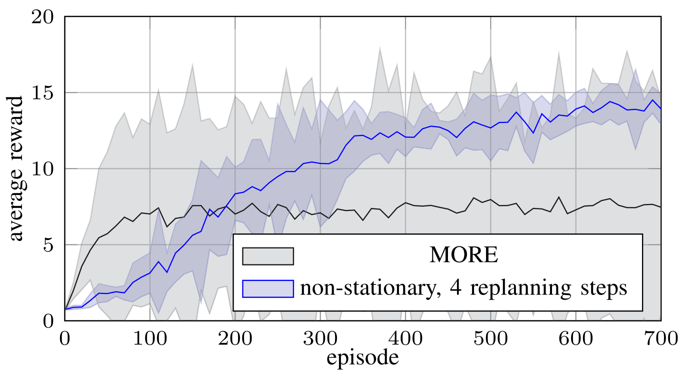

Fig. 6. Table Tennis Learning Curves. Comparison of the average reward of the table tennis task and its variance per episode for non-stationary replanning policies and non-replanning policies.

1) No Replanning (MORE): As shown in Figure 6, learning the table tennis task without replanning only reaches poor performance. The remaining noise in the filtered measurements renders the generated trajectories very inaccurate. Furthermore, the achieved rewards are very noisy.

2) Replanning With a Non-Stationary Policy: Because the context is time-dependent, a stationary policy cannot be applied to this task. To exploit knowledge about the measurement uncertainty, we use the trace of the Kalman filter covariance matrix as additional context. In our experiment, the average performance of the non-stationary policy significantly improves upon MORE. Moreover, the variance of the average rewards is greatly reduced, indicating a reliable policy. This demonstrates that indeed, replanning can robustly improve the performance of a task with partial observability.

VII. Experiment: Real-Robot Ball Tracking

We also evaluated our replanning policies on a real-world ball tracking task. A 7-link WAM robot arm is tasked to reach for flying table-tennis balls with a cup on its end-effector. (Fig. 7). An experimenter throws the ball in a variety of different ball trajectories. Balls are tracked by an optical motion-capture system which offers limited tracking accuracy. As ball velocity estimates are extremely inaccurate, the context only consists of the 3d ball position at the planning step. From this context, a 7-dimensional goal (joint-)position is planned. For this experiment, we used a variant of our method, where replanning sub-policies $\pi(\boldsymbol{\theta}_i|\boldsymbol{c}_i, \boldsymbol{\theta}_{i-1}) = \mathcal{N}(\boldsymbol{\theta}_i|\boldsymbol{K}\boldsymbol{c} + \mathrm{b} + \boldsymbol{\theta}_{i-1}, \boldsymbol{\Sigma})$ provide additive corrections. The reward function is proportional to the negative squared distance between the ball and cup center at the closest point with a regularization term that quadratically punishes joint accelerations. Our experiment shows that our method can be used to learn reaching motions under these challenging circumstances. The DMP of the motion was initialized from a demonstration. In order to reduce the costly real-world training time, a policy was pre-trained on real recorded throws in a simulation before continuing with real experiments. As depicted in Fig. 8, the real experiment achieves similar performance as the simulation, which leads us to the conclusion that our method can be successfully applied in real-world tasks. In this difficult setting, our method learned to generate trajectories that touch the ball in 74% and catch it in 8% of ball trajectories.

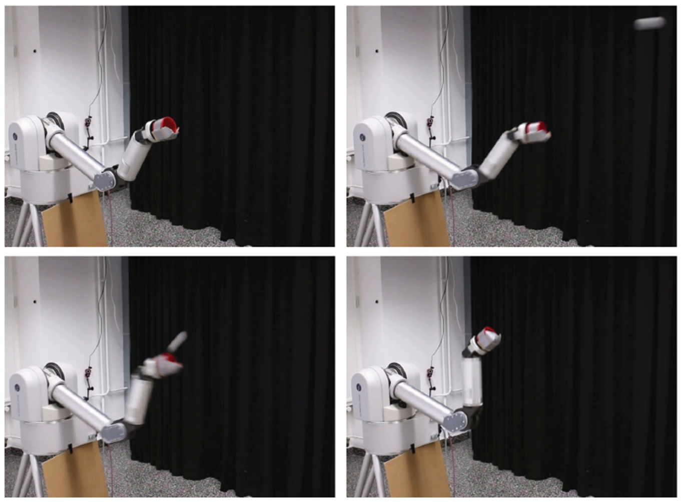

Fig. 7. Ball Tracking Task. The robot has to reach a ball (gray) with a cup on its end-effector (red). The ball is thrown by an experimenter from ca. 3 meters in front of the robot. The time from the first registration of the incoming ball to the moment of contact is around $0.5s$.

