Context-Driven Movement Primitive Adaptation

Daniel Wilbers[1], Rudolf Lioutikov[1], Jan Peters[1,2]

***Abstract*— Humanlike robot skills, e.g., cleaning a table or handing over a plate, can often be generalized to different task variations. Usually, these are start-/goal position, and trained environment changes. We investigate how to modify motion primitives to context changes, which are not included in the training data. Specifically, we focus on maintaining humanlike motion characteristics and generalizability, while adapting to unseen context. Therefore, we present an optimization technique, which maximizes the expected return and minimizes the Kullback-Leibler Divergence to the demonstrations at the same time. Simultaneously, our algorithm learns how to linearly combine the adapted primitive with the demonstrations, such that only relevant parts of the primitive are adapted. We evaluate our approach in obstacle avoidance and broken joint scenarios in simulation, as well as on a real robot.**

I. Introduction and Related Work

Humanlike motions of humanoid robots are desirable in many different areas. In human-robot collaboration settings it is important that the robots intentions are clear to the human. The legibility of robot motions is a requirement for a seamless integration of the robot as a helpful assistant [5]. In settings, where a robot arm is attached to a human as an exoskeleton [6] it is mandatory that the robots movements are humanlike. In this paper, we investigate how to maintain humanlike characteristics in motion primitives, which are learned from demonstrations of a human teacher.

Learning from demonstrations (also known as imitation learning) is a well-established approach for programming robots. We assume, that the reenacted robot trajectories produced by the learned skill from demonstrations naturally inherit humanlike characteristics.

However, directly using demonstrations of a specific task to teach a robot might not be sufficient. Further policy improvement can be necessary for successfully solving a task, e.g. to compensate for the teachers actuation [11].

In comparison to common Motion Planning techniques like STOMP [9], CHOMP [19], and RRTs [14] we do not focus on generating completely new motion plans, but rather on adjusting existing ones. Furthermore, we do not generate single trajectories for a specific task instance, but instead optimize a distribution over trajectories. We use these trajectory distributions to represent a motion primitive, which is a solution to variations of a demonstrated task. In this work we use the Probabilistic Movement Primitives [16] representation, but our approach is applicable to other primitive frameworks as well, e.g., Dynamic Movement Primitives [8].

[1]Intelligent Autonomous Systems, TU Darmstadt
mail-icra@dwilbers.de
{lioutikov,peters}@ias.tu-darmstadt.de
[2]Max Planck Institute Tuebingen

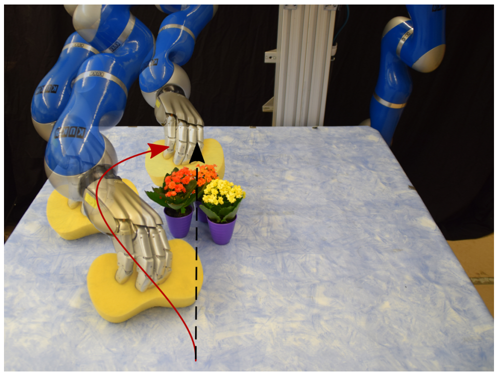


Fig. 1: Illustration of the Table-Cleaning task. The robot should avoid the obstacle with the sponge staying on the table. The dashed black arrow denotes a demonstrated trajectory. The solid red arrow illustrates the context-adapted primitive, which the robot will follow to avoid the flowers on the table.

A key issue of robot skill teaching is the ability of primitives to generalize to different situations. For example a primitive which moves a chess figure forward should be applicable to all tiles on a chess board [2]. Often, the generalization ability corresponds to the adaptation to new start and/or goal positions, while preserving the trajectories shape [20], [10], [17]. Particularly, this means that the primitives can only be adapted to specific pre-trained changes.

Here, we distinguish between two forms of generalizability. The ones described so far are specifically engineered or trained, whereas in this paper we focus on modifying skills to completely new contexts, e.g. new obstacles or broken joints. Various reinforcement learning approaches have been successfully applied to handle different contexts. Work by Pastor et.al. [17] encodes goal-parameters in the primitive representation itself [17], whereas different approaches learn a distribution over meta-parameters [10], [13]. In the latter, variations of the context must be present in the demonstration, so that they can be learned. An advantage of these approaches is, that the adaptation itself is fast to compute, once learned. No further task-relearning is necessary as long as the context changes are covered in the demonstrations.

At the same time, a primitive must be relearned if an adaptation is not possible. In this case, any policy improvement method could be applied, depending on the requirements.

In this paper, we aim to adapt a primitive to an unseen situation in such a way, that it is still applicable to the context which it was already generalized to. An example is adapting a primitive to avoid an obstacle while maintaining

possible start and goal positions. In relation to our work, Lim et al. [15] propose a framework, in which motion primitives are learned from demonstrations with PCA and combined to produce humanlike motions, which are assessed qualitatively. More recently, Huang et. al. [7] introduced a graph- and sample-based motion planner, which can combine two different motions. In their case, multiple upper and lower body motion primitives are coordinated and sequenced to produce a motion graph. The humanlike characteristics arise only from blending between demonstrated motions. Instead of sequencing multiple primitives in order to adapt to a new situation, we investigate how to partially change a single primitive. Closely related to our work Ye and Alterovitz combine motion planning with imitation learning to find task solutions [21]. These can be outside the demonstrations if the demonstrations are blocked. This approach focus on automatically extracting time-dependent task constraints, which are satisfied even after optimization. In our approach we assume the task-constraints are already captured in the given primitives. We punish violations of the task-constraints inside a corresponding reward function.
Being able to re-learn skills through primitive-combination and/or primitive-optimization, such that they can adapt to completely new situations is essential for building and extending skill databases. The contribution of this work is twofold. First we optimize primitives to unseen situations, while binding the solution to stay close to the demonstrations. Second, we also learn how to linearly combine two primitives in order to exactly match the demonstrated primitive given the new context whenever possible.

