Incremental Bootstrapping of Parameterized Motor Skills

Jeffrey Frederic Queißer[1], René Felix Reinhart[2] and Jochen Jakob Steil[1,3]

***Abstract*— Many motor skills have an intrinsic, low-dimensional parameterization, e.g. reaching through a grid to different targets. Repeated policy search for new parameterizations of such a skill is inefficient, because the structure of the skill variability is not exploited. This issue has been previously addressed by learning mappings from task parameters to policy parameters. In this work, we introduce a bootstrapping technique that establishes such parameterized skills incrementally. The approach combines iterative learning with state-of-the-art black-box policy optimization. We investigate the benefits of incrementally learning parameterized skills for efficient policy retrieval and show that the number of required rollouts can be significantly reduced when optimizing policies for novel tasks. The approach is demonstrated for several parameterized motor tasks including upper-body reaching motion generation for the humanoid robot COMAN.**

I. INTRODUCTION

Advanced robotic systems face non-static environmental conditions which require context-dependent adaptation of motor skills. Approaches that optimize motions for a given task by reinforcement learning, like object manipulation [6] or walking gait exploration [1], deal only with a single instance of a potentially parameterized set of tasks. In many cases, a low-dimensional parameterization that covers the variance of a task exists. For example, consider reaching and grasping under various obstacle positions and object postures [20], [25], throwing of objects at parameterized target positions [3] or playing table tennis using motion primitives that are parameterized with respect to the current ball trajectory [11]. A full optimization for each new task parameterization from a reasonable initialization, which was acquired by e.g. kinesthetic teaching, means that many computations and trials need to be performed before the task can be executed. This impedes immediate task execution and is highly inefficient for executing repetitive tasks under some structured variance.

Recent work addresses this issue by introducing parameterized motor skills that estimate a mapping between the parameterization of a task and corresponding solutions in policy parameter space [3], [11], [16]–[18], [20], [25]. Generation of training data for the training of such parameterized skills requires the collection of optimized policies for a number of task parameterizations. In previous work, each training sample is based on a full optimization for a new task parameterization starting from a fixed initialization [3], [19], or gathered in demonstrations e.g. by kinesthetic teaching [16], [18], [20], [25]. On the one hand, requesting demonstrations from a human teacher for many task parameterizations is not only time-consuming, but also includes the risk of collecting very different solutions to similar tasks due to the redundancy of the problem. Solutions on a smooth manifold are a prerequisite to allow for generalization for unknown tasks. On the other hand, full optimization from a single initial condition requires many rollouts and neglects the already acquired knowledge about the motor skill.

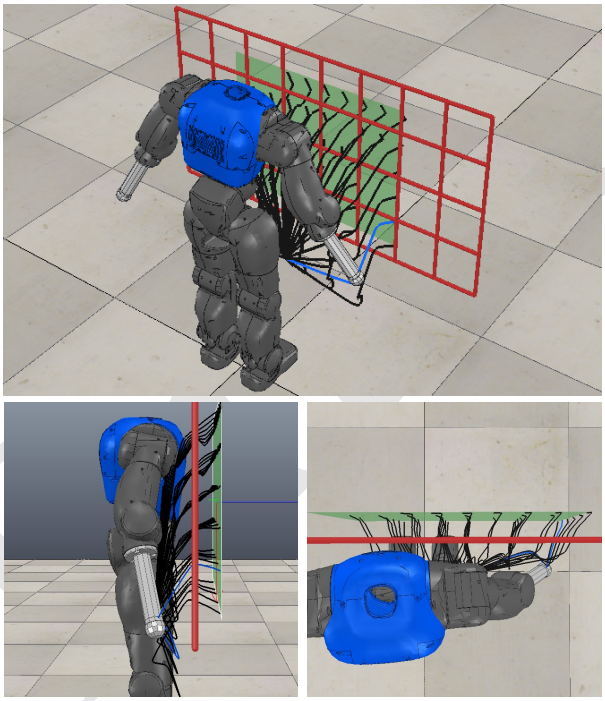


Fig. 1: Constrained reaching scenario with a humanoid upper body and a grid-shaped obstacle. End effector trajectories for different reaching targets retrieved from the learned parameterized skill are shown by black lines.

In this paper, we follow the idea of [3], [19] to apply dedicated policy optimization for new task parameterizations instead of gathering demonstrations from a tutor. We propose an incremental algorithm to establish parameterized skills that reuse previous experience to successively improve the initialization of the optimization process. Thereby we are able to incorporate state-of-the-art optimization of the policy, e.g. by CMA-ES, instead of optimizing meta-

[1]Jeffrey Frederic Queißer & Jochen Jakob Steil are with the Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld University, Universitätsstr. 25, 33615 Bielefeld, Germany jqueisse@cor-lab.uni-bielefeld.de

[2]René Felix Reinhart is with the Fraunhofer IEM Research Institution for Mechatronic Systems Design, Zukunftsmeile 1, 33102 Paderborn, Germany felix.reinhart@iem.fraunhofer.de

[3]Jochen Jakob Steil is with the Institute for Robotics and Process Control, Technical University Braunschweig, Mühlenpfordtstr. 23, 38106 Braunschweig, Germany jsteil@rob.cs.tu-bs.de

parameters of policies [11] and do not rely on library based approaches [17]. In contrast to [3], [19], the optimizer is initialized with the current estimate of the iteratively trained skill and allows for generalization of policy parameters to unknown task parameters. We show that this leads to a significant reduction of the number of required rollouts during skill acquisition. We refer to the process of incremental skill acquisition as *bootstrapping*.

