Bootstrapping of Parameterized Skills Through Hybrid Optimization in Task and Policy Spaces

Jeffrey F. Queißer[1]* *and Jochen J. Steil*[2]

[1] *Research Institute for Cognition and Robotics (CoR-Lab), Machine Learning Group, CITEC, Bielefeld University, Bielefeld, Germany,* [2] *Institute for Robotics and Process Control, Technische Universität Braunschweig, Braunschweig, Germany*

Modern robotic applications create high demands on adaptation of actions with respect to variance in a given task. Reinforcement learning is able to optimize for these changing conditions, but relearning from scratch is hardly feasible due to the high number of required rollouts. We propose a parameterized skill that generalizes to new actions for changing task parameters, which is encoded as a meta-learner that provides parameters for task-specific dynamic motion primitives. Our work shows that utilizing parameterized skills for initialization of the optimization process leads to a more effective incremental task learning. In addition, we introduce a hybrid optimization method that combines a fast coarse optimization on a manifold of policy parameters with a fine grained parameter search in the unrestricted space of actions. The proposed algorithm reduces the number of required rollouts for adaptation to new task conditions. Application in illustrative toy scenarios, for a 10-DOF planar arm, and a humanoid robot point reaching task validate the approach.

Keywords: reinforcement learning, policy optimization, memory, learning, hybrid optimization, dimensionality reduction, parameterized skills

OPEN ACCESS

Edited by:
Alexandre Bernardino,
Universidade de Lisboa, Portugal

Reviewed by:
Erol Sahin,
Middle East Technical University, Turkey
John Nassour,
Technische Universität Chemnitz, Germany

****Correspondence:***
Jeffrey F. Queißer
jqueisse@cor-lab.uni-bielefeld.de

Specialty section:
This article was submitted to Humanoid Robotics, a section of the journal Frontiers in Robotics and AI

Received: *13 June 2017*
Accepted: *11 April 2018*
Published: *08 June 2018*

Citation:
Queißer JF and Steil JJ (2018) Bootstrapping of Parameterized Skills Through Hybrid Optimization in Task and Policy Spaces. Front. Robot. AI 5:49. doi: 10.3389/frobt.2018.00049

1. 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 (Günter, 2009) or walking gait exploration (Cai and Jiang, 2013), 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 (Ude et al., 2007; Stulp et al., 2013), throwing of objects at parameterized target positions (Silva et al., 2014) or playing table tennis using motion primitives that are parameterized with respect to the current ball trajectory (Kober et al., 2012). 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 variation.

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 (Ude et al., 2007; Mülling et al., 2010; Matsubara et al., 2011; Kober et al., 2012; Baranes et al., 2013; Stulp et al., 2013; Silva et al., 2014; Reinhart and Steil, 2015). Generation of training data for the update 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 (Silva et al., 2012, 2014), or gathered in demonstrations e.g., by kinesthetic teaching (Ude et al., 2007; Matsubara et al., 2011; Stulp et al., 2013; Reinhart and Steil, 2015). 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 by machine learning algorithms. On the other hand, full optimization from a single initial condition requires many rollouts and ignores the already acquired knowledge about the motor skill.

In this work, we follow the idea of (Silva et al., 2012, 2014; Baranes et al., 2013) to apply dedicated policy optimization for new task parameterizations instead of gathering demonstrations from a tutor. In a similar way as (Baranes et al., 2013), we generalize for new task parameterizations to transfer optimization results. We investigate an incremental algorithm to establish parameterized skills that reuse previous experience to successively improve the initialization of the optimization process (Queißer et al., 2016). Thereby we are able to incorporate state-of-the-art optimization of the policy, i.e., by CMA-ES, instead of optimizing meta-parameters of policies (Kober et al., 2012) and do not rely on library based approaches (Mülling et al., 2010). In contrast to (Silva et al., 2012, 2014), the optimizer is initialized with the current estimate of the iteratively updated parameterized skill. The parameterized skill is a meta learner for generalization of DMP parameterizations for given task parameterizations. 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.

Based on previous experiments (Queißer et al., 2016), we argue for a utilization of the parameterized skill as a projection of the low dimensional manifold of task-space to the high dimensional search space of policy parameters. By performing a policy optimization in this low dimensional manifold, a further speed-up, in terms of number of rollouts, during the optimization process can be observed. But to cope with the very likely case that no sufficient solution for the required task can be found in the manifold of the parameterized skill, we propose to perform a hybrid search in both spaces. Therefore we introduce an hybrid optimization algorithm that samples rollouts in both spaces and performs an estimation for a combined parameter update, as outlined in **Figure 1**.

This work extends our previous method (Queißer et al., 2016) and its contribution aims at the experimental verification of the following hypotheses:

(H1) Initialization of the optimizer with the current estimate of the parameterized skill leads to a faster optimization and convergence of the skill learning. (Sec. 3)

(H2) Searching in the manifold of previous solutions leads to a reduction of search space and thereby to a more efficient acquisition of the parameterized skill. (Sec. 4)

We evaluate the bootstrapping and the hybrid search of the proposed algorithm on a via point task with a planar 10-DOF robot arm (see **Figure 2**). Additionally we investigate the properties of the proposed optimization in hybrid spaces on toy examples. 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 **Figure 3**) in end-effector as well as joint space control.

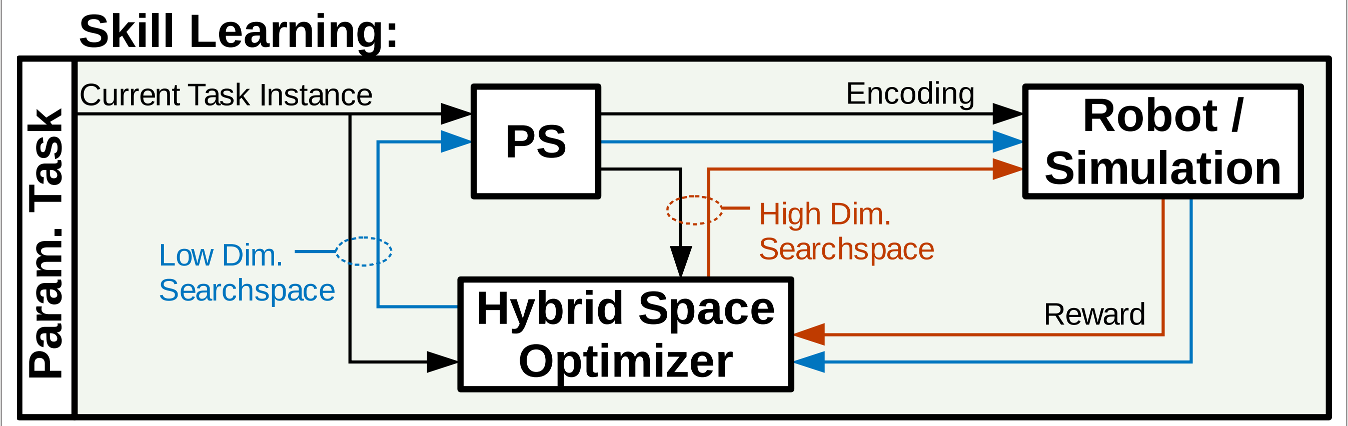


FIGURE 1 | Hybrid optimization framework, the optimizer is initialized (**H1**) by the current estimate (gray) of the parameterized skill PS and performs a hybrid optimization (**H2**) in the low dimensional manifold of previous solutions (blue) and the high dimensional space of motion primitives (red).

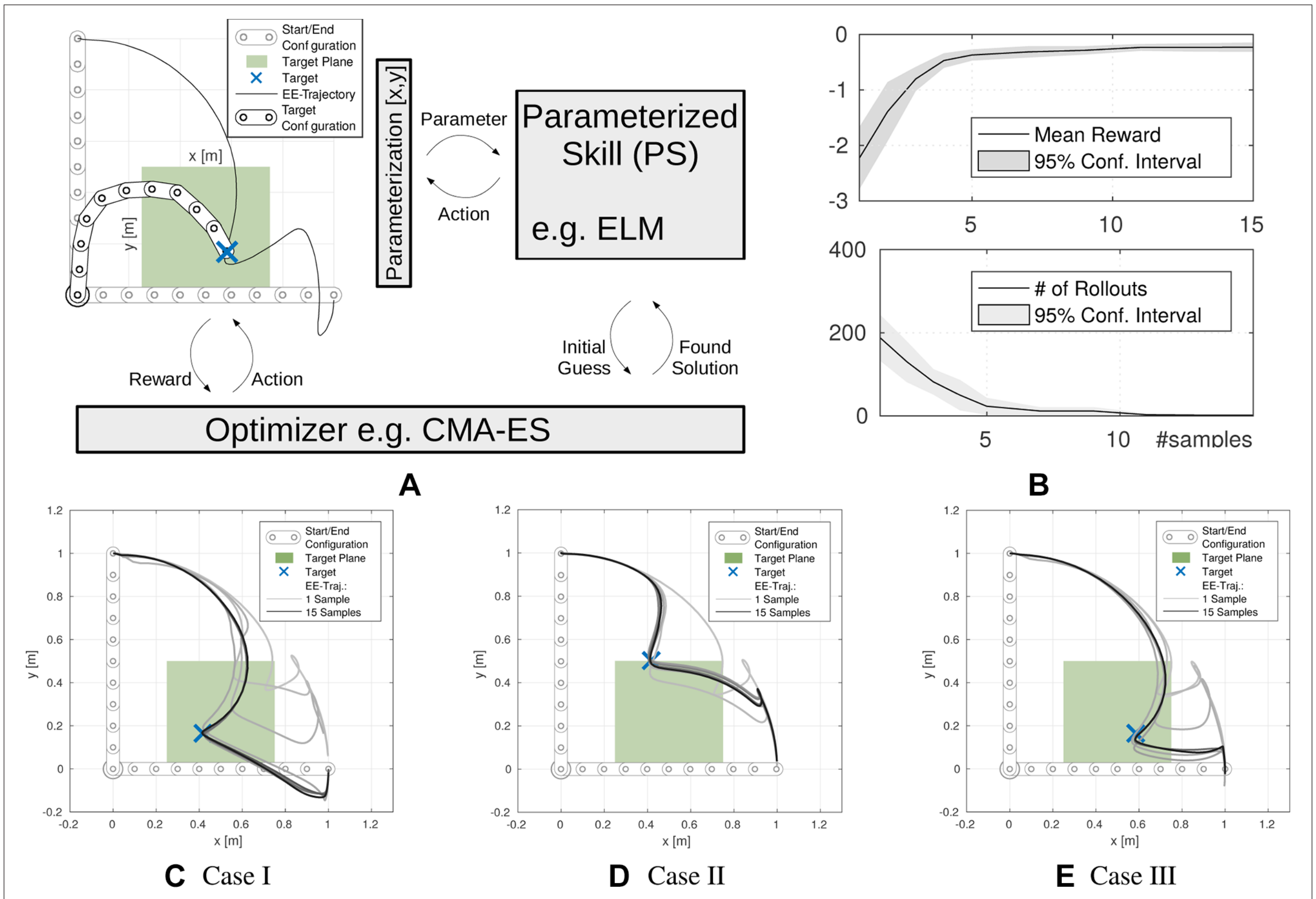


FIGURE 2 | Bootstrapping loop of parameterized skills as proposed in (Queißer et al., 2016). **(A)** System overview including simulation of a 10-DOF planar arm, parameterized reaching target at $\frac{T}{2}$, parameterized skill and optimization module. **(B)** Result of the bootstrapping experiment, as more samples have been presented to the parameterized skill, the higher the initial reward and the lower the number of rollouts to fulfil unseen tasks. In **(C)-(E)**, three exemplary test cases for τ are shown. The representation of the parameterized skill in relation to the number (gray scale) of consolidated samples is visualized. © 2016 IEEE, reproduced with permission from (Queißer et al., 2016).

2. PARAMETERIZED SKILLS

We consider policies π_θ that are parameterized by $\theta \in \mathrm{R}^F$. We further assume that tasks are parameterized by $\tau \in \mathrm{R}^E$. Task instances defined by τ are distributed according to the probability density function $P(\tau)$. The task parameterization τ reflects the variability of the task, e.g., position of obstacles, target positions or load attached to an end-effector. With reference to (Silva et al., 2014), we introduce the notion of a parameterized skill, which is given by a function $\mathrm{PS} : \mathrm{R}^E \rightarrow \mathrm{R}^F$ that maps task parameters τ to policy parameters θ. The goal is to find a parameterized skill PS(τ) that maximizes $\int P(\tau)\, J\left(\pi_{\mathrm{PS}(\tau)}, \tau\right)\, d\tau$ where $J(\pi, \tau) = E\{R(\pi_\theta, \tau)|\pi, \tau\}$ is the expected reward for using policy π_θ to solve a task τ. The reward function $R(\pi_\theta, \tau)$ assesses each action of the robot defined by the policy π_θ with respect to the current task parameterization τ. In **Figure 4** we visualize the relation between task space and the policy parameterization. We expect that multiple manifolds for a given task parameterization exist. We therefore have to select one of the candidated manifolds for approximation by the parameterized skill. We support the incremental exploration of a continuous mapping between τ and θ by imposing a respective preference for solutions that are close to the current estimate of the parameterized skill.