II. Primitive Optimization for Context-Adaptation

We develop a reinforcement learning strategy, based on the Relative Entropy Policy Search [18] algorithm. The two main differences are the simultaneous optimization of two sub-policies and the explicit minimization of the KL-Divergence [12] to a target distribution of one sub-policy.

A. Notation

Throughout this paper a primitive is defined as a policy $\pi(w)$. The policy represents a Gaussian distribution over parameters $w \sim \mathcal{N}(\mu_w, \Sigma_w)$. Realizations of w can be used to generate a trajectory $\tau(w)$, so that by placing a distribution over w we hierarchically define a distribution over trajectories. In our case we use S radial basis functions to approximate trajectories $\tau(w) = \{y_1, \dots, y_T\}$ as

$$y_t(w) = \sum_{i}^{S} w_i \exp\left(-k(t-c_i)^2\right), \tag{1}$$

where c_i defines the basis center and k modulates the basis width. In order to adapt a trajectory, we learn how to adjust the weights w. Hence, any optimization of a trajectory distribution is equal to adjusting the distribution over w.
In relation to the REPS formalism, we assume that our policy is a joint distribution, which can be split into $\pi(w)$ and $\lambda(a)$. We refer to these as sub-policies of a Gaussian joint-policy

$$\mathcal{N}\left(\begin{bmatrix} w \\ a \end{bmatrix} \mid \begin{bmatrix} \mu_w \\ \mu_a \end{bmatrix}, \begin{bmatrix} \Sigma_w & 0 \\ 0 & \Sigma_a \end{bmatrix}\right) = \pi(w)\lambda(a). \tag{2}$$

By doing so, we can improve two different parameter sets, while both jointly determine a reward function $R(w, a)$. Specifically, we use $\pi(w)$ to represent a trajectory distribution and $\lambda(a) = \mathcal{N}(a|\mu_a, \Sigma_a)$ as a distribution over activation parameters (see Section II-D).
The distribution $\pi_d(w)$ is a target distribution, to which we want to stay close. In general, the target distribution could be arbitrary. In our case it is the policy learned from demonstrations. We denote the context to which we want to adapt the primitive as κ. In the obstacle avoidance scenario κ represents the obstacles position and shape parameters.

B. Problem Statement

Given a target policy $\pi_d(w)$ and an unseen context κ find an optimized joint policy $(\pi(w)\lambda(a))^*$, which maximizes the expected reward of $R(w, a, \kappa)$ while minimizing the KL-divergence $\mathcal{D}(\pi(w)||\pi_d(w))$. The full optimization problem is given as

$$\begin{aligned} \max_{\pi,\lambda} \quad & J = \int_w \int_a R(w,a,\kappa)\pi(w)\lambda(a)\mathrm{d}a\mathrm{d}w \\ & \qquad\qquad - \gamma\mathcal{D}(\pi(w)||\pi_d(w)) \\ \text{s.t.} \quad & \int_w \pi(w)\log\left(\frac{\pi(w)}{q_w(w)}\right)\mathrm{d}w \leq \epsilon_1, \\ & \int_a \lambda(a)\log\left(\frac{\lambda(a)}{q_a(a)}\right)\mathrm{d}a \leq \epsilon_2, \\ & \int_w \pi(w)\mathrm{d}w = 1, \\ & \int_a \lambda(a)\mathrm{d}a = 1. \end{aligned} \tag{3}$$

The distributions q_w and q_a represent the current estimates of the sub-policies, from which we can sample. The first two constraints ϵ_1 and ϵ_2 limit the exploration of the policy update by limiting the KL-Divergence between the current estimate and the new policy. These constraints prevent that the policy is destroyed by an too excessive update step [18]. The last two constraints make sure that each sub-policy is a probability distribution and sum up to one. Solving the optimization problem (see Appendix) with the method of lagrangian multipliers yields the sub-policy update rules

$$\begin{aligned} \pi^*(w) &= \frac{q_w(w)^{\frac{\eta_1}{\gamma+\eta_1}}\pi_d(w)^{\frac{\gamma}{\gamma+\eta_1}}\exp\left(\frac{R(w)}{\gamma+\eta_1}\right)}{\int_w q_w(w)^{\frac{\eta_1}{\gamma+\eta_1}}\pi_d(w)^{\frac{\gamma}{\gamma+\eta_1}}\exp\left(\frac{R(w)}{\gamma+\eta_1}\right)\mathrm{d}w}, \\ \lambda^*(a) &= \frac{q_a(a)\exp\left(\frac{R(a)}{\eta_2}\right)}{\int_a q_a(a)\exp\left(\frac{R(a)}{\eta_2}\right)\mathrm{d}a}. \end{aligned} \tag{4}$$

The terms $R(a)$ and $R(w)$ represent the expected reward for specific parameters w and a given the other sub-policy

$$R(w) = \int_a R(w,a,\kappa)\lambda(a)\mathrm{d}a,$$
$$R(a) = \int_w R(w,a,\kappa)\pi(w)\mathrm{d}w. \tag{5}$$

Due to the recursive dependencies respectively on the other sub-policy we can only approximate $R(a)$ and $R(w)$. Both terms measure how good the parameters w and a perform locally. Given that we iteratively update our policies and restrict the KL-Divergence between update steps we can locally test our parameters against the policies from the previous iteration.
The update of the unrestricted sub-policy $\lambda^*(a)$ is equal to the standard REPS formulation [18]. It is an exponential re-weighting of the old sampling distribution. The restricted sub-policy $\pi^*(w)$ is a geometric average of the sampling distribution, the target distribution, and the exponential returns. By setting $\gamma = 0$ we obtain the standard REPS formulation and both updates have the same form. The parameters η_1 and η_2 are the lagrangian multipliers. The dual formulation leads to the optima η_1, η_2 respectively and is given as

$$\begin{aligned} g(\eta_1,\eta_2) = \\ &- \mathbb{E}\left[R(w,a,\kappa)\right]_{\pi^*\lambda^*} + \eta_1\epsilon_1 + \eta_2\epsilon_2 \\ &+ (\gamma+\eta_1)\log\left(\int_w q_w(w)\left[e^{R(w)}\pi_d(w)^\gamma q_w(w)^{-\gamma}\right]^{\frac{1}{\gamma+\eta_1}}\mathrm{d}w\right) \\ &+ \eta_2\log\left(\int_a q_a(a)e^{\left(\frac{R(a)}{\eta_2}\right)}\mathrm{d}a\right). \end{aligned} \tag{6}$$

Solving Equation 3 reduces to minimizing the dual function

$$\begin{aligned} &\underset{\eta_1,\eta_2}{\text{minimize}} && g(\eta_1,\eta_2) \\ &\text{s.t.} && \eta_i \geq 0,\ i = 1,2, \end{aligned} \tag{7}$$

which is much easier to optimize.

