# Latent Space Policy Search for Robotics

Kevin Sebastian Luck[1], Gerhard Neumann[1], Erik Berger[2], Jan Peters[1,4] and Heni Ben Amor[3]

*Abstract*— **Learning motor skills for robots is a hard task. In particular, a high number of degrees-of-freedom in the robot can pose serious challenges to existing reinforcement learning methods, since it leads to a high-dimensional search space. However, complex robots are often intrinsically redundant systems and, therefore, can be controlled using a latent manifold of much smaller dimensionality. In this paper, we present a novel policy search method that performs efficient reinforcement learning by uncovering the low-dimensional latent space of actuator redundancies. In contrast to previous attempts at combining reinforcement learning and dimensionality reduction, our approach does not perform dimensionality reduction as a preprocessing step but naturally combines it with policy search. Our evaluations show that the new approach outperforms existing algorithms for learning motor skills with high-dimensional robots.**

## I. INTRODUCTION

Creating autonomous robots that can adapt to the current task by interacting with their environment is an important vision of artificial intelligence. In recent years, many successful applications of reinforcement learning (RL) to complex robot tasks have been reported, including autonomous helicopter flight [1], robot table-tennis [2], or quadruped locomotion [3]. One of the most successful methods for learning such motor tasks is *policy search* [4].

Policy search tries to directly uncover the parameters of a given policy representation that yield high rewards. In this paper we focus on policy search for robots with a high number of degrees-of-freedom (DOF). Typically, the number of parameters of our control policy heavily depends on the number of DOFs of the robot. Hence, we generally need a large number of evaluations to learn acceptable policies. However, evaluating hundred thousands of different policies on a real robot is often infeasible due to wear and tear, the required logistics, or space and time constraints. At the same time, many robot control tasks, such as motor skills, are highly redundant in the controlled DOFs. Typically, the intrinsic dimensionality of such movements is much smaller than the actual controlled number of DOFs. Hence, robot learning can be performed much more efficiently if we can determine the lower-dimensional latent space of the movement we want to learn.

Fig. 1: A NAO robot learns to lift up one leg and stay balanced using a novel latent space policy search method. The co-articulation of the joints, needed for successful execution of the motor skill, is represented in the low-dimensional latent space.

In this paper we present an efficient policy search algorithm for learning policies in low-dimensional latent spaces. The learning algorithm produces control signals for high-dimensional robot systems by estimating policies in a latent space with a significantly lower number of dimensions. The latent space encodes correlations between the controlled DOFs of the robot. The parameters of the policy as well as the projection parameters of the latent space are efficiently estimated from samples during the policy search iterations. The key insight to our algorithm is that policy search as well as dimensionality reduction can be integrated in an expectation-maximization (EM) framework. As a result,

[1]Kevin S. Luck, Gerhard Neuman and Jan Peters are with the Department of Computer Science, Technische Universität Darmstadt, 64289 Darmstadt, Germany
{luck, geri, peters}@ias.tu-darmstadt.de
[2]Erik Berger is with the Department of Mathematics and Computer Science, Technische Universität Bergakademie Freiberg, 09599 Freiberg, Germany
Erik.Berger@informatik.tu-freiberg.de
[3]Heni Ben Amor is with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, GA 30332, USA
hbenamor@cc.gatech.edu
[4]Jan Peters is with the Max Planck Institute for Intelligent Systems, 72076 Tübingen, Germany

we can formulate a coherent algorithmic approach that naturally combines policy search and dimensionality reduction.

In contrast to previous attempts for combining reinforcement learning and dimensionality reduction for robotic applications, our approach does not perform dimensionality reduction as a preprocessing step. Instead, the parameters of the latent space are adapted based on the reward signal from the environment.

## II. Related Work

Policy search has attracted considerable attention in the robot learning community. An excellent overview of the topic and detailed descriptions of various state-of-the-art algorithms can be found in [5] and [4].