3. BOOTSTRAPPING OF PARAMETERIZED SKILLS

We propose an algorithm to bootstrap a parameterized skill PS(τ) by consolidating optimized θ for given τ. We assume that some sort of policy representation, e.g., a motion primitive model, and policy search algorithm, e.g., REINFORCE (Williams, 1992) or CMA-ES (Hansen, 2006), are available. The idea is to incrementally train the parameterized skill PS(τ) with task-policy parameter pairs (τ, θ^*), where θ^* are optimized policy parameters obtained by executing the policy search algorithm for task τ. The key step is that the current estimate PS(τ) of policy parameters is used as initial condition for policy optimization of new tasks τ. With reference

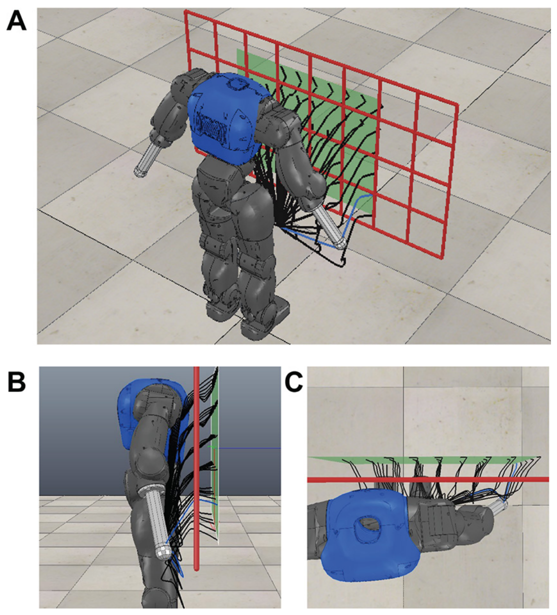
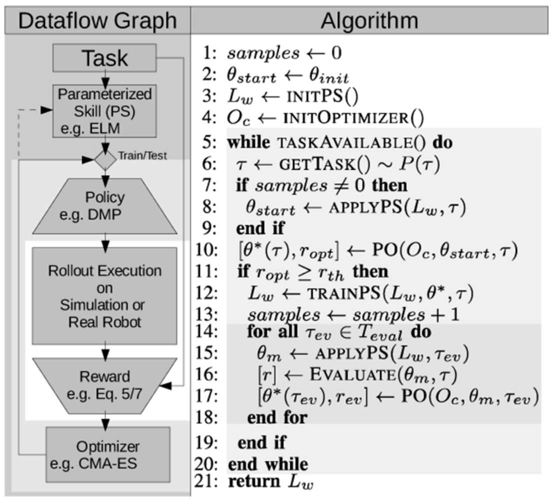


FIGURE 3 | Constrained reaching scenario with a humanoid upper body and a grid-shaped obstacle. Generalized end-effector trajectories for different reaching targets retrieved from the iteratively trained parameterized skill are shown by black lines. © 2016 IEEE, reproduced with permission from (Queißer et al., 2016).



```
1: $samples \leftarrow 0$
2: $\theta_{start} \leftarrow \theta_{init}$
3: $L_w \leftarrow$ INITPS()
4: $O_c \leftarrow$ INITOPTIMIZER()
5: while TASKAVAILABLE() do
6:   $\tau \leftarrow$ GETTASK() $\sim P(\tau)$
7:   if $samples \neq 0$ then
8:     $\theta_{start} \leftarrow$ APPLYPS$(L_w, \tau)$
9:   end if
10:  $[\theta^*(\tau), r_{opt}] \leftarrow$ PO$(O_c, \theta_{start}, \tau)$
11:  if $r_{opt} \geq r_{th}$ then
12:    $L_w \leftarrow$ TRAINPS$(L_w, \theta^*, \tau)$
13:    $samples \leftarrow samples + 1$
14:    for all $\tau_{ev} \in T_{eval}$ do
15:      $\theta_m \leftarrow$ APPLYPS$(L_w, \tau_{ev})$
16:      $[r] \leftarrow$ EVALUATE$(\theta_m, \tau)$
17:      $[\theta^*(\tau_{ev}), r_{ev}] \leftarrow$ PO$(O_c, \theta_m, \tau_{ev})$
18:    end for
19:  end if
20: end while
21: return $L_w$
```

FIGURE 5 | Dataflow and pseudocode of the proposed bootstrapping algorithm. The parameterized skill (PS) estimates a policy parameterization $\boldsymbol{\theta}_{\text{start}}$. 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 p_t^*. © 2016 IEEE, reproduced with permission from (Queißer et al., 2016).

to hypothesis H1 of Sec. 1, the central outcome of this procedure is that policy search becomes very efficient due to incrementally better initial conditions of the policy search. Ultimately, PS(τ) directly provides optimal policy parameters and no further policy optimization needs to be conducted.

The algorithm for the parameterized skill acquisition is outlined in **Figure 5**. For each new task τ, the parameterized skill provides an initial policy parameterization $\boldsymbol{\theta}_{start}$ = startPS(τ) (line 8). After collecting a sufficient number of pairs (τ, $\boldsymbol{\theta}^*$), the proposed parameterization $\boldsymbol{\theta}_{start}$ can achieve satisfactory rewards such that no further policy optimization (PO) by reinforcement learning is necessary. The optimization from initial condition PS(τ) is initiated if the estimated policy parameters can not yet solve the given task or further training is desired (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 (τ, $\boldsymbol{\theta}^*$) with optimized policy parameters

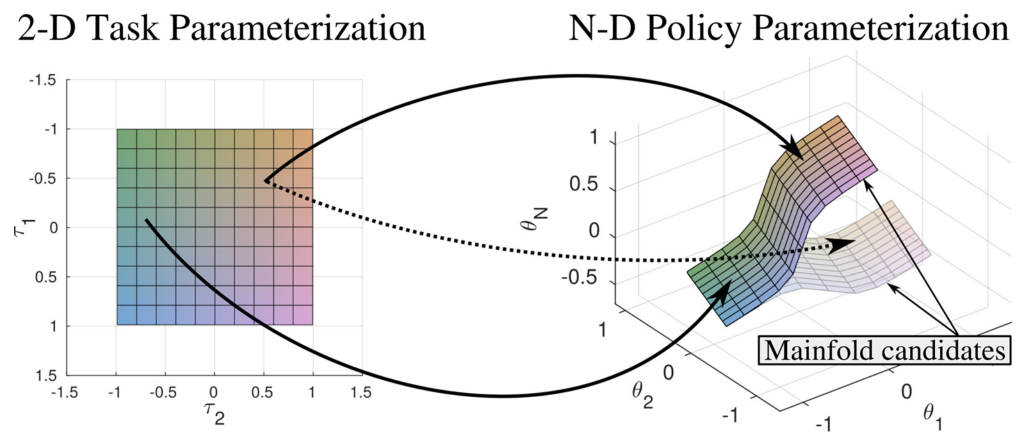


FIGURE 4 | We expect that multiple manifolds exist that are suitable to describe a given task. Therefore the estimation of policy parameterizations that lie close to only one of the manifold candidates allows to estimate a smooth mapping between task and policy parameterization. Policy parameterizations that originate from different manifold candidates can result in ambiguous training data and decrease generalization capabilities of the parameterized skill. Coloring indicates mapping from input space to position on manifold.

$\boldsymbol{\theta}^*$ is used for supervised learning of PS($\boldsymbol{\tau}$) (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.

3.1. 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) (Günter et al., 2007) or Neural Imprinted Vector Fields (Lemme et al., 2014). We employ Dynamic Motion Primitives [DMP, (Schaal, 2006; Ijspeert et al., 2013)], 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

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

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 (Kulvicius et al., 2012). The shape of the primitive is defined by

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

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 θ_k. We assume fixed variances $\boldsymbol{V}_k$ and a regular spacing of centers $\boldsymbol{C}_k$ as in (Ijspeert et al., 2013; Reinhart and Steil, 2015).

3.2. Selection of Policy Optimization Algorithm

We apply the Covariance Matrix Adaptation Evolutionary Strategy [CMA-ES, (Hansen, 2006)] for optimization of DMP parameters $\boldsymbol{\theta}$, given a task $\boldsymbol{\tau}$. Stulp et al. (Stulp and Sigaud, 2013) 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 (Theodorou et al., 2010) or REINFORCE (Williams, 1992), 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 is 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.

3.3. Selection of Learning Algorithm

For learning of parameterized skills PS(τ), we apply an incremental variant of the Extreme Learning Machine [ELM, (Huang et al., 2006)]. ELMs are feedforward neural networks with a single hidden layer:

$$\theta_i\left(\boldsymbol{\tau}\right) = \sum_{j=1}^{H} \mathbf{W}_{ij}^{out}\sigma\left(\sum_{k=1}^{E} \mathbf{W}_{jk}^{inp}\tau_k + b_j\right) \quad \forall i = 1, \quad \ldots, N \tag{3}$$

with input dimensionality E, hidden layer size H and output dimensionality F. Hidden Layer size was set to $H = 50$ for generalization in joint space and $H = 20$ in case of Cartesian end-effector space. Regression is based on a random projection of the input $\mathbf{W}^{inp} \in \mathrm{R}^{H\times E}$, a non-linear transformation $\sigma(x) = (1 + e^{-x})^{-1}$ and a linear output transformation $\mathbf{W}^{out} \in \mathrm{R}^{F\times H}$ 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) (Liang et al., 2006) that incorporates the ability to perform an additional regularization on the weights (Huynh and Won, 2009) or exponential forgetting of previous samples (Zhao et al., 2012). Since we expect to deal with a small number of training samples, regularization of the network can help to prevent over-fitting and foster reasonable extrapolation.

3.4. Experiments

The experimental setup for evaluation of our proposed bootstrapping architecture is shown in **Figure 2A**. The optimization algorithm is initialized by an *initial guess* of the parameterized skill. During optimization the optimizer performs rollouts in simulation and observes rewards resulting of the requested actions. In case an appropriate action that fulfils the task could be found, an update of the parameterized skill is conducted as explained previously in detail in Sec. 3

3.4.1. 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 a 10-DOF planar arm passes through a via-point in task space at time step $\frac{T}{2}$ of the movement with duration T. We considered the kinematics of a 10-DOF planar arm. Motions start at initial configuration $\boldsymbol{q}_{start} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^{\mathrm{T}}$ and stop at configuration $\boldsymbol{q}_{end} = (\pi/2, 0, 0, 0, 0, 0, 0, 0, 0, 0)^{\mathrm{T}}$. 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 $\frac{T}{2}$.

. 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 PS($\boldsymbol{\tau}$) 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 **Figure 2**. We utilize a minimum jerk trajectory (Flash and Hogan, 1985) in joint angle space to generate the initial policy parameters $\boldsymbol{\theta}_{init}$.

The overall reward $R(\boldsymbol{\theta}, \boldsymbol{v})$ is given by:

$$R(\boldsymbol{\theta}, \mathbf{v}) = -\underbrace{\alpha_1 \sum_{t=2}^{T} \left(\frac{\partial^3 \mathbf{p}_t^x}{\partial t^3}\right)^2 + \left(\frac{\partial^3 \mathbf{p}_t^y}{\partial t^3}\right)^2}_{\text{Jerk (a)}} - \underbrace{\alpha_2 \|\mathbf{p}_{T/2} - \mathbf{v}_p\|^2}_{\text{Via Point (b)}} - \underbrace{\alpha_3 \|\boldsymbol{\theta}_{\text{init}} - \boldsymbol{\theta}\|}_{\text{Regularization (c)}} \tag{4}$$

The reward depends on the DMP parameters $\boldsymbol{\theta}$ that result in a 10 dimensional joint trajectory transformed by the kinematics of the robot arm to the end-effector trajectory p_t. The jerk is based on the third derivative of the end-effector trajectory p_t (Hogan, 1984; Flash and Hogan, 1985) and is represented as one objective in the reward function Equation 4(a). Additional terms of the reward function refer to the distance to the desired via-point v of the end-effector trajectory Equation 4(b) and the regularization Equation 4(c).

Coefficients α_i are fixed for all experiments to $\boldsymbol{\alpha} = (10^2, 15, 10^{-3})^{\mathrm{T}}$, resulting 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 $\boldsymbol{\tau}$, i.e., via-point positions drawn from the green target plane in **Figure 2**. Generalization performance is evaluated on a fixed test set $\boldsymbol{\tau}_{ev}$ of $N_{test} = 16$ 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 $F = 60$ dimensional policy parameterization $\boldsymbol{\theta}$. On the top of **Figure 2B**, the mean initial reward for all tasks $\boldsymbol{\tau}_{ev}$ in the test set is shown. The initial reward is based on the reward for estimated policy parameters PS($\boldsymbol{\tau}$) as function of the number of incorporated training samples. The lower part of **Figure 2B** shows that policy optimization benefits from the improved initial policy parameters PS($\boldsymbol{\tau}$) by reducing the number of required rollouts to solve novel tasks (exceed a certain reward threshold). The results show a significant reduction of the required number of rollouts compared to the initialization with the first training sample $\boldsymbol{\theta}^*$, that is regarded as the baseline. **Figure 2C-E** 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. Further evaluation of the properties of the parameterized skill can be found in preliminary work (Queißer et al., 2016), this includes a comparison of different learner and exponential forgetting of old training data.