C. Approximation with Samples

In the following we explain how to approximate the integrals and the expectation term in the dual with samples. We estimate the expectation $\mathbb{E}\left[R(w,a,\kappa)\right]_{\pi^*\lambda^*}$, which is part of the original formulation Equation 3, with importance sampling, using $q_w(a)$ and $q_a(a)$ as the sampling distribution. The resulting approximation with importance weights $\psi(w_i,a_i)$ and N samples is

$$\begin{aligned} \mathbb{E}\left[R(w,a,\kappa)\right]_{\pi^*\lambda^*} &\approx \frac{\frac{1}{N}\sum_i R(w_i,s_i,\kappa)\psi(w_i,a_i)}{\frac{1}{N}\sum_i \psi(w_i,s_i)}, \\ \psi(w_i,a_i) &= \left[e^{R(w_i)}\pi_d(w_i)^\gamma q_1(w_i)^{-\gamma}\right]^{\frac{1}{\gamma+\eta_1}} e^{\left(\frac{R(a_i)}{\eta_2}\right)}. \end{aligned} \tag{8}$$

We can further replace the integrals inside both logarithms of Equation 6 using samples based on $\int_y p(y)f(y)\mathrm{d}y \approx \frac{1}{N}\sum_{i=1}^N f(y_i)$. Given these approximations the dual is purely sample-based and can be solved with any constraint optimizer. For numerical stability we suggest to rewrite the sample-based dual according to the log-sum-exp[1] and exp-normalize[2] identities. In our case we exclusively use Gaussians for all policies, so that the new means and variances can be computed with closed-form reward-weighted maximum likelihood updates of the samples s_i [4] as

$$\begin{aligned} \mu_{\text{new}} &= \frac{\sum_{i=1}^N \phi_i s_i}{\sum_{i=1}^N \phi_i}, \\ \Sigma_{\text{new}} &= \frac{\sum_{i=1}^N \phi_i (s_i-\mu)(s_i-\mu)^T}{Z}, \\ \text{with}\quad Z &= \frac{\left(\sum_{i=1}^N \phi_i\right)^2 - \sum_{i=1}^N (\phi_i)^2}{\sum_{i=1}^N \phi_i}. \end{aligned} \tag{9}$$

The term Z is used to calculate an unbiased covariance-estimate. Algorithm-Box 1 gives an overview of the iterative procedure.

Algorithm 1: Primitive Optimization

```
Input: Context Situation $\kappa$, Target Policy $\pi_d(w)$
Output: Optimal sub-policies $\pi^*(w)$ and $\lambda^*(a)$
while not converged do
    begin Policy Evaluation
        Sampling:
        Generate $N$ sample pairs
        $w_i \sim \pi(w)$ , $a_i \sim \lambda(a)$
        Evaluation:
        Compute rewards $R(w_i, a_i, \kappa)$ for each sampled
          pair $i$
        Approximate $R(w_i, \kappa)$ and $R(a_i, \kappa)$
        Optimize: Minimize dual:
        $(\eta_1^*, \eta_2^*) = \text{argmin}\quad g(\eta_1, \eta_2)$
    end
    begin Policy Improvement
        Sub-policy updates $\pi^*(w)$ and $\lambda^*(a)$:
          Compute weights $\phi$ for weighted ML
        $\phi_w(w_i) = [\exp(R(w_i))\,\pi_d(w_i)^\gamma q_w(w_i)^{-\gamma}]^{\frac{1}{\gamma+\eta_1}}$
        $\phi_a(a_i) = \exp\left(\frac{R(a_i)}{\eta_2}\right)$
        For Gaussian distributions:
        Compute improved $\mu_w$,$\Sigma_w$ and $\mu_a$,$\Sigma_a$ (see
          Equation 9)
    end
end
```

The trajectory distributions are represented as $\pi(w)$ and $\pi_d(w)$, while $\lambda(a)$ denotes the distribution over the activation parameters. These distributions could be arbitrary Gaussian distributions for a completely different setting.

D. Combination of Primitives

Combining simple primitives to generate more complex behaviors, which can solve new tasks is a highly desireable feature, especially when using Movement Primitive Libraries. In the following, we describe how we combine

[1]$\log\left(\sum_i \exp(x_i)\right) = k + \log\sum_i \exp(x_i - k)$ with $k = \max x_i$

[2]$\frac{\sum_{i=1}^N a_i \exp(z_i)}{\sum_{i=1}^N \exp(z_i)} = \frac{\sum_{i=1}^N a_i \exp(z_i+k)}{\sum_{i=1}^N \exp(z_i+k)}$ with $k = -\max(z_i)$

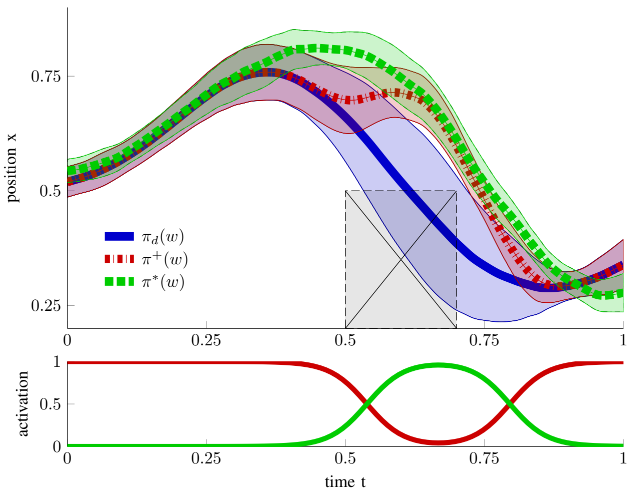


Fig. 2: The demonstrated policy $\pi_d(w)$ is represented as the blue shaded area with mean and two times standard deviation. In green, the optimized policy $\pi^*(w)$ avoids the obstacle, while maintaining the shape of the distribution. In red, the combination of both policies $\pi^+(w)$ avoids the obstacle, but also exactly matches $\pi_d(w)$ in the beginning and end. The corresponding activation function, shown in the second plot, is parametrized with a difference of sigmoid functions and learned accordingly as the sub-policy $\lambda(a)$.