We systematically show that the optimization process benefits from the initial condition proposed by the not yet fully trained parameterized skill and how this benefit depends on the model complexity of the learning algorithm. To cope with redundancy and to support the exploration of smooth manifolds in the policy parameter space, we introduce an additional cost term for optimization that we refer to as *regularization* of policy parameterization. In addition we apply ridge regression with regularization for estimation of a smooth parameterized skill representation. The proposed algorithm for bootstrapping of parameterized skills results in a significant speed-up of the optimization processes for novel task parameterizations. We evaluate these properties of the proposed approach on a via point task with a planar 10 DOF robot arm (see Fig. 5). The scalability of the approach is demonstrated by bootstrapping a parameterized skill for a reaching task incorporating the upper body kinematics of the humanoid robot COMAN (see Fig. 1).

II. BOOTSTRAPPING OF PARAMETERIZED SKILLS

We consider policies $\pi_{\boldsymbol{\theta}}$ that are parameterized by $\boldsymbol{\theta} \in \mathbb{R}^N$. We further assume that tasks are parameterized by $\boldsymbol{\tau} \in \mathbb{R}^M$. Tasks $\boldsymbol{\tau}$ are distributed according to the probability density function $P(\boldsymbol{\tau})$. The task parameterization $\boldsymbol{\tau}$ reflects the variability of the task, e.g. position of obstacles, target positions or load attached to an end effector. With reference to Da Silva et al. [3], we introduce the notion of a parameterized skill, which is given by a function $\Theta : \mathbb{R}^M \rightarrow \mathbb{R}^N$ that maps task parameters $\boldsymbol{\tau}$ to policy parameters $\boldsymbol{\theta}$. The goal is to find a parameterized skill $\Theta(\boldsymbol{\tau})$ that maximizes $\int P(\boldsymbol{\tau}) J(\pi_{\Theta(\boldsymbol{\tau})}, \boldsymbol{\tau}) d\boldsymbol{\tau}$ where $J(\pi, \boldsymbol{\tau}) = E\left\{R(\pi_{\boldsymbol{\theta}}, \boldsymbol{\tau}) | \pi, \boldsymbol{\tau}\right\}$ is the expected reward for using policy $\pi_{\boldsymbol{\theta}}$ to solve a task $\boldsymbol{\tau}$. The reward function $R(\pi_{\boldsymbol{\theta}}, \boldsymbol{\tau})$ assesses each action of the robot defined by the policy $\pi_{\boldsymbol{\theta}}$ with respect to the current task parameterization $\boldsymbol{\tau}$.

We propose an algorithm to build up a parameterized skill $\Theta(\boldsymbol{\tau})$ by consolidating optimized $\boldsymbol{\theta}$ for given $\boldsymbol{\tau}$. For this purpose, we assume that some sort of policy representation, e.g. a motion primitive model, and policy search algorithm, e.g. REINFORCE [26] or CMA-ES [7], is available. The idea is to incrementally train the parameterized skill $\Theta(\tau)$ with task-policy parameter pairs $(\boldsymbol{\tau}, \boldsymbol{\theta}^*)$, where $\boldsymbol{\theta}^*$ are optimized policy parameters obtained by executing the policy search algorithm for task $\boldsymbol{\tau}$. The key step is that the current estimate $\Theta(\boldsymbol{\tau})$ of policy parameters is used as initial condition for policy optimization of new tasks $\boldsymbol{\tau}$. The central outcome of this procedure is that policy search becomes very efficient due to incrementally better initial conditions of the policy search. Ultimately, $\Theta(\boldsymbol{\tau})$ directly provides optimal policy parameters and no further policy optimization needs to be conducted.

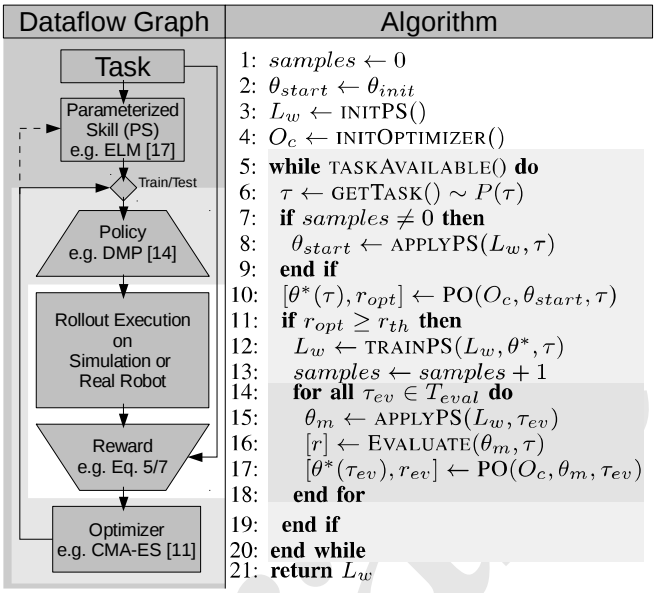


Fig. 2: Dataflow and pseudocode of the proposed bootstrapping algorithm. The parameterized skill (PS) estimates a policy parameterization $\boldsymbol{\theta}_{start}$. In case of training, successive policy optimization (PO) by reinforcement learning results in an update of the parameterized skill. The shading of the background highlights nested processing loops of the system (from outer to inner): (1) Iteration over all tasks; (2) Optimization of $\boldsymbol{\theta}$ by the PO algorithm; (3) Execution and estimation of the reward by iterating over all T timesteps of the trajectory $\boldsymbol{p}_t^*$.