3.4.2. Reaching Through a Grid

The scenario shows the scalability of the proposed approach to more complex tasks. The goal is to reach for variable positions behind a grid-shaped obstacle while avoiding collisions of the arm with the grid as well as self-collisions. The experiments are performed in simulation of the humanoid robot COMAN (Tsagarakis et al., 2011) as shown in **Figure 3**. We control 7 DOF of the upper body including waist, chest and right arm joints. For the first part of the experiment, motions are represented in Cartesian space utilizing 3 DMPs with $K = 5$ basis functions (as in Equation 2), resulting in a $F = 15$ dimensional optimization problem. The respective DMPs are executed yielding Cartesian end-effector trajectories $\mathbf{p}_t^*$. As before, we utilize minimum jerk trajectories to generate the initial policy parameters θ_{init}. The subset of valid and executable end-effector trajectories $p_{r,t}$ in Cartesian space is given by the kinematics as well as the reachability (e.g., joint limits) of the robot joints.

For each time step t of the desired end-effector trajectory $\mathbf{p}_t^*$, an inverse Jacobian controller estimates a configuration of the robot that executes $\mathbf{p}_t^*$ and maximizes the distance to all obstacles in the null-space of the manipulator Jacobian (Liegeois, 1977):

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

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

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

$$R(\boldsymbol{\theta}, \mathbf{v}_p) = -\underbrace{\alpha_1 \sum_{t=2}^{T} \|\mathbf{p}_t^* - \mathbf{p}_{t-1}^*\|}_{\text{Length of Trajecory (a)}} - \underbrace{\alpha_2 \sum_{t=1}^{T} \|\mathbf{p}_t^* - \mathbf{p}_{r,t}\|}_{\text{Task Tracking (b)}} + \underbrace{\alpha_3 \sum_{t=1}^{T} r_{d,t}}_{\text{Dist. to Obstacles (c)}} - \underbrace{\alpha_4 \|\boldsymbol{\theta}_{\text{init}} - \boldsymbol{\theta}\|}_{\text{Regularization (d)}} \tag{7}$$

where T is the duration of the trajectory. The reward in Equation 7 is a weighted sum of four objectives with weighting factors α_i: (a) The length of the desired end-effector trajectory $p_{d,t}$ that is defined by policy parameter θ [Equation 7(a)]; (b) In addition to the punishment of long trajectories, the reward takes the accuracy of the trajectory tracking into account. Therefore, Equation 7(b) punishes deviations of the reached end-effector position $p_{r,t}$ in relation to the desired end-effector position $\mathbf{p}_t^*$; (c) The distance maximization of all links to the grid obstacle $r_{d,t}$ is considered in Equation 7(c). The optimization criterion representing the maximization of the distance to the grid-obstacle $r_{d,t}$ is given by:

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

a quadratic relationship to the minimum distances $d_{min,l}$ over all L links to all obstacles in the scene in case the distance falls below a given threshold d_B, as in (Toussaint et al., 2007) introduced in the context of null-space constraints for humanoid robot movement generation; (d) An additional normalization for small policy parameterizations as given by Equation 7(d).

The second part of the experiment refers to DMPs in joint space to represent the complete motion of the robot. Therefore the policy parameterization has to represent the maximization of the distance to the grid shaped obstacle as well since no additional inverse Jacobian controller is used. For this experiment we utilize 7 DMPs with $K = 15$ basis functions (as in Equation 2) that generate joint space trajectories, resulting in a 105 dimensional optimization problem. For policy optimization, the reward function is similar to that for the for end-effector trajectories Equation 7. The policy parameters are transformed by DMPs to desired joint space trajectories $\mathbf{p}_t^*$. As previously introduced, Equation 7(b) reflects physical constraints of the robot like joint limits. We initialize $\boldsymbol{\theta}_{\text{init}}$ with joint angle trajectories that allow the end-effector to follow a straight line from start to goal position.

3.5. Results

We evaluated the bootstrapping of the parameterized skill as outlined in **Figure 5**. We selected N_{train} = 20 random target positions on the target plane in front of the robot for training. For evaluation, we created a fixed regular grid for point sampling of N_{test} = 39 positions on the target plane. **Figure 6** reveals that the reward of the initial guess $\boldsymbol{\theta}_{start} = \text{PS}(\tau)$ of the parameterized skill increases with the number of presented training samples. In comparison to the previous experiment in Sec. 3.4.1, the optimization algorithm does not always succeed to find a solution for all tasks of the test set. **Figure 6B** 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 PS($\boldsymbol{\tau}$) for policy optimization reduce the risk to get stuck in local minima during optimization. In terms of number of rollouts that are required to fulfil a new task, we observe similar results as in the 10-DOF arm experiment: The number of required rollouts necessary for task fulfilment decreases the more successfully solved task instances have been presented to the parameterized skill as training data. This results in a bootstrapping and acceleration of the parameterized skill learning. Although the experiments in end-effector space utilize a joint controller that maximizes the distances automatically, the system is able to achieve similar performance in joint space except of a slightly lower success rate.

4. OPTIMIZATION IN HYBRID SPACES

Sec. 3 showed that the utilization of a parameterized skill for policy optimization by CMA-ES can significantly reduce the number of rollouts required to solve unseen tasks. A further option to speed up policy search is given by policy optimization in a lower dimensional search space, as stated by hypothesis H2 of Sec. 1. Previous work of (Koutnik et al., 2010; Fabisch et al., 2013) has already demonstrated that a compression of the parameter space by use of multi layer perceptrons (MLPs) leads to an acceleration of optimization for reinforcement tasks. Reduction of the search spaces by manifolds for value function approximation (Glaubius and Smart, 2005) and abstraction of the whole state-space into sub areas for terrain navigation (Glaubius et al., 2005) can be beneficial in case of reinforcement learning. Constrained optimization problems have been tackled by reducing state-space evaluations and focus on the feasible space of parameters (Barkat et al., 2008). It was demonstrated that the reduction of the number of available bio-mechanical DOF helps stabilize the interplay between environmental and neural dynamics (Lungarella and Berthouze, 2002) for robotic tasks. Dimensionality reduction by freezing or synchronization of joints allows for faster skill acquisition, as shown by (Kawai et al., 2012). Further related work has elaborated the intrinsic dimensionality of human movements and demonstrated

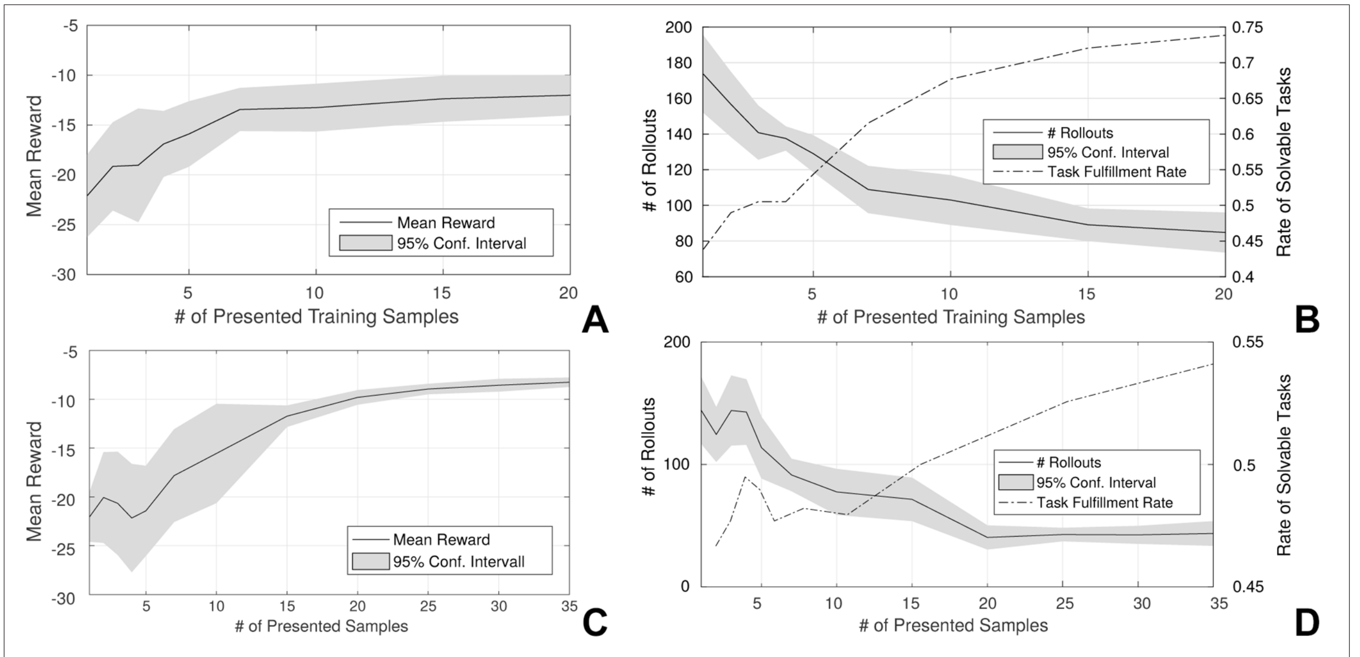


FIGURE 6 | Mean reward of the initial guess $\boldsymbol{\theta}_{\text{start}} = \text{PS}(\boldsymbol{\tau})$ of the parameterized skill in relation to the number of presented training samples **(A/C)** and the mean number of rollouts that are necessary to solve selected test tasks (reward exceeds a threshold) **(B/D)**. Figures **(A/B)** show results of the experiments in end-effector space whereas **(C/D)** show results of the experiments in joint space. The dashed line in **(B/D)** shows the mean rate of solvable task in the test set. Results and confidence intervals are based on ten repeated experiments. Figure **(A/B)** © 2016 IEEE, reproduced with permission from (Queißer et al., 2016).

that dimension reduction is beneficial for reinforcement learning on humanoid robot platforms (Colome et al., 2014).

We assume that previously optimized solutions ($\boldsymbol{\tau}$, $\boldsymbol{\theta}^*$) represent the variability in the task domain and are consolidated in the parameterized skill. We therefore propose to reconsider the parameterized skill as an embedding $f_{\mathbf{emb}}$ of a non-linear manifold of task relevant actions within the full policy space $f_{\mathrm{PS}} : \mathrm{R}^E \to \mathrm{R}^F$, $\boldsymbol{\theta}_{\mathrm{emb}} \mapsto \mathrm{PS}(\boldsymbol{\theta}_{\mathrm{emb}})$. Further, we expect that solutions for unseen tasks are located close to the manifold of the parameterized skill, since we can observe the relation between a higher number of consolidated samples and a higher initial reward, as shown in **Figures 2, 6**. It is thus reasonable to perform policy optimization in the input space of f_{PS}, due to the lower dimensionality of our task parameterization compared to the policy parameterization.

We expect that for points on f_{PS} and their local neighborhood a differentiable map, i.e., a chart of the manifold in the policy space, exists. But on a global scale, it can be expected that the mapping between the task space and the policy space is not invertible since different task parameterizations $\boldsymbol{\tau}$ may require the same policy parameterization $\boldsymbol{\theta}$ and is not differentiable due to e.g., joint limits. Previous work related to our proposed method for dimensionality reduction for policy optimization includes primitive based motion generation by PCA compression (Park and Jo, 2004), lower dimensional primitives that encode differences between trajectories (Stulp et al., 2009) and further library based approaches like (Moro et al., 2012).

But clearly, a search in the task space depends heavily on the number and quality of previously seen samples. We can not expect to be able to find sufficient solutions for all unseen tasks configurations on a low dimensional manifold. More specifically, an exploration on the approximated manifold allows for a coarse search that quickly moves the estimation for $\boldsymbol{\theta}^*$ into the direction of higher rewards. If we are not able to fulfill the given task or we are less efficient to find a better solutions in the task space, we are forced to switch to a slower refinement search in the policy space. But also a temporary switch back from a search in the policy space to the task space would be possible.

Since optimization in policy space is not bound to the manifold of f_{PS}, the joint update between of both spaces requires an inverse estimate of the parameterized skill. We define the local inverse of PS as:

$$\widehat{\mathrm{PS}}^{-1}\left(\boldsymbol{\theta}\right) = \min_{\boldsymbol{\tau}} \| PS\left(\boldsymbol{\tau}\right) - \boldsymbol{\theta} \|, \tag{9}$$

which allows to estimate a point on f_{PS} that gives the closest response for a desired output θ.