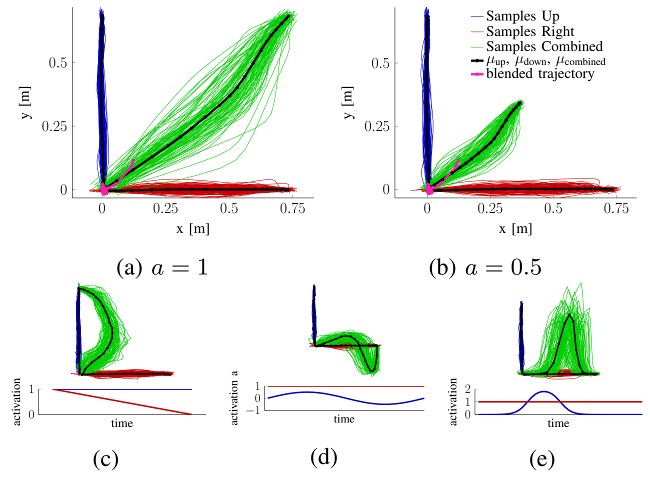


Fig. 3: Linear combination of two primitives, which move up and right respectively, to achieve new behavior.
a) - b): Simultaneous constant activation a over time results in a diagonal movement. Depending on the magnitude of the activation a, the resulting primitives reach further into the upper right.
c) - e): Examples for the effect of different activation functions, which are changing over time. Depending on the characteristics of the activation function the combined primitive can represent completely different behavior.

primitives in such way, that we maintain humanlike characteristic. This can be viewed from two different perspectives. We can either maintain the shape of a trajectory distribution or exactly match the trajectory distribution from demonstrations whenever possible. To achieve the latter we suggest to linearly combine different policies. An illustration of both approaches is given in Figure 2. We make use of some Gaussian properties to combine skills. First, the sum of two Gaussian random variables $x \sim \mathcal{N}(x|\mu_x, \Sigma_x)$ and $y \sim \mathcal{N}(y|\mu_y, \Sigma_y)$ is a Gaussian distribution $z \sim \mathcal{N}(z|\mu_x + \mu_y, \Sigma_x + \Sigma_y)$. Second, the affine transformation $y = c + Bx$ of a Gaussian x is again a Gaussian distribution $y \sim \mathcal{N}(y|c + B\mu, B\Sigma B^T)$. For the combined policy $\pi^+(w)$ we get

$$\begin{aligned} &\pi^+(w) = \mathcal{N}(w|\mu_{\pi^+}, \Sigma_{\pi^+}) \\ &\text{with} \quad \mu_{\pi^+} = A\mu_{\pi_d} + (1 - A)\mu_{\pi^*}, \\ &\qquad\quad \Sigma_{\pi^+} = A\Sigma_{\pi_d}A' + (1 - A)\Sigma_{\pi^*}(1 - A)'. \end{aligned} \tag{10}$$

The combination itself is performed in the weight space of the trajectories. In comparison to the blending approach from [16] we are still able to generate smooth trajectory samples after the combination since we maintain a complete distribution in the weight space. The elements of the diagonal matrix $\mathbf{a} = \text{diag}(A)$ represent the activation factors for each basis weight of the policy. We obtain a combined trajectory distribution $\pi^+(w)$. Typically, we place a probability distribution over the activations directly, so that $\dim(a) = \dim(w)$ or parametrize the activations. For example, if we temporarily want to switch the active robot skill, a useful parametrization would be the difference of two sigmoid functions (e.g. in Figure 2). By doing so we can significantly reduce the number of dimensions. On the other side we must take prior knowledge into account which may also limit the performance. Our approach directly optimizes the activations represented by the sub-policy $\lambda(a)$.

III. Analysis: Different Aspects of our Approach

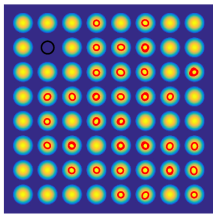
In the following section we individually discuss and demonstrate various aspects of our algorithm.

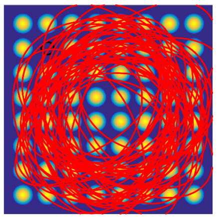
A. *Primitive Combination*

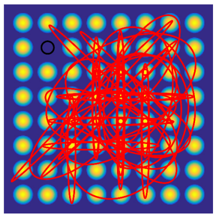
With our linear combination approach, we can achieve a different behavior, based on how we choose the activations. In this example we neglect optimizing the trajectory distribution and focus on the combination itself. Assume we are given two primitives which we want to combine to achieve new behavior. In Figure 3 we show such two planar primitives, which can move up and right respectively. None of them can reach the upper right corner by itself, whereas it is possible with the combination. Simply taking the average as in the $a = 0.5$ case is not enough to fully exploit the combination. The blending approach from [16] fails in this case because it follows the regions with smallest variance. Blending can be useful for fulfilling task-constraints, but not for exploring completely new behavior. The examples in Figure 3(c) to Figure 3(e) show further combinations, yielding completely different results.