The algorithm for the parameterized skill acquisition is outlined in Fig. 2. For each new task $\boldsymbol{\tau}$, the parameterized skill provides an initial policy parameterization $\boldsymbol{\theta}_{start} = \Theta(\tau)$ (line 8). After collecting a sufficient number of pairs $(\boldsymbol{\tau}, \boldsymbol{\theta}^*)$, the proposed parameterization $\boldsymbol{\theta}_{start}$ can achieve satisfactory rewards such that no further policy optimization (PO) by reinforcement learning is necessary. In case the estimated policy parameters can not yet solve the given task or further training is desired, the optimization from initial condition $\Theta(\boldsymbol{\tau})$ is initiated (line 10). To ensure that only successful optimization results are used for training of the parameterized skill, an evaluation of the optimization process (e.g. reward r_{opt} exceeds a threshold r_{th}) is performed (line 11). If the optimization was successful, the pair $(\boldsymbol{\tau}, \boldsymbol{\theta}^*)$ with optimized policy parameters $\boldsymbol{\theta}^*$ is used for supervised learning of $\Theta(\boldsymbol{\tau})$ (line 12).

Finally, lines 14-18 serve evaluation purposes during incremental training. The evaluation was performed on a predefined set of evaluation tasks in $\boldsymbol{\tau}_{ev} \in T_{ev}$ that are disjunct from the training samples.

In the following, the chosen policy representation and algorithm for policy optimization are briefly introduced.

A. Selection of Policy Representation

The proposed method does not rely on a specific type of policy representation. Many methods for compact policy presentation have been proposed, e.g. based on Gaussian Mixture Models (GMM) [5] or Neural Imprinted Vector Fields [13]. We decided for Dynamic Motion Primitives (DMP, [10]), because they are widely used in the field of motion generation. DMPs for point-to-point motions are based on a dynamical point attractor system

$$\ddot{y} = k_S(g - y) - k_D\dot{y} + f(x, \boldsymbol{\theta}) \tag{1}$$

that defines the output trajectory as well as velocity and acceleration profiles. The canonical system is typically defined as $\dot{x} = -\alpha x$ or in our case as a linear decay $\dot{x} = -\alpha$ as in [12]. The shape of the primitive is defined by

$$f(x, \boldsymbol{\theta}) = \frac{\sum_{k=1}^{K} \exp(-\boldsymbol{V}_k(x - \boldsymbol{C}_k))\boldsymbol{\theta}_k}{\sum_{k=1}^{K} \exp(-\boldsymbol{V}_k(x - \boldsymbol{C}_k))}, \tag{2}$$

where a mixture of K Gaussians is used. $\boldsymbol{C}_k$ are the Gaussian centers and $\boldsymbol{V}_k$ define the variance of the Gaussians. The DMP is parameterized by the mixing coefficients $\boldsymbol{\theta}_k$. We assume fixed variances $\boldsymbol{V}_k$ and a fixed distribution of centers $\boldsymbol{C}_k$ as in [18].

B. Selection of Policy Optimization Algorithm

For optimization of DMP parameters $\boldsymbol{\theta}$ given a task $\boldsymbol{\tau}$, we apply the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES, [7]). Stulp et al. [21] have shown that the black-box optimization by CMA-ES is very efficient and reliable in combination with DMPs. In comparison to other reinforcement learning methods like PI^2 [23] or REINFORCE [26], which evaluate the reward at each time step, CMA-ES is a black-box-optimization algorithm and operates only on the total reward of an action sequence. Stochastic optimization by CMA-ES evaluates N_λ rollouts of policy parameters per generation, which are drawn from a Gaussian distribution centered at the current policy parameter estimate. For each generation the current estimate gets updated by a weighted mean of all N_λ rollouts. The final number of rollouts R required for optimization is given by the number of generations times the number N_λ of rollouts per generation.

C. Selection of Learning Algorithm

For learning of parameterized skills $\Theta(\boldsymbol{\tau})$ we apply an incremental variant of the Extreme Learning Machine (ELM, [8]). ELMs are feedforward neural networks with a single hidden layer:

$$\boldsymbol{\theta}_i(\boldsymbol{\tau}) = \sum_{j=1}^{H} \mathbf{W}_{ij}^{out} \sigma(\sum_{k=1}^{M} \mathbf{W}_{jk}^{inp} \boldsymbol{\tau}_k + \boldsymbol{b}_j) \;\; \forall i = 1, ..., N \tag{3}$$

with input dimensionality M, hidden layer size H and output dimensionality N. Hidden Layer size was set to $H = 50$ for generalization in joint space and $N = 20$ in case of Cartesian end-effector space. Regression is based on a random projection of the input $\mathbf{W}^{inp} \in \mathbb{R}^{H \times M}$, a non-linear transformation $\sigma(x) = (1 + e^{-x})^{-1}$ and a linear output transformation $\mathbf{W}^{out} \in \mathbb{R}^{N \times H}$ that can be updated by incremental least squares algorithms. The incremental update scheme of the ELM was introduced as Online Sequential Extreme Learning Machine (OSELM) [14] that incorporates the ability to perform an additional regularization on the weights [9] or exponential forgetting of previous samples [27]. Since we expect to deal with a small number of training data, regularization of the network can help to prevent over-fitting and foster reasonable extrapolation.