Previous combinations of dimensionality reduction and policy search, typically use a clear separation between the reinforcement learning algorithm and the dimensionality reduction step. In [6], data from a simulator was used in a preprocessing step to identify a possible low-dimensional latent space of policies using Reduced Rank Regression. Learning on the real robot was then restricted to the extracted latent space. Similarly, Bitzer et al. [7] used user-provided training data to learn a low-dimensional subspace using linear and non-linear dimensionality reduction for robot learning. Using dimensionality reduction as a preprocessing step, or as an independent process that can be executed after several iterations of reinforcement learning, may lead to serious limitations. First, extracting the latent space as a preprocess requires a significantly large training set of (approximate) solutions, prior simulations, or human demonstrations. Even if such data is available, it can be counterproductive to use it, since the reinforcement learning algorithm cannot change the parameters of the latent space in these approaches. For example, when using human demonstrations, e.g., recorded joint configurations, to identify the latent space, the extracted latent space might not be appropriate for controlling the robot as we neglect the *correspondence problem* [8], i.e., there is no one-to-one mapping of the human joints to the robot joints. Hence, we need to adapt the projection of the latent space during the reinforcement learning process. Using dimensionality reduction as an independent process also leads to a decreased learning efficiency, since it neglects reward information when identifying subspaces.

## III. Policy Search

In the following section, we will introduce the general problem statement for reinforcement learning and dimensionality reduction and introduce the notation that will be used throughout the paper. For a more detailed description of theses topics, the reader is referred to [9] and [10].

### A. Problem Statement

Reinforcement learning methods can be used to autonomously learn robot control strategies through the interaction with an environment. Given the current state $\mathbf{s}_t \in \mathcal{S}$ a robot executes an action $\mathbf{a}_t \in \mathcal{A}$, transitions into the state $\mathbf{s}_{t+1}$ and receives a reward $r_t(\mathbf{s}_t, \mathbf{a}_t)$. The action selection process is governed by the control policy $\pi(\mathbf{a}_t|\mathbf{s}_t, t)$, which is specified as conditional probability distribution over the actions given the current state $\mathbf{s}_t$. Generally, RL algorithms try to determine an optimal policy which maximizes the expected reward.

In this paper, we will focus on policy search methods. Policy search approaches typically use a parametrized stochastic policy represented by a $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t, t)$ with parameters $\theta$. A typical representation of the policy in robotics is to use a Gaussian distribution as policy where the mean depends linearly on an observed *feature vector* $\boldsymbol{\phi}$ of the task, e.g., the location of an object to grasp. The goal of learning is to optimize the *expected return* of the policy with parameters $\theta$ with

$$J(\theta) = \int_{\mathbb{T}} p_\theta(\tau) R(\tau) \, \mathrm{d}\tau, \tag{1}$$

where the expectation integrates over all possible trajectories $\tau$ in the set $\mathbb{T}$. Each trajectory $\tau = [\mathbf{s}_{1:T+1}, \mathbf{a}_{1:T}]$ is specified by a sequence of length $T$ of states and actions. The return $R(\tau)$ of a trajectory is defined as the accumulated immediate rewards $r_t$, i.e.,

$$R(\tau) = \sum_{t=1}^{T} r_t(\mathbf{s}_t, \mathbf{a}_t) + r_{T+1}(s_{T+1}), \tag{2}$$

where $r_{T+1}$ denotes the final reward for reaching state $s_{T+1}$. Note that in many robot applications, the reward function and the policy are explicitly modelled to be time dependent. Due to the Markov property, the trajectory distribution $p_\theta(\tau)$ can be written as

$$p_\theta(\tau) = p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \pi_\theta(\mathbf{a}_t|\mathbf{s}_t, t). \tag{3}$$

Reinforcement learning algorithms try to determine policy parameters $\theta$ that maximize Equation 1.

### B. Expectation Maximization Approaches to Policy Search

In contrast to traditional approaches to reinforcement learning, EM-based methods formalize the policy search problem as inference problem with latent variables. They transform the rewards into an improper probability distribution such that the reward can be interpreted as (unnormalized) probability of a binary reward event. In our discussion, we will assume that the rewards have already been transformed to such a improper probability distribution, i.e., the rewards are non-negative. As in the standard EM-algorithm, we can now optimize a lower bound, that is in this case a lower bound on the expected return, instead of optimizing the original objective. According to Kober and Peters [11], the lower bound of the expected return (1) is given by