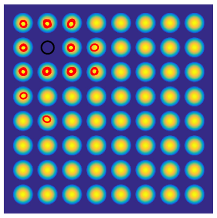
B. *Staying close to Demonstrations*

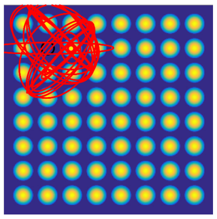
To emphasize the effects of minimizing the KL-Divergence to a target distribution we give a simplified


(a) $\gamma = 0$
$N_i = 10$
$I = 100$


(b) $\gamma = 0$
$N_i = 1000$
$I = 20$


(c) $\gamma = 0$
$N_i = 1000$
$I = 100$


(d) $\gamma = 0.001$
$N_i = 10$
$I = 100$


(e) $\gamma = 0.001$
$N_i = 1000$
$I = 20$

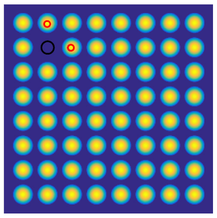
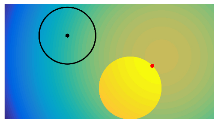
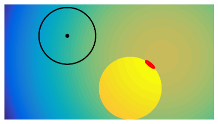
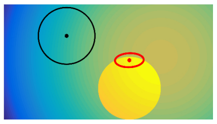
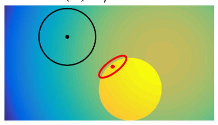

(f) $\gamma = 0.001$
$N_i = 1000$
$I = 20$

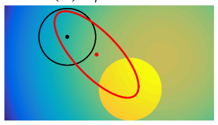
Fig. 4: Comparison between REPS ($\gamma = 0$ case) and our algorithm. Each image contains 50 red ellipses which denote Gaussian distributions after optimization. N_i denotes the number of samples per iteration. I is the number of iterations. The black ellipse denotes the target distribution.
a) - c): REPS either collapses to a random option or maintains a wide distribution over all options.
d) - f): Our algorithm stays close to the target distribution. With a small sample set $N_i = 10$ the solutions are biased towards one option. With enough samples a single option is found near the target distribution.

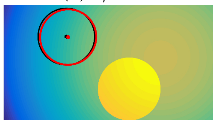

(a) $\gamma = 0$

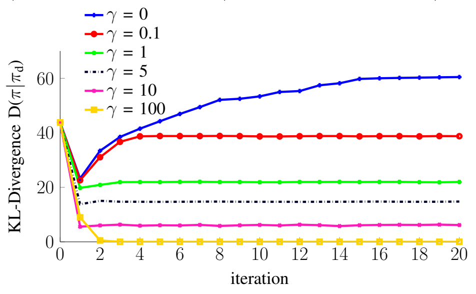

(b) $\gamma = 0.1$


(c) $\gamma = 1$


(d) $\gamma = 5$


(e) $\gamma = 10$


(f) $\gamma = 100$



(g) KL-Divergence

Fig. 5: a) - f): Solutions of the sub-policy $\pi(w)$ in parameter space for six γ variations. All red ellipses contain 90% of the probability mass. The black ellipses denote the target distribution $\pi_d(w)$. The red ellipses are the found solutions $\pi{*}(w)$ after 20 iterations. All experiments are initialized with a much larger variance than the solutions. g): KL-Divergence $\mathcal{D}\left(\pi(w)||\pi_d(w)\right)$ during the iterative optimization for different γ.

example with a highly multimodal reward function, where all optima are equally good. In Figure 4 the bright yellow areas represent a high reward, whereas blue areas are much worse. We now assume that the distribution learned from demonstrations in the upper left area is blocked and can not be reproduced. The substitute solutions should now be as close as possible to the demonstrated distribution. In order to show the effects of our approach sub-policy $\lambda(a)$ does not influence the reward. As we can see, limiting the KL-Divergence yields solutions which are closer to the black target distribution in the upper left. If we compare the solutions on the right side (Figure 4(c) and Figure 4(f)) we see that with enough iterations we can also match the targets variance and hence find a single option.

C. Tuning γ

To emphasize the effects of our approach we demonstrate how the solutions change with different values for the γ-parameter (Equation 3). Basically, γ influences how close to the target distribution we want to be. Setting γ to zero cancels the effect of the KL-Divergence in the optimization. A characteristic of our approach is that we are comparing reward and KL-divergence against each other, which must not necessarily be of the same magnitude. Depending on the reward function tuning γ in an adjusted parameter range is necessary. In Figure 5 we give a simplified example based on a two-dimensional reward function. The toy function is quadratic with an additional circular discontinuity. In order to show the effects of limiting the KL-Divergence sub-policy $\lambda(a)$ has no effect on the reward. The higher the γ-value the more the solutions get pulled towards the target distribution. In the $\gamma = 0$ case the result reduces to REPS and reaches the global optimum of the reward-function. We see the effect in Figure 5(g): The higher γ the lower the KL-Divergence. Notably, the reward is lower the higher γ is. When optimizing we need to cope with this trade-off. As shown in Section III-B if many local optima are available the reward can still be equally good.

In relation to trajectory distributions Figure 6 shows a simplified example based on a one-dimensional trajectory distribution $\pi(w)$ to directly show the effects on the solution.

IV. Evaluation in Simulation and on a Real Robot

We demonstrate different aspects of our algorithm and apply it to various problems. Therefore we evaluate settings in task space, as well as in joint space. In addition to simulations, we also illustrate the execution on a 7-DOF real robot arm.