III. EXPERIMENTS

In the following, we demonstrate the applicability of our proposed bootstrapping algorithm. Therefore we designed two scenarios to test the bootstrapping of parameterized skills according to the algorithm from Sec. II.

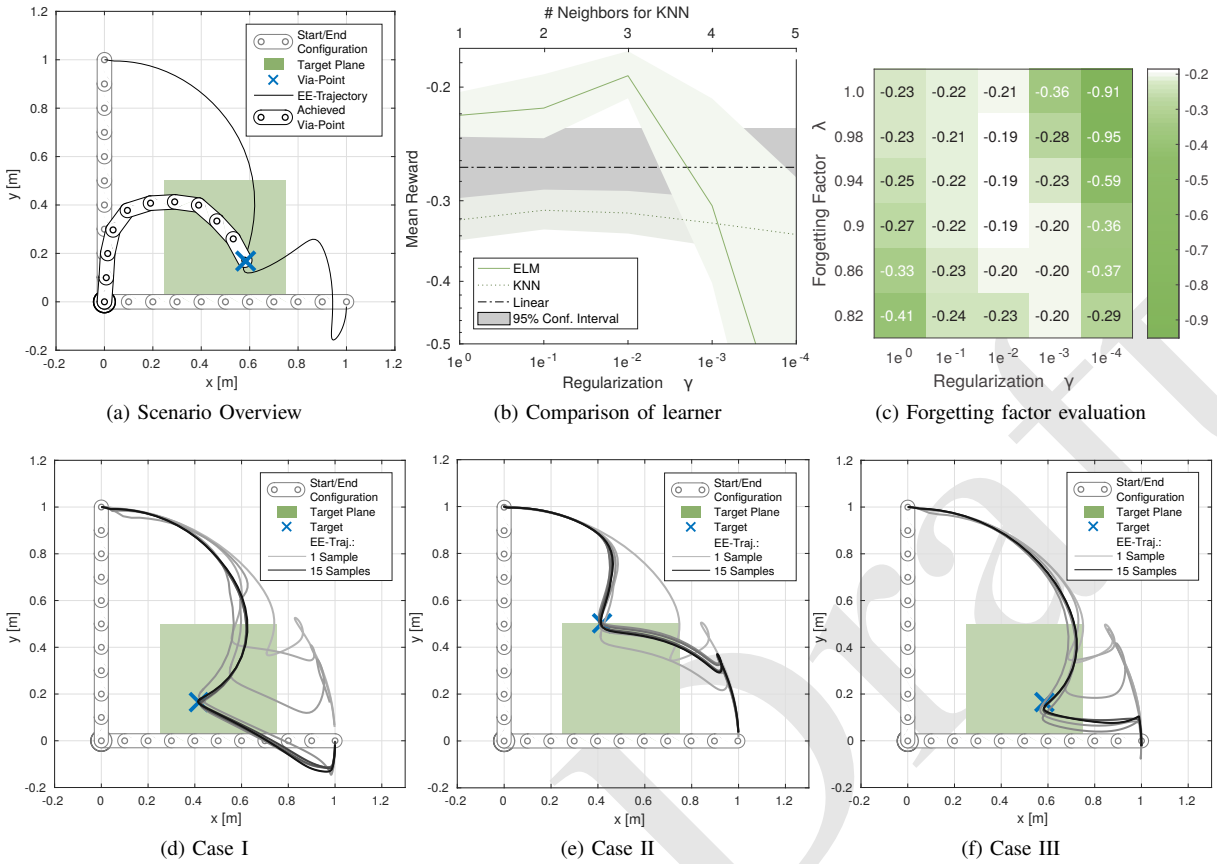
A. 10 DOF Planar Arm Via-Point Task

The goal is to optimize the parameters of a DMP policy to generate joint angle trajectories such that the end-effector of the actuator passes through a via-point in task space at time step $\frac{T}{2}$ of the movement with duration T. For this task we modeled the kinematics of a 10 DOF planar arm. Motions start at initial configuration $\boldsymbol{q}_{start} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^\intercal$ and stop at configuration $\boldsymbol{q}_{end} = (\frac{\pi}{2}, 0, 0, 0, 0, 0, 0, 0, 0, 0)^\intercal$. The task parameterization $\boldsymbol{\tau}$ is given by the 2D via-point position $\boldsymbol{\tau} = (\boldsymbol{v}_x, \boldsymbol{v}_y)$ of the end effector at timestep $\frac{T}{2}$.