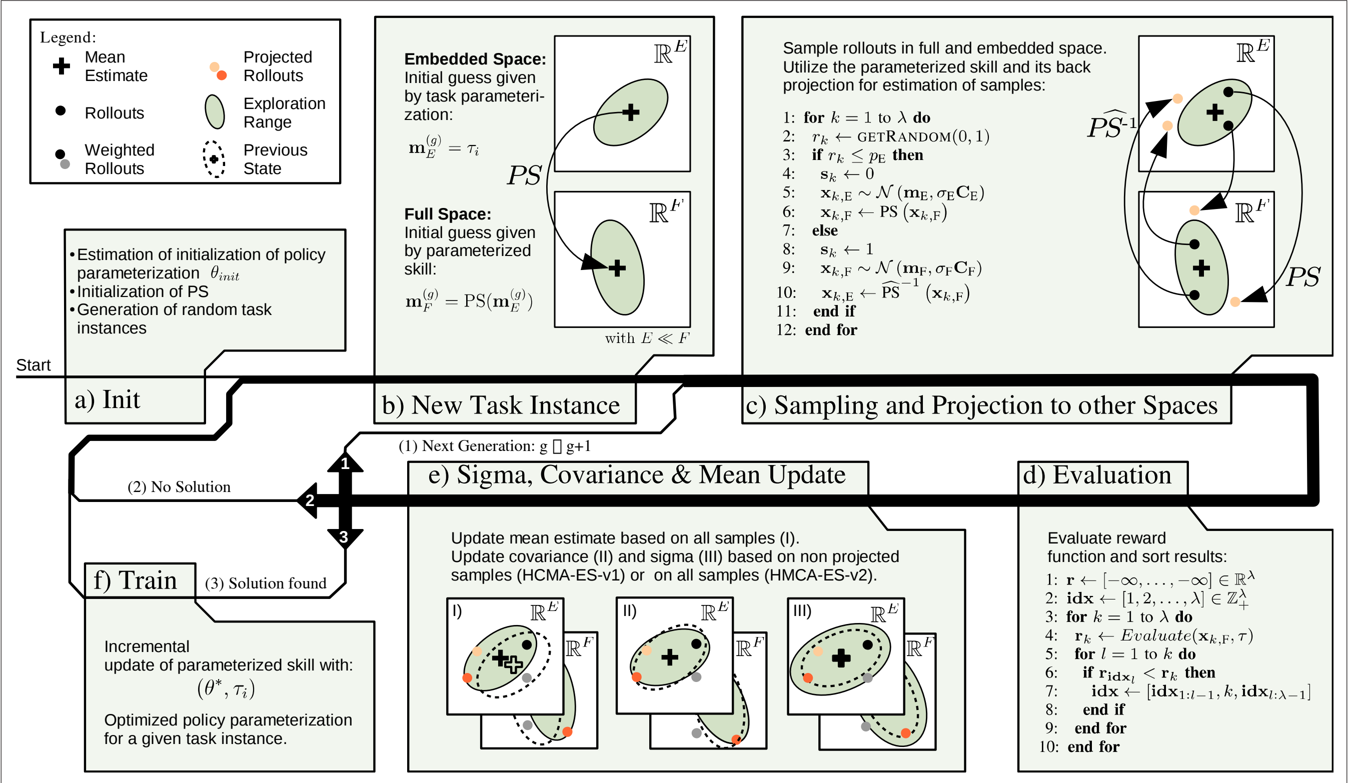
Our approach allows to combine rollouts performed in both spaces for an update of the optimization algorithm. We propose to refer to the success rate of the policies sampled in the respective spaces as defined in Sec. 4.1 for the estimation of the importance of each space during optimization. In general, our combination of optimizers is not bound for a specific optimization algorithm, for this work we refer to a hybrid CMA-ES approach as introduced in Sec. 4.1.

From a policy optimization considering both spaces, we expect the following advantages: First, we expect the algorithm to utilize a low dimensional manifold based on previous samples to perform a fast optimization followed by an optional successive full optimization. Second, by exploration of the manifold based on the parameterized skill, we assume to find solutions that fit to the current estimate of f_{PS}. Therefore, we expect to enhance the consistency of the training data of the parameterized skill for complex reward functions that allow for multiple solutions in policy space. Sections Sec. 5-6 will validate these assumptions. We will visualize and discuss on the ideas on toy data sets and perform algorithm comparisons to CMA-ES in one space.

4.1. CMA-ES in Hybrid Spaces

We refer to CMA-ES (Hansen, 2006) for the implementation of the proposed hybrid optimization method. The original algorithm of CMA-ES relies on four main steps, detailed information can be found in Appendix A. Optimization is performed in generations, which means that an action has to be performed under several perturbations and based on the observation of rewards an updated mean is estimated. CMA-ES has an internal representation of the current mean and of the covariance matrix that allows for sampling of new actions normally distributed around the current mean. In addition, CMA-ES estimates an evolution path for the mean and the covariance matrix update. Those evolution paths allow for more stability to outliers and noise. The first step performs the sampling from a multivariate normal distribution centered at the current estimate (Equation A.1). Followed by the update of the estimated solution for the next generation with respect to the rewards of the sampled rollouts (Equation A.2). The third step targets the update of the covariance matrix and its evolution path (Equation A.3) and (Equation A.4). And the final step performs an update of the exploration width and its assigned evolution path (Equations A.5, A.6).

To be able to perform CMA-ES in hybrid spaces, we apply the CMA-ES algorithm in two parameter spaces simultaneously. We add indices F and E to indicate the affiliation of variables for optimization in policy space (F) and task space (E). Two distinct means $\mathbf{m}_{\mathrm{E}}^{(g+1)}$ and $\mathbf{m}_{\mathrm{F}}^{(g+1)}$ represent the current optimum to minimize the objective function, i.e., negative reward. Covariance matrices $\mathbf{C}_{\mathrm{E}}^{(g+1)}$ and $\mathbf{C}_{\mathrm{F}}^{(g+1)}$ as well as their evolution paths $\mathbf{p}_{c,\mathrm{E}}^{(g+1)}$ and $\mathbf{p}_{c,\mathrm{F}}^{(g+1)}$ allow for random normal distributed perturbation of the respective mean. The variances $\sigma_{\mathrm{E}}^{(g+1)}$ and $\sigma_{\mathrm{F}}^{(g+1)}$ in addition to their evolution paths $\mathbf{p}_{\sigma,\mathrm{E}}^{(g+1)}$ and $\mathbf{p}_{\sigma,\mathrm{F}}^{(g+1)}$ define the exploration size in each space. In comparison to two independent CMA-ES optimizations in each space, we introduce a probability p_{E} respectively $p_{\mathrm{F}} = 1 - p_{\mathrm{E}}$, that indicates in which space we sample the parameterization for rollouts. p_{E} and p_{F} can be interpreted as mixing coefficients that allow for interpolation between a CMA-ES optimization in the task space ($p_{\mathrm{E}} = 1$) and a CMA-ES optimization in policy space ($p_{\mathrm{E}} = 0$). For each update step we estimate $k = 1, \ \ldots, \ \lambda_{\mathrm{H}}^{(g+1)}$ samples in generation $(g+1)$ for a combined update and annotation $\mathrm{s}_k^{(g+1)} = 0$ if rollout k was sampled in the task space or $\mathrm{s}_k^{(g+1)} = 1$ if sampled in the policy space. The initialization (**Figure 7B**) for a new task instance i of the parameterization in the embedded space



- Estimation of initialization of policy parameterization θ_{init}
- Initialization of PS
- Generation of random task instances

Embedded Space: Initial guess given by task parameterization: $\mathbf{m}_E^{(g)} = \tau_i$

Full Space: Initial guess given by parameterized skill: $\mathbf{m}_F^{(g)} = \mathrm{PS}(\mathbf{m}_E^{(g)})$

Sample rollouts in full and embedded space. Utilize the parameterized skill and its back projection for estimation of samples:

```
1: for $k = 1$ to $\lambda$ do
2:   $r_k \leftarrow$ GETRANDOM$(0, 1)$
3:   if $r_k \leq p_E$ then
4:     $\mathbf{s}_k \leftarrow 0$
5:     $\mathbf{x}_{k,E} \sim \mathcal{N}(\mathbf{m}_E, \sigma_E \mathbf{C}_E)$
6:     $\mathbf{x}_{k,F} \leftarrow \mathrm{PS}(\mathbf{x}_{k,F})$
7:   else
8:     $\mathbf{s}_k \leftarrow 1$
9:     $\mathbf{x}_{k,F} \sim \mathcal{N}(\mathbf{m}_F, \sigma_F \mathbf{C}_F)$
10:    $\mathbf{x}_{k,E} \leftarrow \widehat{\mathrm{PS}}^{-1}(\mathbf{x}_{k,F})$
11:  end if
12: end for
```

Evaluate reward function and sort results:

```
1: $\mathbf{r} \leftarrow [-\infty, \ldots, -\infty] \in \mathbb{R}^\lambda$
2: $\mathbf{idx} \leftarrow [1, 2, \ldots, \lambda] \in \mathbb{Z}_+^\lambda$
3: for $k = 1$ to $\lambda$ do
4:   $\mathbf{r}_k \leftarrow Evaluate(\mathbf{x}_{k,F}, \tau)$
5:   for $l = 1$ to $k$ do
6:     if $\mathbf{r}_{\mathbf{idx}_l} < \mathbf{r}_k$ then
7:       $\mathbf{idx} \leftarrow [\mathbf{idx}_{1:l-1}, k, \mathbf{idx}_{l:\lambda-1}]$
8:     end if
9:   end for
10: end for
```

Update mean estimate based on all samples (I). Update covariance (II) and sigma (III) based on non projected samples (HCMA-ES-v1) or on all samples (HMCA-ES-v2).

Incremental update of parameterized skill with: (θ^*, τ_i)

Optimized policy parameterization for a given task instance.

FIGURE 7 | Proposed optimization loop for bootstrapping of parameterized skills in hybrid spaces. After initialization **(A)**, optimization for a new task instance is initiated **(B)**. Optimization is performed (1) until stopping criterion is reached and no solution was found (2) or the optimized solution fulfils the task (3). Update of CMA-ES (I-III) is performed for the task and policy space simultaneously.

is $\mathbf{m}_E^{(g=0)} = \boldsymbol{\theta}_i$ and the initialization in full space is given by the generalization of the parameterized skill $\mathbf{m}_F^{(g+1)} = PS\left(\mathbf{m}_E^{(g=0)}\right)$. The sampling of rollouts as shown in **Figure 7C** is defined in (Equation B.1), for each rollout the target space for sampling is based on probabilities p_E and p_F. The number of evaluated rollouts per generation is defined as a linear interpolation between λ_E and λ_F of both spaces as shown in Equation B.2. In a next step, the parameterized skill performs a mapping of samples originated in task space to policy space and vise versa (see **Figure 7C**, line 6 and 10). Therefore, a parameterized skill is required that allows for inverse evaluation notated as $\widehat{\mathrm{PS}}^{-1}$. This mapping process is given by Equation B.3. A representation of all rollouts in $\mathbf{x}_{k,F}^{(g+1)}$ allows for execution of the policy and evaluation of the reward function. The rollouts are ordered based on the magnitude of the respective reward as proposed by the original CMA-ES approach (Hansen, 2006), as shown in **Figure 7D**. At this point an update of the means $\mathbf{m}_E^{(g+1)}$ and $\mathbf{m}_F^{(g+1)}$ with respect to $\mathbf{x}_{k,E}^{(g+1)}$ and $\mathbf{x}_{k,F}^{(g+1)}$ by applying Equation A.2 is possible. This allows for an update of the estimated means in both spaces based on all rollouts that have been evaluated in the current generation. Note, that the means $\mathbf{x}_k^{(g+1)}$ do not develop independently. Rather they are linked by the mapping of the parametrized skill. For an adaptation of the ratio of rollouts performed in the policy and task space (p_F and p_E), we utilize the success rate of both spaces. The success rate is defined as the ratio of successful rollouts (rollouts that exceed the current reward maximum), encoded by the weights $\mathbf{w}_k^{(g+1)}$ as well as space that was used for sampling $\mathrm{s}_k^{(g+1)}$ of the performed rollouts, as shown in Equation B.4.

We evaluate two approaches for an update of the covariance and exploration width: The first version utilizes only samples that originate in the same space for an update of the covariance C and exploration width σ. The second version utilizes the mapping of PS and $\widehat{\mathrm{PS}}^{-1}$ to estimate an additional update of the covariance and the exploration width with respect to all samples. We refer to the first version as **H**ybrid **C**ovariance **M**atrix **A**daptation - **E**volutionary **S**trategy - **V**ersion **1** (*HCMA-ES-v1*) and to the second version as *HCMA-ES-v2* .

4.1.1. HCMA-ES-V1

The update of the covariance, exploration radius and their evolution paths is performed as in the original CMA-ES algorithm, depicted in Equations A.3–A.6. The update step for each space, encoded in $\mathrm{s}_k^{(g+1)}$, considers only rollouts sampled in the same space. The normalization of $\sum \mathbf{w}_k^{(g+1)} = 1$ for all $\mathrm{s}_k^{(g+a)} = 0$ in case of the task space as well as $s_k^{(g+a)} = 1$ in case of the policy space is necessary since not all samples are used in each space. Additionally, the estimation of $\mu_{eff,\mathrm{E}}$ and $\mu_{eff,\mathrm{F}}$ with respect to s has to be performed as well. The neglection of projected samples for the update of the covariance and its exploration width allows for simplification of the combination. But this simplification prevents that e.g., the covariance in the high dimensional policy space is able to shape into the direction of samples along the low dimensional manifold of the task parameterization.

4.1.2. HCMA-ES-V2

The parameterized skill can be regarded as a mapping between the high dimensional policy parameterization and a low dimensional embedding. The shape of the multivariate normal distribution that is used for samples that get mapped to other spaces does not include a normal distribution in the target space due to the nonlinear transformation of the parameterized skill. For the integration of projected samples in the update of the covariance, we perform a rescaling of projected samples to cancel out the effect of the exploration width σ. The update of the exploration width σ requires the estimation of the distribution of projected samples, but the estimation of a covariance requires a much larger number of samples which is not feasible for our scenarios ($\approx$ 10 rollouts). We decided for an update of exploration width $\sigma_{\mathrm{E}}^{(g+1)}$ and $\sigma_{\mathrm{F}}^{(g+1)}$ with respect to each other by the estimation of a scaling factor between the evaluated rollouts of the current generation which is applicable for low sample numbers.