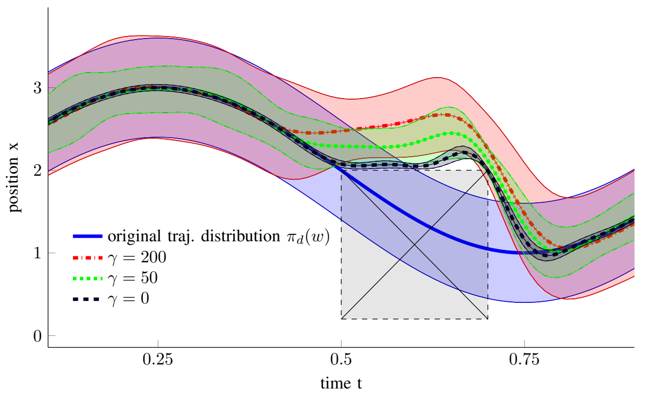
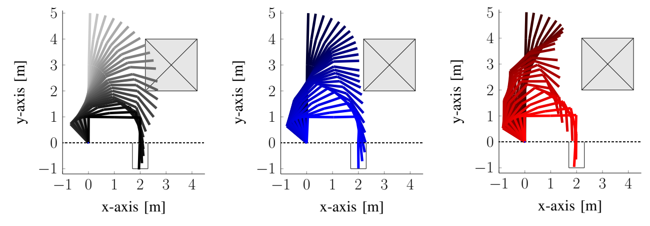


Fig. 6: Comparison of the effects of γ on the optimal sub-policies $\pi^*(w)$. The blue distribution is the target policy $\pi_d(w)$. Sub-Policy $\lambda(a)$ is independent of the reward function and neglected here. The gray rectangle is a restricted area with undesirable parameters.

A. *Hole-Reaching Task*

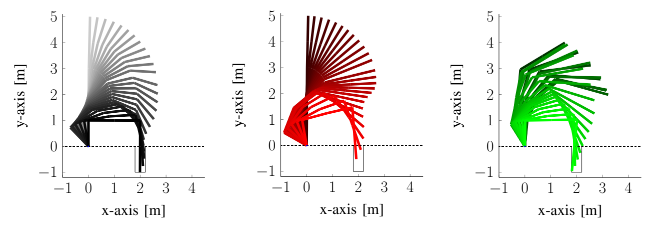
We apply our algorithm on a planar 5-DOF robot. Each link is one meter long and has no joint limits. The robots end-effector must reach the bottom of a hole, which is two meters away, one meter deep, and 30-60cm wide. In total, we optimize 60 parameters. With ten basis functions for each degree of freedom we have a 50-dimensional parameter vector for sub-policy $\pi(w)$. Additionally, we learn the activations a without parameterizing them, but assume all dimensions have the same activation. The reward depends on a collision cost, acceleration punishment, and a reward for reaching the goal position. We initialize our optimization with the policy learned from the demonstrations. In the demonstrations the robot always starts from a roughly upright position and reaches the bottom of the hole.
Obtaining a suitable covariance matrix is especially challenging in this task. Due to the high dimensional parameter space and limited number of available samples per iteration, the weighted-ML updates during the optimization are most likely biased. Because this task is operating in joint space it is crucial that we obtain a proper estimate of the covariances between joints. The first joints highly influence the range of suitable states of the following joints and vice versa. Due to this high correlation a precise estimate of the covariance matrix is necessary for finding successful solutions. Inspired by the CECER approach [1], which in practice is hard to tune, we modify the covariance matrix estimate Σ_w after the weighted-ML update. We use a convex combination of the current estimate Σ_i with the covariance matrix Σ_{i-1} from the last iteration to limit the covariance shrinkage $\Sigma_w^{\text{new}} = \delta\Sigma_w^{i-1} + (1-\delta)\Sigma_w^i$. No further adjustments were needed for Σ_a. We apply this setting in two different context scenarios.

1) Obstacle Avoidance: In the first scenario an obstacle is added to the scene. The robot is supposed to reach the 60 cm wide hole without colliding with the obstacle. As it can be seen in Figure 7 we can successfully learn such a behavior. The optimization is performed in joint space. Thus, we use forward kinematics to calculate collisions with the obstacle, which is given in task space. The obstacle is a one-by-one meter square block. We compare the $\gamma = 0$ case with the best working $\gamma \geq 0$ case ($\gamma = 10^{-7}$). Furthermore, the activation function was only allowed to be either one or zero and was clipped at the first and last value to stay close to the demonstrations. By doing so, we ensure that we specifically learn where to switch between the adapted skill and the one from demonstrations. We don't need to encode a start or goal position inside the reward function, as its maintained through the combination with demonstrations. Both cases were executed with identical parameters except γ. We repeated the experiment five times with each 40 iterations and 250 samples per iteration. Following the ideas of [9], the covariance matrix Σ_w was initialized with a scaled version of the one learned from demonstrations. Therefore only the variances belonging to the middle of the trajectory were scaled up. Additionally, for this task $\delta = 0.9$ was found practical. After we adapted the skill, we tested our results by sampling from the learned trajectory distributions. Sampling 500 trajectories from the original distribution resulted in only 0.008% collision free ones. In the $\gamma = 0$ case 0.45% were collision free, whereas our approach succeeded in average 68% of the time, which is close to one standard deviation.



(a) Original Skill (b) $\gamma = 0$ (c) $\gamma = 10^{-7}$

Fig. 7: Obstacle Avoidance in the Hole-Reaching task. The robot must reach the bottom of the hole without touching the obstacle or ground. b) and c): Mean trajectories of $\pi^+(w)$.



(a) Original Skill (b) $\pi^+(w)$, $\gamma = 50$ (c) $\pi^*(w)$, $\gamma = 0$

Fig. 8: Broken joint scenario in the Hole-Reaching task. b): Optimized mean trajectory $\pi^+(w)$ with learned activations, so that the solution is close to the demonstrations in the beginning. c): Standard REPS solution $\pi^*(w)$ with a different start position since no activations are learned.

The corresponding solutions in joint space are given in Figure 9.