Since there exists no unique mapping between task and policy parameter space in this example, infinite action parameterizations can be found that sufficiently solve a given task (e.g. exceed a reward threshold). To reduce ambiguities in the training data for parameterized skill learning, we add a *policy regularization* term to the reward function. This *regularization* punishes the deviation of the policy parameters $\Theta(\tau)$ from the initial parameters $\boldsymbol{\theta}_{init}$ and additionally rewards small jerk of the end effector trajectory. The initial and final arm configurations are shown in Fig. 3a. We utilize a minimum jerk trajectory [22] in joint angle space to generate the initial policy parameters $\boldsymbol{\theta}_{init}$. The overall reward is given by:

$$R(\boldsymbol{\theta}, \boldsymbol{v}) = \underbrace{-\boldsymbol{\alpha}_1 \sum_{t=2}^{T} \left(\frac{\partial^3 \boldsymbol{p}_t^x}{\partial t^3}\right)^2 + \left(\frac{\partial^3 \boldsymbol{p}_t^y}{\partial t^3}\right)^2}_{\text{Jerk (a)}} - \underbrace{\boldsymbol{\alpha}_2 \|\boldsymbol{p}_{T/2} - \boldsymbol{v}_p\|^2}_{\text{Via Point (b)}} - \underbrace{\boldsymbol{\alpha}_3 \|\boldsymbol{\theta}_{init} - \boldsymbol{\theta}\|}_{\text{Regularization (c)}} \tag{4}$$

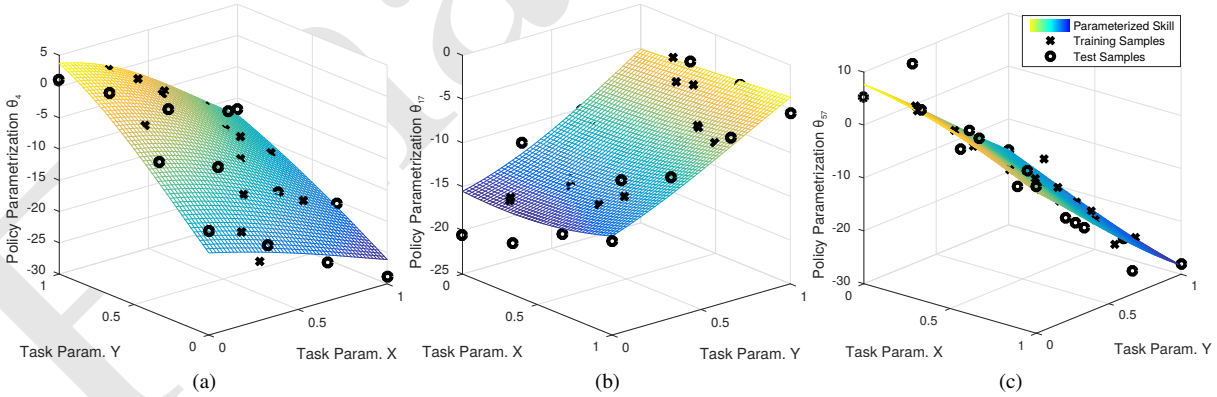
The reward depends on the DMP parameters θ that result in a 10 dimensional joint trajectory transformed by the kinematics of the robot arm to the end-effector trajectory $\boldsymbol{p}_t$. The jerk is based on the third derivative of the end-effector trajectory $\boldsymbol{p}_t$ as proposed in [4] and is represented as one objective in the reward function Eq. 4(a). In addition the reward function refers to the distance to the desired via-point $\boldsymbol{v}$ of the end-effector trajectory Eq. 4(b) and the regularization term Eq. 4(c).



(a) Scenario Overview (b) Comparison of learner (c) Forgetting factor evaluation

(d) Case I (e) Case II (f) Case III

Fig. 3: (a) Experimental setup including start/end configuration as well as an optimized solution for one task. (b) Comparison of the generalization of $\Theta(\boldsymbol{\tau})$ to unseen tasks by linear regression, KNN and ELM depending on the regularization γ. The evaluation shows the mean reward and confidence interval for all test samples $\boldsymbol{\tau}_{ev}$. (c) Forgetting factor evaluation: Mean reward on test samples for $\boldsymbol{\theta}_{start}$ after bootstrapping depending on regularization γ and forgetting factor λ. At the bottom (d)-(f), three exemplary test cases for $\boldsymbol{\tau}$ are shown. They show the content of the learned parameterized skill in relation to the number of training samples. The gray scale indicates the number of consolidated training samples.



(a) (b) (c)

Fig. 4: (a)-(c) show three exemplary dimensions of the parameterized skill $\Theta(\boldsymbol{\tau})$ output in relation to the task parameterization. Task parameterization is the 2D position of the via-point, i.e. $\boldsymbol{\tau} = (\boldsymbol{v}_x, \boldsymbol{v}_y)$.

The coefficients α_i are fixed for all experiments to $\boldsymbol{\alpha} = (10^2, 15, 10^{-3})^\intercal$. The selection of $\boldsymbol{\alpha}$ results in a magnitude of the regularization of ca. 10% of the overall reward of an optimized task. For the training phase we selected $N_{train} = 15$ random tasks τ, i.e. via-point positions, drawn from the green target plane in Fig. 3a. Evaluation was done on a fixed test set $\boldsymbol{\tau}_{ev}$ including $N_{test} = 16$ via-points arranged in a grid on the target plane. For each of the 10 joints of the robot we selected a DMP with $K = 6$ basis functions, resulting in a $M = 60$ dimensional policy parameterization $\boldsymbol{\theta}$. Fig. 3d-