To consider samples from other spaces for an update of the covariance and its evolution path, samples from other spaces have to be rescaled to keep the covariance C at a constant size i.e., $det\ (\mathbf{C}^T\ \mathbf{C}) = \prod\lambda_i = const.$, the product of eigenvalues of $\mathbf{C}$ is constant. From the update of the exploration width given in Equation A.6 and Equation A.5, we are able to infer the condition $\sqrt{\mu_{eff}}\mathbf{C}^{(g)^{-\frac{1}{2}}}\mathbf{y}_w = \mathrm{E}\|\mathcal{N}\left(\mathbf{0},\mathbf{I}\right)\|$ for constant covariance size. Therefore, we add an appropriate scaling factor to the calculation of the weighted sum, resulting in a modified estimation of $\tilde{\mathbf{y}}_{\mathrm{w,E}}$ with an additional scaling of samples that originate in the full space Equation B.5. The estimation of $\tilde{\mathbf{y}}_{\mathrm{w,F}}$ is performed likewise, given by Equation B.6. The update of $\mathbf{p}_c$ and $\mathbf{C}$ can be achieved by Equation A.3, A.4 with respect to $\tilde{\mathbf{y}}_{\mathrm{w,E}}$ and $\tilde{\mathbf{y}}_{\mathrm{w,F}}$. The final step updates the exploration width $\sigma_{\mathrm{E}}^{(g+1)}$ and $\sigma_{\mathrm{F}}^{(g+1)}$. We achieve this by performing a mixing of the updated sigma of the own space and the rescaled sigma of the other space based on the success rate of the spaces (Equation B.7).

We will evaluate the properties of both algorithm versions and compare the results in successive experiments sections Sec. 5 and Sec. 6.

4.2. Implementation of the Parameterized Skill

The proposed optimization method does not rely on a specific learning method. But in comparison to the bootstrapping of the parameterized skills as proposed in Sec. 3, the policy search in hybrid spaces requires an inverse estimate of the parameterized skill. Therefore the learner must be continuous and locally differentiable. Candidates for this task are associative memories due to their intrinsic capabilities to be able to estimate an input for a given output of the mapping. For the evaluations presented in this paper we refer to a different approach, we utilize the Jacobian of the parameterized skill as proposed in Sec. 3 to iteratively estimate a proper input $\boldsymbol{\tau}$ for a required output $\boldsymbol{\theta}$. We refer to the Inverse Function Theorem by (Spivak, 1971) that states that we are able to estimate a local inverse of a function if the determinant of the Jacobian is not zero. The estimation of the change in the policy parameter space is caused by a change in the task space is given by:

$$\Delta\boldsymbol{\theta}^* \approx \mathbf{J}_{PS}\left(\boldsymbol{\tau}^*\right)\Delta\boldsymbol{\tau}^* \tag{10}$$

Since the parameterized skill is not a bijective mapping, multiple solutions can exist. We assume to sample in the local neighbourhood of our current estimate, therefore we initialize the gradient descent with PS($\boldsymbol{\tau}$). Gradient descent is implemented by the Levenberg-Marquardt method (Liu and Han, 2003), also referred to as Damped Least-Squares method as depicted e.g., in (Buss, 2004), due to numerical stability in comparison to pseudoinverse and Jacobian transposed based methods. The incremental update of the estimated task space τ^* is based on the Jaocbian $\boldsymbol{J}_{PS}(\boldsymbol{\tau}^*)$ of PS with respect to the input $\boldsymbol{\tau}^*$:

$$\begin{aligned}\Delta\boldsymbol{\tau}^* &= \mathbf{J}_{PS}(\boldsymbol{\tau}^*)^{\mathrm{T}}\left(\mathbf{J}_{PS}(\boldsymbol{\tau}^*)\mathbf{J}_{PS}(\boldsymbol{\tau}^*)^{\mathrm{T}} + \lambda^2 I\right)^{-1}\mathbf{e}\\ &\text{with } \mathbf{e} = \left(\boldsymbol{\theta}^* - PS(\boldsymbol{\tau}^*)\right)\end{aligned} \tag{11}$$

5. EVALUATION ON TOY SCENARIOS

To gain insight into the proposed hybrid search method, we investigate two test cases. For simplicity and visualization purposes, the policy is defined as the Identity and the reward function operates directly on $\boldsymbol{\theta} \in \mathbb{R}^2$, the 2D space of the policy parameterization. The reward function is parameterized by $\tau \in \mathbb{R}^1$ defining the position of maximum reward in the 2D space. This allows us to plot the reward function in relation to a fixed value of τ. A visualization of both reward functions for several fixed parameterizations τ are shown in **Figure 8**. The color intensity encodes the reward for a given task parameterization $\boldsymbol{\theta}$. The first scenario describes a circular manifold with a maximum at $\mathbf{m}_\tau$, reward is given by:

$$\begin{aligned}R_{\mathrm{a}}\left(\boldsymbol{\theta},\ \tau\right) &= \frac{1}{\sqrt{2\pi\sigma_{\mathrm{a}}^2}} exp - \frac{|atan2\left(\mathbf{m}_\tau \times \boldsymbol{\theta},\right)\mathbf{m}_\tau \cdot \boldsymbol{\theta}|}{2\sigma_{\mathrm{a}}^2}\\ R_{\mathrm{r}}\left(\boldsymbol{\theta}\right) &= \frac{1}{\sqrt{2\pi\sigma_{\mathrm{r}}^2}} exp - \frac{\left(1 - \|\boldsymbol{\theta}\|\right)^2}{2\sigma_{\mathrm{r}}^2}, \text{with } \mathbf{m}_\tau = \begin{bmatrix} sin\left(\tau\right)\\ cos\left(\tau\right)\end{bmatrix}\end{aligned} \tag{12}$$

The reward function includes the angular deviation $R_{\mathrm{a}}(\boldsymbol{\theta})$ as well as the deviation in the radius $R_{\mathrm{r}}(\boldsymbol{\theta})$, which are weighted by Gaussian functions. The overall reward is given by $R_{\mathrm{circular}}(\boldsymbol{\theta}, \tau) = R_{\mathrm{a}}(\boldsymbol{\theta}, \tau) \cdot R_{\mathrm{r}}(\boldsymbol{\theta})$.

The second reward function is based on a branch manifold. For parameterizations $\tau \leq 1$ the maximum reward is located at $[\tau; 0]$. For $\tau > 1$ two maxima can be found at $[\tau; 0]$ and $[\tau; 1 + \tau]$. It is based

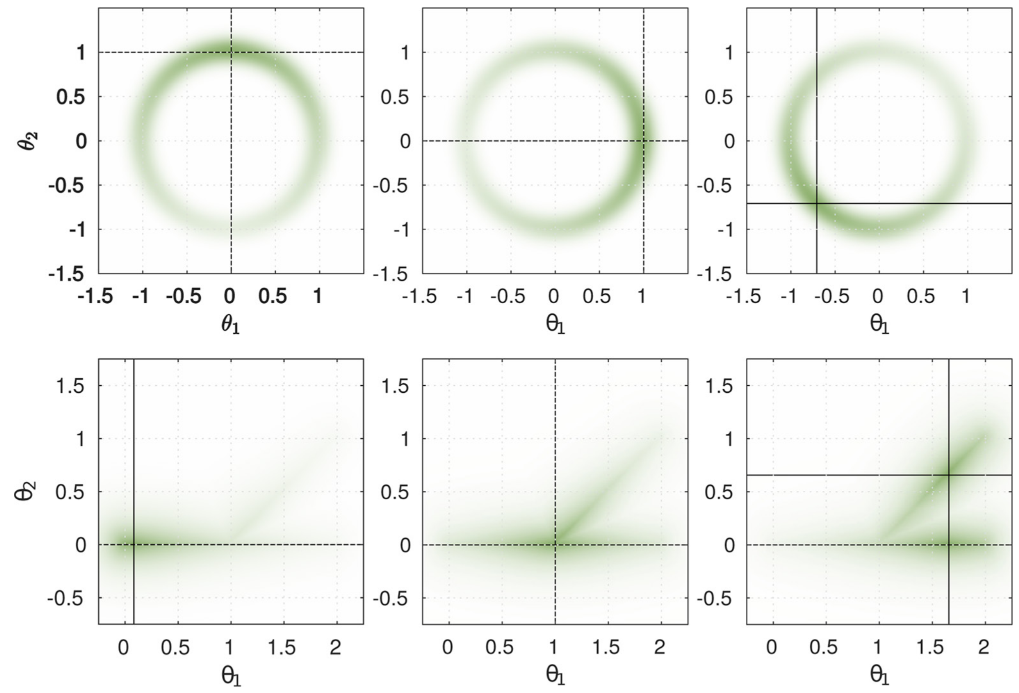


FIGURE 8 | Visualization of designed reward functions. Circular reward unction *R*circular (top) and branch reward *R*branch (bottom) for three different task parameterizations are shown. Crossing points of horizontal and vertical black lines indicate maxima of reward functions. For $\tau > 1$ multiple maxima of the reward function exist (bottom-right). Colour intensity indicates magnitude of reward for depicted parameterization $\boldsymbol{\theta}$

on a combination of the distances to the parametrized maxima of the function $R_{\mathrm{m}}(\boldsymbol{\theta}, \tau)$ and the distance to the branch manifold R_{b}:

$$
\begin{aligned}
R_{\mathrm{m}}(\boldsymbol{\theta}, \tau) &= \frac{1}{\sqrt{2\pi\sigma_{\mathrm{d}}^2}} \exp - \frac{\mathrm{d}_{\min}}{2\sigma_{\mathrm{d}}^2}, \text{ with} \\
d_{\min} &= \begin{cases} ||\boldsymbol{\theta} - \left[\tau; 0\right]||, & \text{if } \tau \leq 1 \\ min(||\boldsymbol{\theta} - \left[\tau; 0\right]||, ||\boldsymbol{\theta} - \left[\tau; \tau - 1\right]||), & \text{else} \end{cases} \\
\text{and} \quad R_{\mathrm{b}}(\boldsymbol{\theta}) &= \frac{1}{\sqrt{2\pi\sigma_{\mathrm{r}}^2}} \exp - \frac{\mathrm{Dist}_{\mathrm{branch}}(\boldsymbol{\theta})}{2\sigma_{\mathrm{r}}^2}
\end{aligned}
\tag{13}
$$

With $Dist_{\mathrm{branch}}(\theta)$, the minimum distance of θ to the line segments [0; 0] – [2; 0] and [1; 0 – 2; 1]. The combination of both reward terms results in the final reward function $R_{\mathrm{branch}}(\boldsymbol{\theta}, \tau) = R_{\mathrm{m}}(\boldsymbol{\theta}, \tau) \cdot R_{\mathrm{b}}(\boldsymbol{\theta})$. We designed the scenario to reflect expected real world problems: The space of all possible actions includes a subset of appropriate actions on a manifold that have have higher rewards. Within this subset we expect a maximum of the reward function at parameterizations that solve the task in an appropriate way. We evaluated three cases as shown in **Figure 9** to **Figure 10**. Each plot

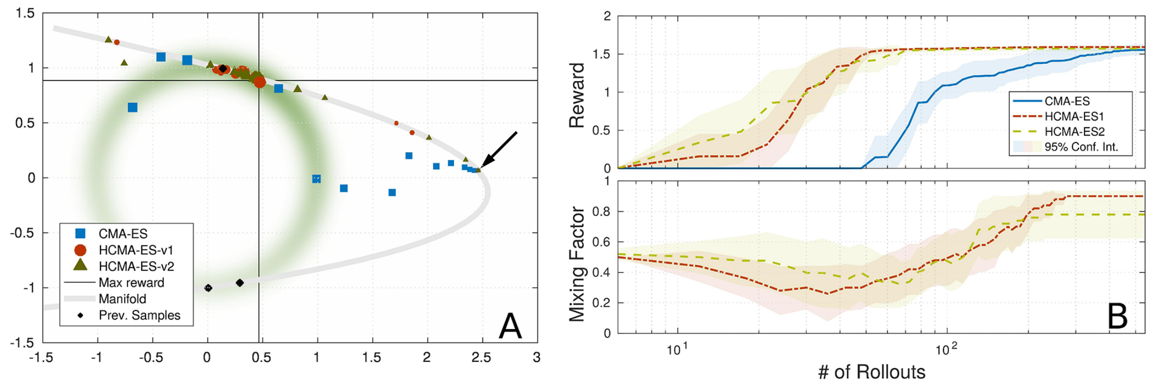


FIGURE 9 | Comparison of optimization algorithms on 2D reward function: Overshoot of PS, hybrid optimization is able to utilize manifold of the parameterized skill (gray line) to perform optimization in 1D space. **(A)** Estimated means of algorithms during optimization, marker size indicates generation. Black arrow points to initial guess on manifold (gray line) of parameterized skill. **(B)** Comparison of reward and mixing factor during optimization is shown.