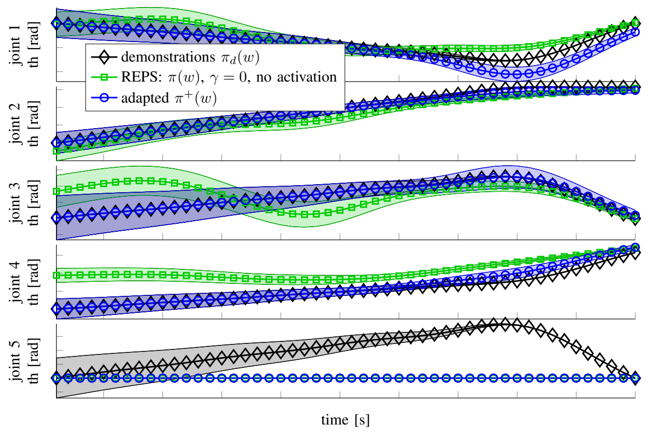


Fig. 9: The five joint dimensions in the broken joint case of the Hole-Reaching task. The working joints are forced to change in order to compensate for the broken 5th joint. The corresponding solutions in task space are given in Figure 8.

2) Broken Joint Scenario: In the second scenario we assume that the last link of the robot is broken and cannot move anymore. Therefore the skill must be adapted, such that the robot can still reach the hole. We enforce parameters of the broken joint dimension to be zero. The reward function is equal to the setting above, but without an additional obstacle. The target distribution is the same distribution, which was learned from demonstrations with all joints working. Figure 8(b) shows our results after 30 iterations with each 2500 samples. In comparison to the REPS case $\pi^*(w)$ with $\gamma = 0$ and without activation, our solution maintains the start position (see Figure 9) while still being able to move the arm into the hole. Figure 10 shows the KL-Divergences in both cases.

B. Table-Cleaning Task

We also tested our approach on a 7-DOF real robot arm. The robot is supposed to pick up a sponge and wipe multiple times over a table. We demonstrate the task three times via kinesthetic teaching with an empty table. We extract the contact points with the table from the learned trajectory distribution to specify the task constraints. During optimization these points should still be in contact with the table. The goal is to execute the skill successfully even if items are still present on the table as illustrated in Figure 1. We optimize in joint space, but currently only check for collisions of the end-effector in task space. With 50 basis functions per degree of freedom and additional 50-activation parameters we optimize a 400 dimensional parameter vector in total. The activation parameters are clipped to either zero or one. As shown in Figure 11 the activations are learned and the obstacles are avoided accordingly. The solution was found after 30 iterations with each 500 samples. Unlike the necessary adjustments of the covariance matrix Σ_w in the Hole-Reaching Task, it was not required for the Table-Cleaning even if the number of parameters is much higher. In the obstacle avoidance case of the Hole-Reaching task

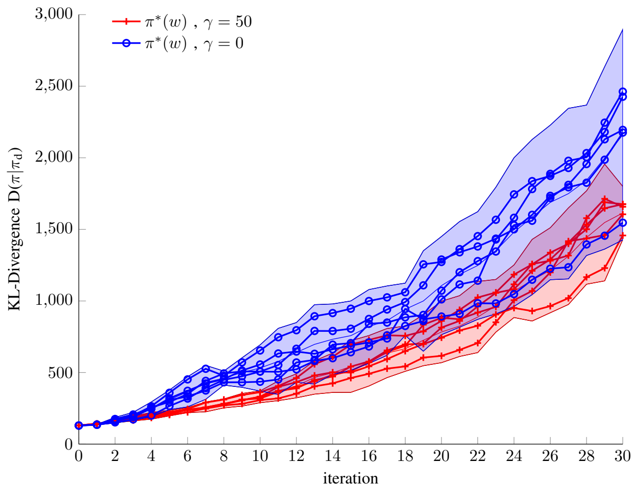
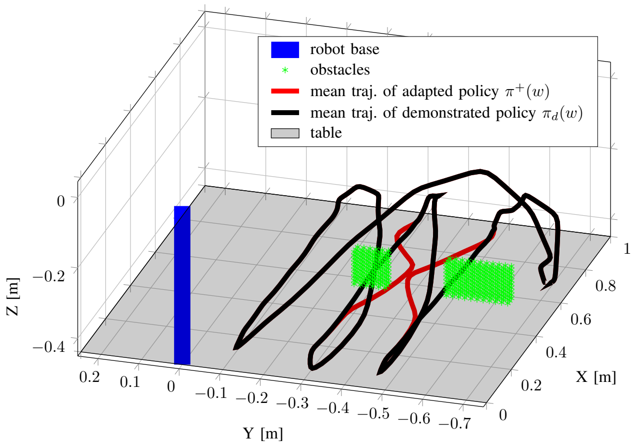
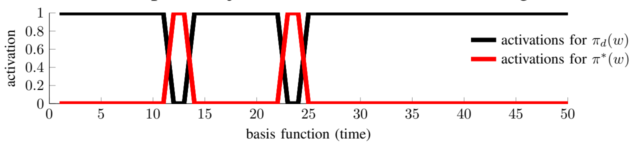


Fig. 10: KL-Divergence $\mathcal{D}(\pi^*(w)||\pi_d(w))$ during optimization in the broken joint case of the Hole-Reaching task. The KL-Divergence of the bounded policy case $\gamma = 50$ is lower than the one of the unbounded policy.

a wide range of the trajectory distribution needs to be changed to successfully avoid the obstacle. In this instance of the Table-Cleaning task only a limited region of the trajectory distribution needs to be changed in order to avoid the obstacle. During the optimization this is learned by the sub-policy $\lambda(a)$, such that the optimization can locally



(a) Task-space trajectories of the Table-Cleaning task



(b) Learned Activations

Fig. 11: Obstacle Avoidance in the Table-Cleaning task.
a): The adapted policy $\pi^+(w)$ locally avoids the obstacles and exactly matches the demonstrated policy $\pi_d(w)$ when possible.
b): Policy $\pi^*(w)$ is only activated in regions where obstacle avoidance is necessary.

concentrate on adapting the important regions. Hence, for the solution $\pi^+(w)$ most of the parts of difficult to estimate covariance matrix Σ_w are not considered. Therefore, valid solutions can still be found even if Σ_w is biased.