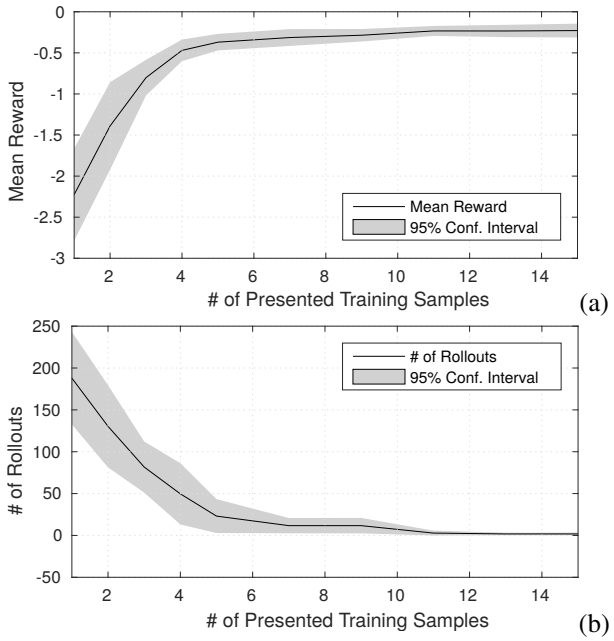


Fig. 5: This figure reveals the mean reward of the initial guess $\boldsymbol{\theta}_{start} = \Theta(\boldsymbol{\tau})$ of the parameterized skill in relation to the number of presented training samples (a) and the mean number of rollouts that are necessary to solve (reward exceeds a threshold) the test tasks (b). Results and confidence interval are based on ten repeated experiments.

3f shows solutions for three exemplary tasks τ from the test set. The gray scale of the end effector trajectories refers to the number of consolidated training samples and shows that the parameterized skill improves as more optimized samples have been used for training. In addition we evaluated the overall performance that can be achieved by the ELM learner in comparison to KNN Regression and Linear Regression as well as the effect of regularization. Those results are shown in Fig. 3b and reveal that the ELM, a non-linear, global learner for $\Theta(\boldsymbol{\tau})$, is able to gain the highest rewards on the test set.

The effect of an exponential forgetting of training data can be seen in Fig. 3c. The forgetting factor is implemented by weighted linear regression of the readout weights of the learner of $\Theta(\boldsymbol{\tau})$. By forgetting earlier training samples ($\lambda < 1$), higher rewards can be achieved after bootstrapping. As the parameterized skill provides a better initialization for the policy search, better solutions can be found since a better initialization reduces the risk of getting stuck in a local minimum. Therefore it is beneficial to forget earlier solutions in favor of new policy search results. In case not all tasks can be solved by policy search due to local minima (as in Sec. III-B), an improved initial guess $\Theta(\boldsymbol{\tau})$ can affect the rate of solvable tasks as well.

Fig. 5(a) shows the mean initial reward for all tasks $\boldsymbol{\tau}_{ev}$ in the test set for the estimated policy parameters $\Theta(\boldsymbol{\tau})$ as function of the number of incorporated training samples. Fig. 5(b) shows that policy optimization benefits from the improved initial policy parameters $\Theta(\boldsymbol{\tau})$ by reducing the number of required rollouts to solve novel tasks (exceed a certain reward threshold). A significant reduction of the required number of rollouts compared to the initialization with the first training sample θ_{init}, e.g. baseline, can be seen.

B. Reaching Through a Grid

In this scenario we optimize a reaching task defined by end effector trajectories to reach certain points on a plane behind a grid-shaped obstacle. The scenario shows the scalability of the purposed approach to more complex tasks. The goal is to reach points on a 2D target plane behind the grid-shaped obstacle without collisions. The experiments are performed in simulation with the humanoid robot COMAN [2] as shown in Fig. 1. We utilize 7 DOF of the upper body including waist, chest and right arm joints. Motions are represented in Cartesian space utilizing 3 DMPs with $K = 5$ basis functions (as in Eq. 2), resulting in a $M = 15$ dimensional optimization problem. The policy parameters are transformed by DMPs to desired Cartesian end effector trajectories $\boldsymbol{p}_t^*$. The kinematics as well as reachability of the robot lead to executable end effector trajectories $\boldsymbol{p}_{r,t}$ in Cartesian space. For each time step t of the desired end effector trajectory $\boldsymbol{p}_t^*$, an inverse Jacobian controller tries to find a configuration of the robot that complies with $\boldsymbol{p}_t^*$ and maximizes the distance to all obstacles in the null-space of the manipulator Jacobian [15]:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}\left(\boldsymbol{p}_t^* - \boldsymbol{p}_{r,t}\right) + \alpha\left(\mathbb{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}\right) Z \tag{5}$$

$$Z = \sum_{l=1}^{L} -\boldsymbol{J}_{p,l}^{\intercal} \cdot \boldsymbol{d}_{min,l} \tag{6}$$

where $\boldsymbol{p}_t^* - \boldsymbol{p}_{r,t}$ is the distance between the desired end effector trajectory $\boldsymbol{p}_t^*$ and the trajectory $\boldsymbol{p}_{r,t}$ reached by the robot. The term Z maximizes the distances $||d_{min,l}||$ of all L links to the grid obstacle in the null-space $\mathbb{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}$. The maximization of the distance to the closest point can be achieved by following the direction $-\boldsymbol{d}_{min,l}$ in joint space by the point Jacobian $\boldsymbol{J}_{p,l}^{\intercal}$ of the closest point to the obstacle. For policy optimization, the reward function is given by