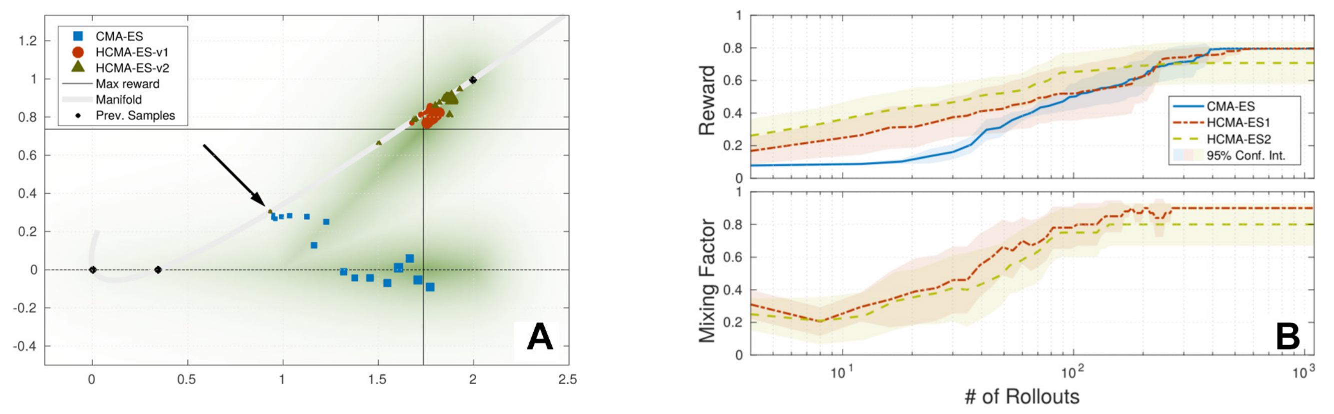


FIGURE 10 | Comparison of optimization algorithms on 2D reward function: Multiple maxima of reward function. **(A)** Estimated means of algorithms during optimization, marker size indicates generation. Black arrow points to initial guess on manifold (gray line) of parameterized skill. **(B)** Comparison of reward and mixing factor during optimization is shown.

shows the comparison between a search in the policy space by CMA-ES as well as the behavior of our proposed hybrid algorithms. The green color intensity encodes the reward for a depicted policy parameterization $\boldsymbol{\theta}$. Previous training data (obtained by estimation of one maximum of the reward function) of the parameterized skill is indicated by a black dot (◆). Based on the training data, the mapping f_{PS} on the manifold in the policy space, i.e., PS(τ), is constructed and shown as a gray line. The symbols ■, ● and ▲ represent the current estimates of the means $\mathbf{m}_F^g$ in the policy space, whereas the size ■-■, ●-● and ▲-▲ show the history of previous mean estimates $\mathbf{m}_F^{g-n}$, $\forall n \in \{1, \ldots, g\}$ up to the first generation with decreasing size of the symbol. The real maximum of the reward function is marked by black crossing lines and the location of the initial estimate $\boldsymbol{\theta}_{start}$ = PS(τ_i) on f_{PS} is highlighted by a black arrow (➚).

In the following we will describe all three scenarios and the optimization process in detail.

5.1. Overshoot

The scenario in **Figure 9** shows a situation in which an overshoot of the estimation of the parameterized skill occurs. We utilize the circular reward R_{circular} and perform an exponential distortion $f(\tau) = exp(\tau) * \pi / \exp(\pi)$ of the parameterization to enforce a generalization error of the memory resulting in $R(\boldsymbol{\theta}, \tau) = R_{\mathrm{circular}}[\boldsymbol{\theta}, f(\tau)]$. For training of the parametrized skill we estimate optimal parameterizations $\boldsymbol{\theta}_i$ for three different tasks τ_i.

For the depicted task parameterization, the parameterized skill proposes a solution that is located in a region with a little gradient information. By following the low dimensional embedding of the parameterized skill the hybrid approach is able to guide the optimizer into a region with stronger gradient and that is closer to a desired maximum of the reward function. In case of the original CMA-ES approach, it takes longer to reach a region with more informative gradient information and requires therefore more rollouts in comparison to the hybrid optimization in both spaces. But the evaluation of this scenario not always leads to a faster convergence of HCMA-ES-v1 and HCMA-ES-v2 in comparison to CMA-ES. In cases, where the estimate of the parameterized skill is of very low quality, optimization in the low dimensional space of f_{PS} can lead to a fast convergence to a region with higher rewards (as in **Figure 9**). But the algorithm could end up in an area that is far away from the final solution, so that an optimization by CMA-ES can reach a high reward with less number of executed rollouts. This results in a fast rising reward at the beginning of the hybrid optimization process followed by a period with a slowly rising reward as the estimate moves along the manifold towards the optimum of the reward function.

Algorithm HCMA-ES-v1 and HCMA-ES-v2 show comparable performance and a similar behavior during optimization. Investigations of the shape of the covariance reveal the extended update policy of HCMA-ES-v2 . Since the shape and size of the covariance of the policy space integrates rollouts sampled in the task space as well, the covariance grows and shapes aggressively into the direction of the real maximum and the shape of the manifold of the parameterized skill. Close to the maximum, i.e., the covariance shrinks but keeps the shape influenced by the previous fast approaching phase in the low dimensional manifold. **Figure 9B** shows the probability of performing a rollout in the policy parameter space, starting at equal probabilities for both spaces, the algorithm first shifts its focus to the task space and switches to a fine-tuning at the end of the optimization phase.

5.2. Multiple Minima

This scenario explores tasks with multiple solutions for a certain range of tasks parameterizations. This time we utilize the circular reward R_{branch} in combination with the exponential distortion $f(\tau) = exp(\tau) * \pi / \exp(\pi)$ used in the overshoot scenario. Therefore the reward function is given by $R(\boldsymbol{\theta}, \tau) = R_{\mathrm{branch}}[\boldsymbol{\theta}, f(\tau)]$. The presented training samples for the parameterized skill as well as the experimental setup can be seen in **Figure 10**. As discussed in Sec. 2, multiple maxima of the reward function bear the risk of generating inconsistent training data for the parameterized skill and impede generalization capabilities. It is beneficial to prefer solutions for tasks that are close to the manifold of the parameterized skill, in this case solutions on the upper branch of the reward function in **Figure 10A**. A hybrid optimization that performs a search along the manifold of the parameterized skill enhances the

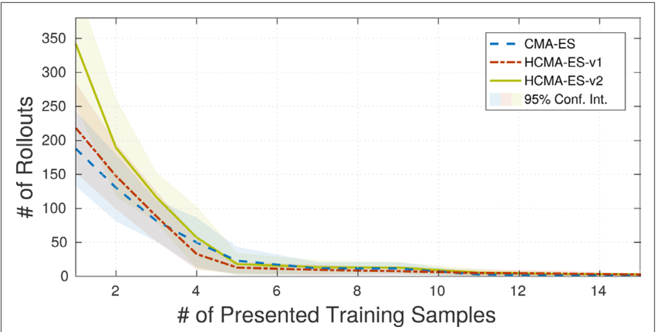


FIGURE 11 | Results of the comparison of HCMA-ES to optimization in the policy parameter space for the point reaching scenario. It can be seen, that the number of required rollouts for task fulfilment is not significantly reduced by one of the optimization techniques.

probability to find an optimum close to the manifold of the already established parameterized skill. As shown in **Figure 10A,B**, starting from the initial guess, the standard CMA-ES approach follows the gradient towards the manifold of the reward function. The covariance, responsible for perturbation of sampling, starts to shape into that direction and causes the optimizer to follow the gradient toward the lower branch of the reward function. The estimated solution is far off the manifold of the parameterized skill and would result in inconsistent training data since previous training data was selected from the upper branch. The standard CMA-ES optimization is able to find a solution for the given task without requiring a significant different number of rollouts than the hybrid optimization methods.

Although the hybrid search can not speed up the optimization process, the optimizer first prefers the manifold of the parameterized skill to move towards the gradient of the reward function, as shown in **Figure 10B**. For the final phase, the optimizer prefers the policy parameter space for optimization and is able to find a maximum of the reward function that is consistent with previous training data since it is located in the upper branch of the manifold of the reward function.

5.3. Results

As shown in **Figure 9** to **Figure 10**, we were able to identify two situations in which our proposed hybrid CMA-ES algorithm is able to speed up optimization significantly. The mean rewards obtained for a given number of rollouts indicate a slightly faster convergence of HCMA-ES-v2 , but we can not observe a strong significant difference between HCMA-ES-v1 and HCMA-ES-v2 for those simple optimization tasks. We show that in case of a faulty estimate of the parameterized skill in a region with low gradient information the hybrid optimization scheme allows a faster convergence. The later experiment identifies a situation in which the consistency of the parameterized skill can be enhanced by a preference of solutions close to the previously established manifold.

6. EVALUATION ON ROBOTIC SCENARIOS

The evaluation of the hybrid optimization scheme as proposed in Sec. 4 refers to the previously performed experiments as described in Sec. 3. We compare the original CMA-ES search in policy space to our hybrid search algorithms. To be able to compare the algorithms without the effect of different states of the memory we stored the memory states during performing the experiments in section Sec. 3.4. In the following experiments, we replicate the same conditions and replace the optimization algorithm by our proposed hybrid spaces optimization methods. **Figure 11** shows the results of the 10-DOF planar arm scenario. HCMA-ES-v2 requires slightly more rollouts for task completion than HCMA-ES-v2 and plain CMA-ES in case the memory has been trained with less than 4 samples. This is caused by updating the covariance matrix of the policy space based on rollouts in task space. To reduce the overhead of the hybrid search algorithms, the initialization of p_E and p_F plays a crucial role. It can be expected that a search in the policy space is more beneficial as long as the number of training samples for the parameterized skill is low. No substantial difference between the CMA-ES and the hybrid search can be seen in the case the parameterized skills consolidated more than 4 samples, The update policies of HCMA-ES-v1 and HCMA-ES-v2 do not lead to significantly different results.

The results for the second scenario, Sec. 3.4.2, show that the proposed hybrid search is able to reduce the number of required rollouts for solving unseen tasks as expected. The parameterized skill of the joint space experiments requires more training samples due to the lack of the inverse Jacobian controller that copes with distance maximization. The results are shown in **Figure 12**, (A–E) show results for experiments in end-effector space in the same way as (F–J) show results for joint space. Both hybrid optimization methods show a tendency to exceed the rate of solvable tasks of the standard CMA-ES method for the experiments in the end-effector space **Figure 12**. The results of the joint space experiments the are not that clear **Figure 12**. The different update policies of HCMA-ES-v1 and HCMA-ES-v2 can be seen by a comparison of the development of the mixing factors p_F in **Figure 12**(C–E; H-J) for 1, 5 and 20 presented samples to the parameterized skill. In case the parameterized skill has a good representation, HCMA-ES-v1 switches to an optimization in the policy space at a later stage **Figure 12**(E+J), whereas HCMA-ES-v2 clearly prefers the policy space for optimization. Both algorithms are switching to a search in the policy space in case of a low number of training samples and in case the memory has seen a certain amount of training samples, HCMA-ES-v2 supports a faster switching from task to policy space search. The visualization of the variance in the low dimensional parameter space is shown in **Figure 13**. We compare three different states of the parameterized skill by plotting estimated solutions for variations of the input around the current task parameter. We can observe different strategies of the robot like approaching the target point from top or from bottom.

7. DISCUSSION AND CONCLUSION

We were able to identify two situations in which we expect our algorithm to exceed the performance of an optimization in policy space, as discussed in Sec. 5. We created three test scenarios, in which we were able to show the benefits of the proposed