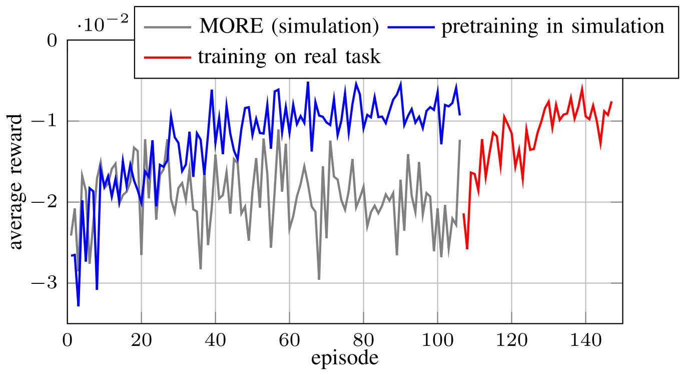

Fig. 8. Ball Tracking Learning Curves. Blue: average reward per episode for the policy pre-training in simulation. Red: average rewards after transferring the policy that was learned in the simulation to the real robot. Gray: MORE is unable to converge to a good policy.

VIII. Conclusion

Real-world robotic tasks often suffer from partial observability or perturbations of their environment state, rendering trajectories that are planned ahead of the motion inadequate. In order to adapt online to changes in the environment state or its measurements, we presented a framework for learning replanning policies with very little modification to standard contextual policy search and discussed under which conditions they can be applied to different scenarios. We then performed experiments on different tasks, comparing the performance of stationary and non-stationary policies with non-replanning policies. Our experiments demonstrate that replanning policies outperform non-replanning policies for tasks with partially observable or changing states due to their ability to adapt online. These results, together with our real experiment, indicate that replanning offers great potential to be applied to real-world robotic tasks, as they are often challenging due to partial observability and perturbations.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[2] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *Int. J. Robot. Res.*, vol. 32, no. 3, pp. 263–279, 2013.

[3] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 849–856.

[4] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with EM-based reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 3232–3237.

[5] B. C. Da Silva, G. Konidaris, and A. G. Barto, "Learning parameterized skills," in *Proc. 29th Int. Conf. Mach. Learn.*, 2012, pp. 1443–1450.

[6] F. End, R. Akrour, J. Peters, and G. Neumann, "Layered direct policy search for learning hierarchical skills," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 6442–6448.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1, no. 1. Cambridge, MA, USA: MIT Press, 1998.

[8] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[9] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Auton. Agents Multi-Agent Syst.*, vol. 27, no. 1, pp. 1–51, Jul. 2013.

[10] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 7–9, pp. 1180–1190, 2008.

[11] R. Akrour, G. Neumann, H. Abdulsamad, and A. Abdolmaleki, "Model-free trajectory optimization for reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2961–2970.

[12] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control—A unifying view," *Prog. Brain Res.*, vol. 165, pp. 425–445, 2007.

[13] M. P. Deisenroth *et al.*, "A survey on policy search for robotics," *Found. Trends Robot.*, vol. 2, no. 1/2, pp. 1–142, 2013.

[14] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 229–256, 1992.

[15] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.

[16] J. Peters, K. Mülling, and Y. Altun, "Relative entropy policy search," in *Proc. AAAI Conf. Artif. Intell.*, 2010, pp. 1607–1612.

[17] A. G. Kupcsik *et al.*, "Data-efficient generalization of robot skills with contextual policy search," in *Proc. 27th Conf. Artif. Intell.*, 2013, pp. 1401–1407.

[18] A. Abdolmaleki, R. Lioutikov, J. Peters, N. Lau, L. P. Reis, and G. Neumann, "Model-based relative entropy stochastic search," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3537–3545.

[19] H. Miyamoto *et al.*, "A Kendama learning robot based on bi-directional theory," *Neural Netw.*, vol. 9, no. 8, pp. 1281–1302, 1996.

[20] D. E. Seborg, D. A. Mellichamp, T. F. Edgar, and F. J. Doyle III, *Process Dynamics and Control*. Hoboken, NJ, USA: Wiley, 2010.

[21] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 2587–2592.

[22] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, vol. 2, pp. 1398–1403.

[23] J. Kober, B. Mohler, and J. Peters, "Learning perceptual coupling for motor primitives," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 834–839.

[24] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 365–371.

[25] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.

[26] A. Gabriel, R. Akrour, J. Peters, and G. Neumann, "Empowered skills," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 6435–6441.

[27] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Trans. Robot.*, vol. 26, no. 5, pp. 800–815, Oct. 2010.

[28] Z. Wang, A. Boularias, K. Mülling, B. Schölkopf, and J. Peters, "Anticipatory action selection for human–robot table tennis," *Artif. Intell.*, vol. 247, pp. 399–414, 2017.