$$R(\theta, \boldsymbol{v}_p) = \underbrace{-\boldsymbol{\alpha}_1 \sum_{t=2}^{T} \|\boldsymbol{p}_t^* - \boldsymbol{p}_{t-1}^*\|}_{\text{Length of Trajecory (a)}} - \underbrace{\boldsymbol{\alpha}_2 \sum_{t=1}^{T} \|\boldsymbol{p}_t^* - \boldsymbol{p}_{r,t}\|}_{\text{Reproducibility (b)}} + \underbrace{\boldsymbol{\alpha}_3 \sum_{t=1}^{T} \boldsymbol{r}_{d,t}}_{\text{Dist. to Obstacles (c)}} - \underbrace{\boldsymbol{\alpha}_4 \|\boldsymbol{\theta}\|}_{\text{Regularization (d)}} \tag{7}$$

where T is the duration of the trajectory. The reward in Eq. 7 is a weighted sum of four parts with weighting factors $\boldsymbol{\alpha}_i$: (1) The length of the desired end effector trajectory $\boldsymbol{p}_{d,t}$ that is defined by policy parameter $\boldsymbol{\theta}$; (2) In addition to the punishment of long trajectories (Eq. 7(a)), the reward takes the reproducibility of the trajectories into account. Therefore, Eq. 7(b) punishes deviations of the reached end effector position $\boldsymbol{p}_{r,t}$ in relation to the desired end effector position $\boldsymbol{p}_t^*$; (3) The distance maximization of all links to the grid

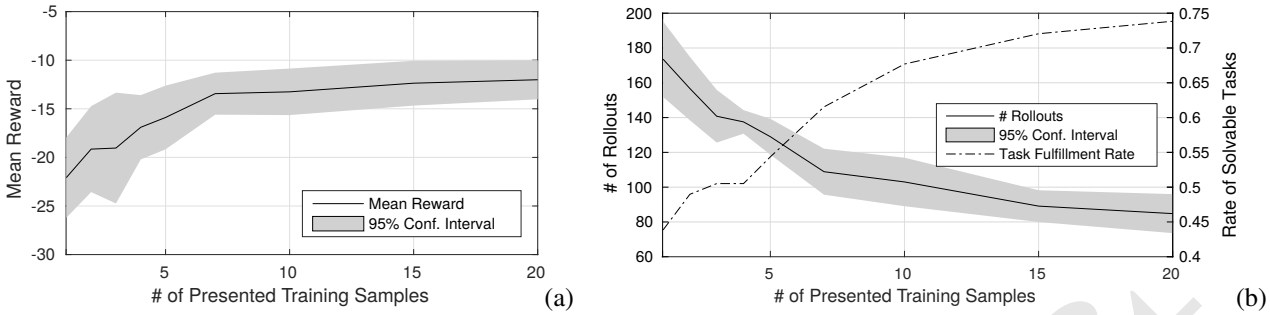


Fig. 6: This figure reveals the mean reward of the initial guess $\boldsymbol{\theta}_{start} = \Theta(\boldsymbol{\tau})$ of the parameterized skill in relation to the number of presented training samples (a) and the mean number of rollouts that are necessary to solve the test tasks (reward exceeds a threshold) (b). The dashed line in (b) shows the mean rate of solvable task in the test set. Results and confidence intervals are based on ten repeated experiments.

obstacle $\boldsymbol{r}_{d,t}$ is considered in Eq. 7(c). The optimization criterion representing the maximization of the distance to the grid-obstacle $\boldsymbol{r}_{d,t}$ is given by:

$$\boldsymbol{r}_{d,t} = -\sum_{l=1}^{L} \min\left(0, \|\boldsymbol{d}_{min,l}\| - \boldsymbol{d}_B\right)^2 \tag{8}$$

It represents a quadratic relationship to the minimum distances $\boldsymbol{d}_{min,l}$ over all L links to all obstacles in the scene in case the distance falls below a given threshold $\boldsymbol{d}_B$. This criterion refers to the the work presented by Toussaint et al. [24] where it was used in the context of null-space constraints for humanoid robot movement generation; (4) An additional normalization for small policy parameterizations as given by Eq. 7(d).

Results: We evaluated the bootstrapping of the parameterized skill as described in Sec. II. For training, we selected $N_{train} = 20$ random target positions on the target plane in front of the robot. For evaluation, we created a fixed regular grid for point sampling of $N_{test} = 39$ positions on the target plane. Fig. 6 reveals that the reward of the initial guess $\boldsymbol{\theta}_{start} = \Theta(\boldsymbol{\tau})$ of the parameterized skill increases with the number of presented training samples. In comparison to the previous experiment in Sec. III-A, the optimization algorithm does not always succeed to find a solution for all tasks of the test set. Fig. 6(b) shows an increasing success rate in relation to the number of consolidated samples and thereby the reward of the initial parameters $\boldsymbol{\theta}_{start}$ of the policy search. This indicates that increasingly better initial conditions $\Theta(\boldsymbol{\tau})$ for policy optimization reduce the risk to get stuck in local minima during optimization. In terms of number of rollouts that are required to fulfill a new task, we observe similar results as in the 10 DOF arm experiment: The number of required rollouts necessary for task fulfillment decreases the more training data were consolidated. This results in a bootstrapping and acceleration of the parameterized skill learning.