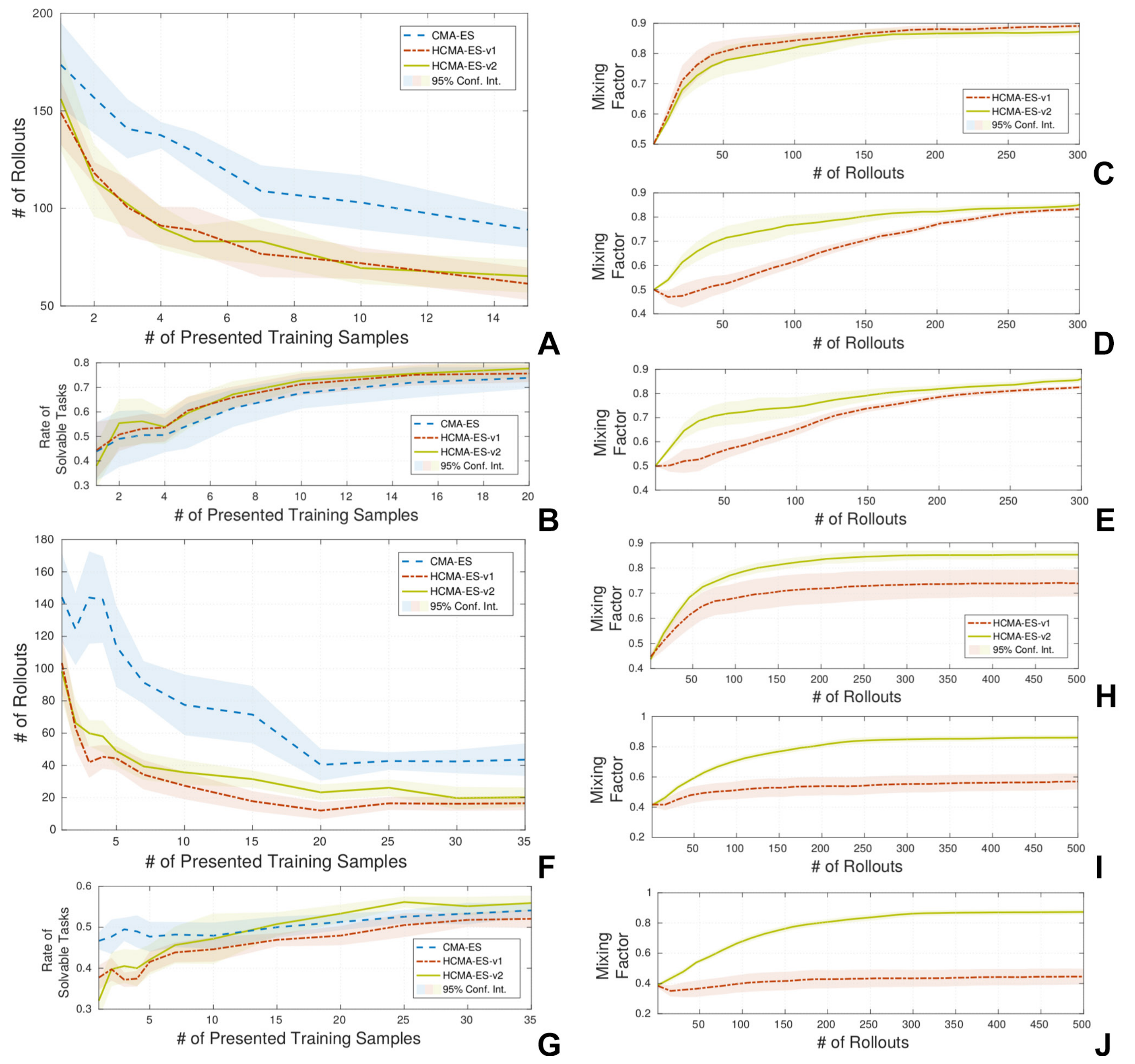


FIGURE 12 | Results of the comparison of HCMA-ES to optimization in the policy parameter space for the point reaching scenario. Experiments **(A–E)** show results in end-effector space and **(F–J)** in joint space It can be seen, that the number of required rollouts for task fulfilment is significantly reduced by the proposed hybrid optimization technique **(A+F)**. The success rate of the optimization process (i.e., exceed a certain threshold on reward) shows stays the same compared to the optimization on the policy parameter space **(B+G)**. In **(C–E; H-J)** the behaviour of the mixing factor between the search spaces is shown for 1**(C+H)**, 5**(D+I)** and 20**(E+J)** training samples.

algorithm as well as one case in which our algorithm underlays a plain policy space search. We could see a clear advantage of the proposed hybrid optimization although the reduction ratio of the task space to the policy parameterization is only 2:1 for these idealized test cases. The scalability of our proposed method was evaluated in complex robot scenarios. We were not able to show significant performance improvements of the hybrid search for the optimization of a 10-DOF robot scenario, while an optimization of a point reaching task of a humanoid robot showed the expected advantages of our approach. We believe that the design of the 10-DOF reaching task, e.g., no obstacles, results in a simple reward function in the high dimensional policy space. The optimizer in the full policy space is able to follow the gradient efficiently after initial estimation of the covariance, e.g., direction, and a reduction of the search space is not necessary. In such a situation, our algorithm is not able

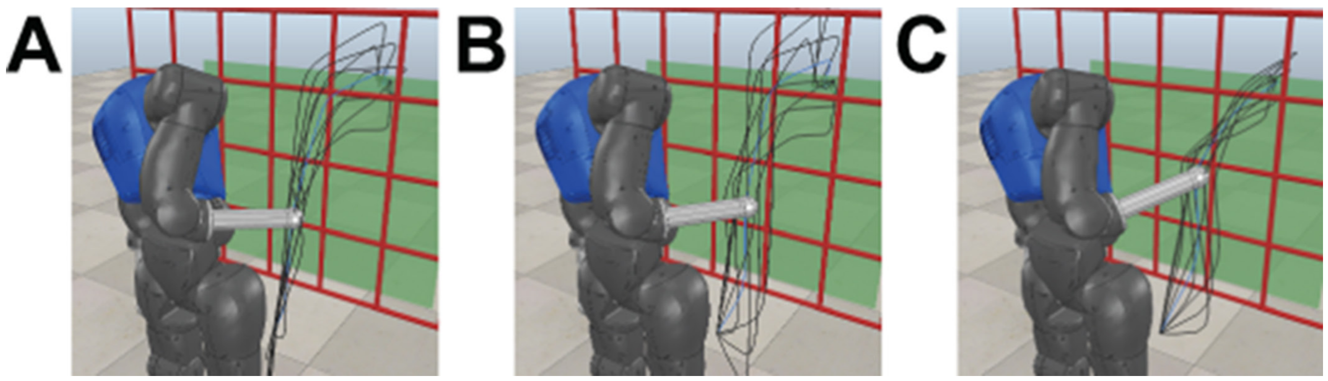


FIGURE 13 | COMAN robot during execution of a estimated end-effector trajectory (blue) of the parameterized skill PS($\boldsymbol{\tau}_i$) for one fixed reaching target $\boldsymbol{\tau}_i$. Black trajectories visualize the variability in low dimensional search space ±50% of the input range PS($\boldsymbol{\tau}_i + \boldsymbol{\delta}$±50%). From left to right, different states of the memory are shown (3,5 and 10 training samples).

to exploit the benefits of the low dimensional embedding of the parameterized skill and has to cope with overhead produced by the combination of both spaces. The skill learning is faced with a much more complex optimization problem, in case of the humanoid robot reaching task, like joint limits and obstacle constraints. Even not all requested task instances are solvable by the kinematics of the robot and CMA-ES can not solve all tasks as it gets stuck in local minima. We are able to show the benefits of our proposed combined optimization scheme for this complex scenario. Although the evaluation was limited to reaching tasks, we demonstrated the applicability of the approach in different domains by an evaluation of control in joint and Cartesian space. An extension to rhythmic movements can be achieved by modification of the underlying DMP representation (Ijspeert et al., 2002). Due to the modular design of the framework other policy representations, black-box-optimizer and learning algorithms can be integrated. One crucial benefit of the point-attractor representation of the DMP is the linearity of its parameterization in relation to the task parameterization (e.g., target position). In comparison to e.g., vector field representations, instabilities can be avoided and the dimensionality of the policy parameterization is reduced. The system is designed to rely on the results of the optimization process, therefore it has no implicit capabilities of dealing with multiple objectives, like in e.g. (Pirotta et al., 2015; Parisi et al., 2017). The pre-designed reward function has to reflect appropriate goals to fulfil the range of parameterized task instances. Policy estimation for multiple objectives can only be achieved by an encoding of the relevance of the objectives as task parameterization.

7.1. Conclusion

We propose the exploration of a parameterized skill by an extension of the CMA-ES optimization to hybrid spaces. We evaluate scenarios in which we are able to observe a low dimensional parameterization for a new task instance. By consolidation of found solutions and their parameterizations of previous tasks we are able to incrementally learn a parameterized skill. The parameterized skill is able to generalize for new policy parameters from task parameterizations, resulting in better start configurations of the optimizer or in the optimal case in a sufficient solution without further optimization. A hybrid search is performed in the space of the policy parameters as well as in the low dimensional manifold that is generated by the parameterized skill in case further optimization is necessary. We have been able to identify and verify several scenarios in which this hybrid approach shows a faster convergence. Additionally we evaluate our approach on complex robotic scenarios targeting on end-effector and joint space control. Our results show that for high task complexity, i.e., upper body reaching, our proposed hybrid optimization is able to significantly speed up policy optimization.

AUTHOR CONTRIBUTIONS

JQ: Conception and design of the research, acquisition, analysis and interpretation of data; Writing of the paper. JS: Conception and design of the research, analysis and interpretation of data; Writing of the paper.

FUNDING

JQ received funding from the Cluster of Excellence 277 Cognitive Interaction Technology and has been supported by the CODEFROR project (FP7-PIRSES-2013-612555) - https://www.codefror.eu/

ACKNOWLEDGMENTS

Sections 1–3 of this manuscript are based on our previous conference paper, © 2016 IEEE (Queißer et al., 2016) and all appropriate permissions have been obtained from the copyright holders for the reproduction of material.

REFERENCES

Baranes, A., Oudeyer, P. -Y., Barkat, U., Abu, S. S. M., Sarker, R., and Cornforth, D. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Rob. Auton. Syst.* 61 (1), 49–73. doi: 10.1016/j.robot.2012.05.008

Barkat, U., Abu, SSM., Sarker, R., and Cornforth, D. (2008). "Search space reduction technique for constrained optimization with tiny feasible space" *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA), 881–888.

Buss, S. R. (2004). Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *Tech. rep., IEEE Journal of Robotics and Automation*.

Cai, C., and Jiang, H. (2013). "Performance comparisons of evolutionary algorithms for walking gait optimization" *IEEE international conference on information science and cloud computing companion* . 129–134.

Colome, A., Neumann, G., Peters, J., and Torras, C. (2014). "Dimensionality reduction for probabilistic movement primitives" *14th IEEE-RAS international conference on humanoid robots, Humanoids 2014* (Madrid, Spain), 794–800.

Fabisch, A., Kassahun, Y., Wöhrle, H., and Kirchner, F. (2013). Learning in compressed space. *Neural Netw.* 42, 83–93. doi: 10.1016/j.neunet.2013.01.020

Flash, T., and Hogan, N. (1985). The coordination of arm movements: an experimentally confirmed mathematical model. *J. Neurosci.* 5 (7), 1688–1703. doi: 10.1523/JNEUROSCI.05-07-01688.1985

Glaubius, R., Namihira, M., and Smart, WD. (2005). "Speeding up reinforcement learning using manifold representations: preliminary results" *Proceedings of the IJCAI workshop on reasoning with uncertainty in robotics (RUR)* 62–70.

Glaubius, R., and Smart, W. D. (2005). Report No: WUCSE-2005-19. Manifold representations for continuous-state reinforcement learning. Tech. rep., Department of Computer Science and Engineering, Washington University. (Accessed 2005-05-01).

Günter, F. (2009). "Using reinforcement learning for optimizing the reproduction of tasks in robot programming by demonstration," in *Ph.D. thesis, STI*. Lausanne.

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

Hansen, N. (2006). *The CMA evolution strategy: a comparing review*. Berlin, Heidelberg: Springer Berlin Heidelberg, 75–102.

Hogan, N. (1984). Adaptive control of mechanical impedance by coactivation of antagonist muscles. *IEEE Trans. Automat. Contr.* 29 (8), 681–690. doi: 10.1109/TAC.1984.1103644

Huang, G. -B., Zhu, Q. -Y., and Siew, C. -K. (2006). Extreme learning machine: theory and applications. *Neurocomputing* 70 (1-3), 489–501. doi: 10.1016/j.neucom.2005.12.126

Huynh, HT., and Won, Y. (2009). "Online training for single hidden-layer feedforward neural networks using RLS-ELM" *IEEE international symposium on computational intelligence in robotics and automation* 469–473.

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., and Schaal, S. (2013). Dynamical movement primitives: learning attractor models for motor behaviors. *Neural Comput.* 25 (2), 328–373. doi: 10.1162/NECO_a_00393

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002). "Learning attractor landscapes for learning motor primitives," in *Proceedings of the 15th international conference on neural information processing systems* 1547–1554.

Kawai, Y., Park, J., Horii, T., Oshima, Y., Tanaka, K., and Mori, H. (2012). "Throwing skill optimization through synchronization and desynchronization of degree of freedom" *16th annual robocup international symposium* 178–189.

Kober, J., Wilhelm, A., Oztop, E., and Peters, J. (2012). Reinforcement learning to adjust parametrized motor primitives to new situations. *Auton. Robots* 33 (4), 361–379. doi: 10.1007/s10514-012-9290-3

Koutnik, J., Gomez, F., and Schmidhuber, J. (2010). "Evolving neural networks in compressed weight space," in *Proceedings of the 12th annual conference on genetic and evolutionary computation* 619–626.

Kulvicius, T., Ning, K., Tamosiunaite, M., and Worgötter, F. (2012). Joining movement sequences: modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Trans. Robot.* 28 (1), 145–157. doi: 10.1109/TRO.2011.2163863

Lemme, A., Neumann, K., Reinhart, R. F., and Steil, J. J. (2014). Neural learning of vector fields for encoding stable dynamical systems. *Neurocomputing* 141, 3–14. doi: 10.1016/j.neucom.2014.02.012

Liang, N. Y., Huang, G. B., Saratchandran, P., and Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Trans. Neural Netw.* 17 (6), 1411–1423. doi: 10.1109/TNN.2006.880583

Liegeois, A. (1977). "Automatic supervisory control of the configuration and behavior of multibody mechanisms" *IEEE Transactions Systems, Man and Cybernetics* 842–868.

Liu, G. R., and Han, X. (2003). *Computational inverse techniques in nondestructive evaluation*, chap. 4.6.2. CRC Press 103–105.