V. Conclusion

In this paper, we presented an approach for adapting robot skills to new situations. Our approach uses of three different aspects. First, we simultaneously optimize the expected reward over two sub-policies, instead of one single policy. Second, we explicitly bind the KL-Divergence of one of them to maintain humanlike motion characteristics. And third, we learn a linear combination between the primitive learned from demonstrations and the context-adapted primitive to exactly match the demonstrated policy whenever possible. As our results show, we are able to produce trajectories, which are adapted to the new situation and still stay close to the demonstrated trajectories. In the future, we aim to extend our approach to work as well with mixture models, relating to the HiREPS approach [3].

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7-ICT-2013-10) under grant agreement 610878 (3rdHand).

References

[1] A. Abdolmaleki, N. Lau, L. P. Reis, and G. Neumann. Regularized covariance estimation for weighted maximum likelihood policy search methods. In *Humanoid Robots (Humanoids)*, 2015.

[2] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37:286–298, 2007.

[3] C. Daniel, G. Neumann, O. Kroemer, and J. Peters. Hierarchical relative entropy policy search. *Journal of Machine Learning (JMLR)*, accepted.

[4] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2:1–142, 2013.

[5] Anca D. Dragan, Kenton C.T. Lee, and Siddhartha S. Srinivasa. Legibility and predictability of robot motion. In *Proceedings of the 8th ACM/IEEE International Conference on Human-robot Interaction (HRI)*, 2013.

[6] A. Frisoli, C. Loconsole, R. Bartalucci, and M. Bergamasco. A new bounded jerk on-line trajectory planning for mimicking human movements in robot-aided neurorehabilitation. *Robotics and Autonomous Systems*, 61:404–415, 2013.

[7] Y. Huang, M. Mahmudi, and M. Kallmann. Planning humanlike actions in blending spaces. In *International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[8] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25:328–373, 2013.

[9] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *International Conference on Robotics and Automation (ICRA)*, 2011.

[10] J. Kober, E. Oztop, and J. Peters. Reinforcement learning to adjust robot movements to new situations. In *Proceedings of Robotics: Science and Systems (R:SS)*, 2010.

[11] J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[12] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86, 1951.

[13] Andras Gabor Kupcsik, Marc Peter Deisenroth, Jan Peters, and Gerhard Neumann. Data-efficient generalization of robot skills with contextual policy search. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[14] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Dept., Iowa State University, 1998.

[15] Bokman Lim, Syungkwon Ra, and Frank C Park. Movement primitives, principal component analysis, and the efficient generation of natural motions. In *International Conference on Robotics and Automation (ICRA)*, 2005.

[16] A. Paraschos, C. Daniel, J. Peters, and G Neumann. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[17] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *International Conference on Robotics and Automation (ICRA)*, 2009.

[18] J. Peters, K. Muelling, and Y. Altun. Relative entropy policy search. In *Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence (AAAI), Physically Grounded AI Track*, 2010.

[19] Nathan Ratliff, Matthew Zucker, J. Andrew (Drew) Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *International Conference on Robotics and Automation (ICRA)*, 2009.

[20] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3:233–242, 1999.

[21] Gu Ye and Ron Alterovitz. Demonstration-guided motion planning. In *Proceedings of International Symposium on Robotics Research (ISRR)*, 2011.

Appendix

In the following we give a detailed derivation of our approach: The problem statement in Equation 3 transforms into the lagrangian L with lagrangian multipliers $\eta_1, \eta_2, \alpha_1, \alpha_2$ as

$$\begin{aligned} L = J &- \eta_1 \left[\int_w \pi(w) \log \left(\frac{\pi(w)}{q_w(w)} \right) \mathrm{d}w - \epsilon_1 \right] \\ &- \eta_2 \left[\int_a \lambda(a) \log \left(\frac{\lambda(a)}{q_a(a)} \right) \mathrm{d}a - \epsilon_2 \right] \\ &- \alpha_1 \left[\int_w \pi(w) \mathrm{d}w - 1 \right] - \alpha_2 \left[\int_a \lambda(a) \mathrm{d}a - 1 \right]. \end{aligned} \tag{11}$$

Deriving $\frac{\mathrm{d}L}{\mathrm{d}\pi(w)}$, $\frac{\mathrm{d}L}{\mathrm{d}\lambda(a)}$ and setting both to zero, results in

$$\begin{aligned} \frac{\mathrm{d}L}{\mathrm{d}\pi_(w)} &= R(w,\kappa) - \gamma \log \left(\frac{\pi(w)}{\pi_d(w)} \right) - \gamma \\ &\quad - \eta_1 \log \left(\frac{\pi(w)}{q_w(w)} \right) - \eta_1 - \alpha_1 \overset{!}{=} 0, \qquad (12) \\ \frac{\mathrm{d}L}{\mathrm{d}\lambda(a)} &= R(a,\kappa) - \eta_2 \log \left(\frac{\lambda(a)}{q_a(a)} \right) - \eta_2 - \alpha_2 \overset{!}{=} 0. \end{aligned}$$

Solving for the optimal policies $\pi^*(w)$ and $\lambda^*(a)$, we get

$$\begin{aligned} \pi^*(w) &= \left[e^{R(w)} d(w)^\gamma q_w(w)^{\eta_1} \right]^{\frac{1}{\gamma+\eta_1}} e^{\frac{1}{\gamma+\eta_1}(-\gamma-\eta_1-\alpha_1)}, \\ \lambda^*(a) &= q_a(a) e^{\frac{R(a)}{\eta_2}} e^{\frac{1}{\eta_2}(-\eta_2-\alpha_2)}. \end{aligned} \tag{13}$$

Using the constraint that, $\int_w \pi^*(w)\mathrm{d}w = 1$ and $\int_a \lambda^*(a)\mathrm{d}a = 1$, we can solve Equation 13 for the terms $e^{\frac{1}{\gamma+\eta_1}(-\gamma-\eta_1-\alpha_1)}$ and $e^{\frac{1}{\eta_2}(-\eta_2-\alpha_2)}$. Replacing these terms in Equation 13 yields the policy update rule (Equation 4). The dual is derived by plugging both optimal sub-policies (Equation 13) into the Lagrangian (Equation 11). Simplifying the dual yields Equation 6.