IV. CONCLUSION

We introduced a bootstrapping algorithm to incrementally train parameterized skills. Since we have to operate on a small number of training samples, we assumed smoothness of the mappings between task and policy parameter spaces. Our results indicate that the DMP space is well suited for parameterized robot trajectory generation and a smooth mapping between task parameterization and DMP space is a valid assumption. We verified that the incremental learning of parameterized skills is possible and that the incremental update can significantly speed up policy search for novel task parameterizations. Moreover, we showed that initialization of the optimization process with successively improved solutions (i.e. with higher rewards) extends also the number of successfully solved tasks (i.e. exceed a reward threshold). To support consistent training samples without ambiguities caused by the redundancy of the parameterized skill formulation, we introduced additional cost terms in the optimization, which we call *regularization* of the policy parameterization. To investigate the deeper connection between regularization of the parameterized skill learner and the reward function remains future work.

ACKNOWLEDGMENT

J. Queißer received funding from the Cluster of Excellence 277 Cognitive Interaction Technology. F. Reinhart received funding from the European Community's Horizon 2020 robotics program ICT-23-2014 under grant agreement 644727 - CogIMon.

REFERENCES

[1] C. Cai and H. Jiang. Performance Comparisons of Evolutionary Algorithms for Walking Gait Optimization. In *IEEE Intern. Conf. on Information Science and Cloud Computing Companion*, pages 129–134, 2013.

[2] L. Colasanto, N. G. Tsagarakis, and D. G. Caldwell. A Compact Model for the Compliant Humanoid Robot COMAN. In *BioRob*, pages 688 – 694, 2012.

[3] B. C. da Silva, G. Baldassarre, G. Konidaris, and A. Barto. Learning parameterized motor skills on a humanoid robot. In *IEEE Intern. Conf. Robotics and Automation*, pages 5239–5244, 2014.

[4] N. Fligge, J. McIntyre, and P. van der Smagt. Minimum jerk for human catching movements in 3D. In *Proc. IEEE Intern. Conf. on Biomedical Robotics and Biomechatronics*, pages 581–586, 2012.

[5] F. Guenter, M. Hersch, S. Calinon, and A. Billard. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics, Special Issue on Imitative Robots*, 21(13):1521–1544, 2007.

[6] F. Günter. *Using reinforcement learning for optimizing the reproduction of tasks in robot programming by demonstration*. PhD thesis, STI, Lausanne, 2009.

[7] N. Hansen. The CMA evolution strategy: a comparing review. In J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[8] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[9] H. T. Huynh and Y. Won. Online training for single hidden-layer feedforward neural networks using RLS-ELM. In *IEEE Intern. Symp. on Comp. Intelligence in Robotics and Automation*, pages 469–473, 2009.

[10] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013.

[11] J. Kober, A. Wilhelm, E. Oztop, and J. Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, 33:361–379, 2012.

[12] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Wrgtter. Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Trans. Robotics*, 28(1):145–157, 2012.

[13] A. Lemme, K. Neumann, R. Reinhart, and J. Steil. Neural learning of vector fields for encoding stable dynamical systems. *Neurocomputing*, 141:3–14, 2014.

[14] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 17(6):1411–1423, 2006.

[15] A. Liegeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. Systems, Man and Cybernetics*, 7(12):842–868, 1977.

[16] T. Matsubara, S.-H. Hyon, and J. Morimoto. Learning parametric dynamic movement primitives from multiple demonstrations. *Neural Networks*, 24(5):493–500, 2011.

[17] K. Mülling, J. Kober, and J. Peters. Learning table tennis with a mixture of motor primitives. *Proc. of the 10th IEEE-RAS Intern. Conf. on Humanoid Robots (Humanoids 2010)*, pages 411–416, Dec. 2010.

[18] R. F. Reinhart and J. J. Steil. Efficient policy search in low-dimensional embedding spaces by generalizing motion primitives with a parameterized skill memory. *Autonomous Robots*, 38(4):331–348, 2015.

[19] B. D. Silva, G. Konidaris, A. G. Barto, and B. Castro. Learning Parameterized Skills. In *Intern. Conf. on Machine Learning*, pages 1679–1686, 2012.

[20] F. Stulp, G. Raiola, A. Hoarau, S. Ivaldi, and O. Sigaud. Learning compact parameterized skills with a single regression. In *IEEE-RAS Intern. Conf. on Humanoid Robots*, pages 417–422, 2013.

[21] F. Stulp and O. Sigaud. Policy Improvement Methods: Between Black-Box Optimization and Episodic Reinforcement Learning, Oct. 2012.

[22] N. H. Tamar Flash. The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model. *The Journal of Neuroscience*, 5(7):1688–1703, 1984.

[23] E. A. Theodorou, J. Buchli, and S. Schaal. Learning policy improvements with path integrals. In *Intern. Conf. on Artificial Intelligence and Statistics*, pages 828–835, 2010.

[24] M. Toussaint, M. Gienger, and C. Goerick. Optimization of sequential attractor-based movement for compact behaviour generation. In *IEEE-RAS Intern. Conf. on Humanoid Robots*, pages 122–129, 2007.

[25] A. Ude, M. Riley, B. Nemec, A. Kos, T. Asfour, and G. Cheng. Synthesizing goal-directed actions from a library of example movements. In *IEEE-RAS Intern. Conf. on Humanoid Robots*, pages 115–121, 2007.

[26] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, May 1992.

[27] J. Zhao, Z. Wang, and D. S. Park. Online sequential extreme learning machine with forgetting mechanism. *Neurocomputing*, 87:79–89, 2012.