Lungarella, M., and Berthouze, L. (2002). On the interplay between morphological, neural, and environmental dynamics: a robotic case study. *Adapt. Behav.* 10 (3-4), 223–241. doi: 10.1177/1059712302919993005

Matsubara, T., Hyon, S. H., and Morimoto, J. (2011). Learning parametric dynamic movement primitives from multiple demonstrations. *Neural Netw.* 24 (5), 493–500. doi: 10.1016/j.neunet.2011.02.004

Moro, F. L., Tsagarakis, N. G., and Caldwell, D. G. (2012). On the kinematic motion primitives (kmps) - theory and application. *Front. Neurorobot.* 6:10. doi: 10.3389/fnbot.2012.00010

Mülling, K., Kober, J., and Peters, J. (2010). "Learning table tennis with a mixture of motor primitives," in *Proceedings of the 10th IEEE-RAS international conference on humanoid robots (Humanoids 2010)* 411–416.

Parisi, S., Pirotta, M., and Peters, J. (2017). Manifold-based multi-objective policy search with sample reuse. *Neurocomputing* 263, 3–14. doi: 10.1016/j.neucom.2016.11.094

Park, F. C., and Jo, K. (2004). *Movement primitives and principal component analysis*. Dordrecht: Springer Netherlands, 421–430.

Pirotta, M., Parisi, S., and Restelli, M. (2015). "Multi-objective reinforcement learning with continuous pareto frontier approximation," in *Proceedings of the Twenty-Ninth AAAI conference on artificial intelligence* 2928–2934.

Queißer, JF., Reinhart, RF., and Steil, JJ. (2016). "Incremental bootstrapping of parameterized motor skills" *Proceedings IEEE humanoids* 223–229.

Reinhart, R. F., and Steil, J. J. (2015). Efficient policy search in low-dimensional embedding spaces by generalizing motion primitives with a parameterized skill memory. *Auton. Robots* 38 (4), 331–348. doi: 10.1007/s10514-014-9417-9

Schaal, S. (2006). *dynamic movement primitives - a framework for motor control in humans and humanoid robotic*. Tokyo: Springer Tokyo, 261–280.

Silva, BD., Baldassarre, G., Konidaris, G., and Barto, A. (2014). "Learning parameterized motor skills on a humanoid robot" *IEEE international conference robotics and automation* 5239–5244.

Silva, BD., Konidaris, G., Barto, AG., and Castro, B. (2012). "Learning parameterized skills" *international conference on machine learning* 1679–1686.

Spivak, M. (1971). *Calculus on manifolds: a modern approach to classical theorems of advanced calculus*, chap. 2.5. Westview Press.

Stulp, F., Oztop, E., Pastor, P., Beetz, M., and Schaal, S. (2009). "Compact models of motor primitive variations for predictable reaching and obstacle avoidance" *9th IEEE-RAS international conference on humanoid robots* 589–595.

Stulp, F., Raiola, G., Hoarau, A., Ivaldi, S., and Sigaud, O. (2013). "Learning compact parameterized skills with a single regression" *IEEE-RAS international conference on humanoid robots* 417–422.

Stulp, F., and Sigaud, O. (2013). "Policy improvement: between black-box optimization and episodic reinforcement learning" *Journées francophones planification, décision, et apprentissage pour la conduite de systèmes* (France),

Theodorou, EA., Buchli, J., and Schaal, S. (2010). "Learning policy improvements with path integrals" *International conference on artificial intelligence and statistics* 828–835.

Toussaint, M., Gienger, M., and Goerick, C. (2007). "Optimization of sequential attractor-based movement for compact behaviour generation" *IEEE-RAS international conference on humanoid robots* 122–129.

Tsagarakis, N., Li, Z., Saglia, J., and Caldwell, D. (2011). "The design of the lower body of the compliant humanoid robot 'cCub'" *IEEE international conference on robotics and automation* 2035–2040.

Ude, A., Riley, M., Nemec, B., Kos, A., Asfour, T., and Cheng, G. (2007). "Synthesizing goal-directed actions from a library of example movements" *IEEE-RAS international conference on humanoid robots* 115–121.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* 8 (3-4), 229–256. doi: 10.1007/BF00992696

Zhao, J., Wang, Z., and Park, D. S. (2012). Online sequential extreme learning machine with forgetting mechanism. *Neurocomputing* 87, 79–89. doi: 10.1016/j.neucom.2012.02.003

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

APPENDIX A: DETAILS OF CMA-ES UPDATE

Sampling from a multivariate normal distribution centered at the current estimate:

$$\mathbf{x}_k^{(g+1)} = \mathbf{m}^{(g)} + \sigma^{(g)}\mathbf{y}_\mathrm{k}^{(g+1)} \quad \text{with} \quad \mathbf{y}_\mathrm{k}^{g+1} \sim \mathcal{N}_\mathrm{k}(0,(\mathbf{C}^{(g)}) \quad \text{for k} = 1,\ldots,\lambda \tag{A.1}$$

Update of mean m(g+1) given by:

$$\mathbf{m}^{(g+1)} = \mathbf{m}^{(g)} + \sigma^{(g)}\mathbf{y}_\mathrm{w}^{(g+1)}, \quad \text{with } \mathbf{y}_\mathrm{w}^{(g+1)} = \textstyle\sum_{i=1}^{\mu} \mathbf{w}_i \mathbf{y}_{i:\lambda}^{(g+1)} \tag{A.2}$$

Where $\mathbf{x}_{i:\lambda}^{(g+1)}$denotes the i-th best individual and the index i : λ denotes the index of the i-th ranked individual $R(\mathbf{x}_{1:\lambda}^{g+1}) \leq R(\mathbf{x}_{2:\lambda}^{g+1}) \leq \cdots \leq R(\mathbf{x}_{\lambda:\lambda}^{g+1})$. With covariance $\mathbf{C}$(g)∈ $\mathrm{R}^{\mathrm{FxF}}$ scaled by σ(g)∈ R_+ for generation g. Update of the covariance matrix and its evolution path:

$$\mathbf{p}_c^{(g+1)} = (1-c_c)\mathbf{p}_c^{(g)} + \sqrt{1-(1-c_c)^2}\sqrt{\mu_{eff}}\mathbf{y}_\mathrm{w} \tag{A.3}$$

$$\mathbf{C}^{(g+1)} = (1-c_1-c_\mu)\mathbf{c}^{(g)} + c_1\mathbf{p}_c^{(g+1)}\mathbf{p}_c^{(g+1)\top} + c_\mu\sum_{i=1}^{\mu}\mathbf{w}_i\mathbf{y}_{i:\lambda}^{(g+1)}\mathbf{y}_{\mathrm{i}:\lambda}^{(g+1)\top} \tag{A.4}$$

Update of the scaling sigma and its assigned evolution path:

$$\mathbf{p}_\sigma^{(g+1)} = (1-c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{1-(1-c_\sigma)^2}\sqrt{\mu_{eff}}\mathbf{C}^{(\mathrm{g})^{-1/2}}\mathbf{y}_\mathrm{w}^{(g+1)} \tag{A.5}$$

The operation performed by results in a rescaling of the expected distance of samples to the center, as described in Hansen (2006), Eq. 23. The update of sigma is performed by:

$$\sigma^{(g+1)} = \sigma^{(g)} \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma^{(g+1)}\|}{\mathbb{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right) \tag{A.6}$$

APPENDIX B: DETAILS OF HYBRID CMA-ES UPDATE

Update of the optimization by CMA-ES in hybrid spaces as refered to in Sec. 4.1. Update of the estimated means:

$$\begin{aligned} \mathbf{x}_{k,\mathrm{E}}^{(g+1)} &= \mathbf{m}_{\mathrm{E}}^{(g)} + \sigma_{\mathrm{E}}^{(g)} \mathbf{y}_{k,\mathrm{E}}^{(g+1)} && \text{for } k = 1, \ldots, \lambda_H \,|\, \mathrm{s}_k^{(g+1)} = 0 \\ \mathbf{x}_{k,\mathrm{F}}^{(g+1)} &= \mathbf{m}_{\mathrm{F}}^{(g)} + \sigma_{\mathrm{F}}^{(g)} \mathbf{y}_{k,\mathrm{F}}^{(g+1)} && \text{for } k = 1, \ldots, \lambda_H \,|\, \mathrm{s}_k^{(g+1)} = 1 \end{aligned} \tag{B.1}$$

Interpolated number of rollouts per generation given by:

$$\lambda_{\mathrm{H}} = p_{\mathrm{E}}\lambda_{\mathrm{E}} + p_{\mathrm{F}}\lambda_{\mathrm{F}} = 4 + p_{\mathrm{E}}\lfloor 3\ln(E)\rfloor + p_{\mathrm{F}}\lfloor 3\ln(\mathrm{F})\rfloor \tag{B.2}$$

Projection of samples by parameterized skill:

$$\begin{aligned} \mathbf{x}_{k,\mathrm{E}}^{(g+1)} &= PS^{(g)^{-1}}(\mathbf{x}_{k,\mathrm{F}}^{(g+1)}) && \text{for } k = 1, .., .\lambda_H \,|\, \mathrm{s}_k^{(g+1)} = 1 \\ \mathbf{x}_{k,\mathrm{F}}^{(g+1)} &= PS^{(g)^{-1}}(\mathbf{x}_{k,\mathrm{F}}^{(g+1)}) && \text{for } k = 1, .., .\lambda_H \,|\, \mathrm{s}_k^{(g+1)} = 0 \end{aligned} \tag{B.3}$$

Update of probabilities to sample from embedded or full space:

$$\delta p_{\mathrm{E}}^{(g+1)} = \frac{\sum^{\mu}_{\substack{k=1, \\ \mathrm{s}_k^{(g+1)}=0}} \mathrm{W}_k^{(g+1)}}{\sum_{k=1}^{\mu} \mathrm{W}_k^{(g+1)}} - \delta p_{\mathrm{E}}^{(g)}, \quad \delta p_{\mathrm{F}}^{(g+1)} = -\delta p_{\mathrm{E}}^{(g+1)} \tag{B.4}$$

Rescaling of samples for update of covariance:

$$\tilde{\mathbf{y}}_{\mathrm{w,E}} = \sum_{\substack{i=1 \\ \mathrm{s}_{i:\lambda}^{(g+1)}=0}}^{\mu} \mathbf{w}_i \mathbf{y}_{i:\lambda,\mathrm{E}}^{(g+1)} + \frac{\chi_E}{\beta_{\mathrm{E}}} \sum_{\substack{i=1 \\ \mathrm{s}_{i:\lambda}^{(g+1)}=1}}^{\mu} \mathbf{w}_i \mathbf{y}_{i:\lambda,\mathrm{E}}^{(g+1)} \quad \text{with}$$

$$\beta_{\mathrm{E}} = \sqrt{\mu_{eff}} \left\| \mathbf{C}_{\mathrm{E}}^{-1/2} \sum_{\substack{i=1 \\ \mathrm{s}_{i:\lambda}^{(g+1)}=1}}^{\mu} \frac{\mathbf{w}_i \mathbf{y}_{i:\lambda,E}^{(g+1)}}{\alpha_{\mathrm{F}}} \right\|, \quad \alpha_{\mathrm{F}} = \sum_{\substack{j=1 \\ \mathrm{s}_{j:\lambda}^{(g+1)}=1}}^{\mu} \mathbf{w}_j \tag{B.5}$$

With $\chi_N \overset{def}{=} \sqrt{\mathrm{E}(\chi_N^2)} = \mathrm{E}\,\|\mathcal{N}(\mathbf{0}, \mathbf{I}_{\mathrm{N}})\|$ referring to the chi-squared distribution χ_N^2 with N degrees of freedom.

$$\tilde{\mathbf{y}}_{\mathrm{w,F}} = \sum_{\substack{i=1 \\ \mathrm{s}_{i:\lambda}^{(g+1)}=1}}^{\mu} \mathbf{w}_i \mathbf{y}_{i:\lambda,\mathrm{F}}^{(g+1)} + \frac{\chi_N}{\beta_{\mathrm{F}}} \sum_{\substack{i=1 \\ \mathrm{s}_{i:\lambda}^{(g+1)}=0}}^{\mu} \mathbf{w}_i \mathbf{y}_{i:\lambda,\mathrm{F}}^{(g+1)} \quad \text{with}$$

$$\beta_{\mathrm{F}} = \sqrt{\mu_{eff}} \left\| \mathbf{C}_{\mathrm{F}}^{-1/2} \sum_{\substack{i=1 \\ \mathrm{s}_{i:\lambda}^{(g+1)}=0}}^{\mu} \frac{\mathbf{w}_i \mathbf{y}_{i:\lambda,F}^{(g+1)}}{\alpha_{\mathrm{E}}} \right\|, \quad \alpha_{\mathrm{E}} = \sum_{\substack{j=1 \\ \mathrm{s}_{j:\lambda}^{(g+1)}=0}}^{\mu} \mathbf{w}_j \tag{B.6}$$

With β_E^{-1} responsible for a rescaling of samples from policy parameter space to task space and β_F^{-1} from task parameter space to policy space.

And the update of the exploration width:

$$\begin{aligned} \sigma_{\mathrm{E}}^{(g+1)} &= p_{\mathrm{E}}\tilde{\sigma}_{\mathrm{E}}^{(g+1)} + p_{\mathrm{F}} \frac{\tilde{\sigma}_{\mathrm{F}}^{(g+1)} \beta_E}{\chi_E} \\ \sigma_{\mathrm{F}}^{(g+1)} &= p_{\mathrm{F}}\tilde{\sigma}_{\mathrm{F}}^{(g+1)} + p_{\mathrm{E}} \frac{\tilde{\sigma}_{\mathrm{E}}^{(g+1)} \beta_F}{\chi_F} \end{aligned} \tag{B.7}$$