$$L_{\theta_{\text{old}}}(\theta) = \int_{\mathbb{T}} p_{\theta_{\text{old}}}(\tau)\, R(\tau) \log p_{\theta}(\tau)\, \mathrm{d}\tau$$
$$= \mathbb{E}_{p_{\theta_{\text{old}}}(\tau)}\left[\sum_{t=1}^{T} Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t, t) \log \pi_{\theta^*}(\mathbf{a}_t|\mathbf{s}_t, t)\right], \tag{4}$$

where $Q^{\pi}$ is defined as the expected reward to come for time step $t$, when the robot is in state $\boldsymbol{s}_t$ and execute action $\boldsymbol{a}_t$,

$$Q^{\pi}(\mathbf{s}, \mathbf{a}, t) = \mathbb{E}\left[\sum_{\tilde{t}=t}^{T} r_{\tilde{t}}(\mathbf{s}_{\tilde{t}}, \mathbf{a}_{\tilde{t}})\, |\mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}\right]. \tag{5}$$

In practice, $Q^{\pi}(\mathbf{s}, \mathbf{a}, t)$ is estimated by a single rollout, i.e, $Q^{\pi}\left(\mathbf{s}_t^{[i]}, \mathbf{a}_t^{[i]}, t\right) \approx \sum_{\tilde{t}=t}^{T} r_{\tilde{t}}^{[i]}$, where $i$ denotes the index of the episode.

An important advantage of this approach is that the policy update is formulated as a weighted maximum likelihood (ML) estimate for the parameters $\boldsymbol{\theta}$, where the reward to come $Q^{\pi}(\mathbf{s}, \mathbf{a}, t)$ is used as weight for the samples. Due to the weighted ML update, there is no need for a user-specified *learning rate* which is often a critical factor for achieving good performance in policy gradient algorithms [12]. The policy is typically modelled as linear policy with Gaussian noise. In the PoWER [11] algorithm, this Gaussian noise is added to the parameter vector of the policy, i.e.,

$$\mathbf{a} = (\mathbf{M} + \mathbf{E})\, \boldsymbol{\phi}. \tag{6}$$

$\mathbf{M}\boldsymbol{\phi}$ is the mean of the policy and $\mathbf{E}\boldsymbol{\phi}$ denotes a Gaussian noise term that is either isotropic or anisotropically distributed. In our experiments, we will use the more commonly used isotropic version of the noise. In contrast to the standard formulation of PoWER [11], we use *matrix-variate normal distributions* [13] for the exploration noise $\mathbf{E} \sim \mathcal{N}_{d,p}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$, where $\mathbf{0}$ has $d$ rows and $p$ columns. We will use the notation $\mathcal{N}_{d,p}(\cdot,\cdot)$ for such matrix-variate normal distributions and $\mathcal{N}(\cdot,\cdot)$ for multi-variate normal distributions.

In the remainder of this paper, we will write the stochastic policy $\pi_{\theta}(\mathbf{a}_t|\mathbf{s}_t, t)$ as $p_{\theta}(\mathbf{a}_t|\mathbf{s}_t, t)$ to ensure consistent notation.

### C. Using Structured Policies with Latent Variables

Another important advantage of weighted ML updates, is that we can use structured policy representations that again include latent variables $\mathbf{z}$. For example, mixture models [14] or low-dimensional factor models can be used. In our specific case, the latent variable defines the exploration of the policy in a lower dimensional latent space. This low-dimensional exploration $\mathbf{z}$ is then projected in to the high-dimensional original space by a projection matrix. In order to infer such a model with latent variables, we can again use the expectation maximization algorithm. This time we infer a structured policy from the weighted data points. More specifically we use the marginalization rule [15] to introduce a hidden variable $\mathbf{z}$ to our policy by specifying that $p_{\theta}(\mathbf{a}_t|\mathbf{s}_t, t) = \int_{\mathbb{Z}} p_{\theta}(\mathbf{a}_t, \mathbf{z}|\mathbf{s}_t, t)\, \mathrm{d}\mathbf{z}$. This step leads to a new lower bound given by

$$\mathbb{E}_{p_{\theta_{\text{old}}}(\tau)}\left[\sum_{t=1}^{T} Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t, t) \log \int_{\mathbb{Z}} p_{\theta}(\mathbf{a}_t, \mathbf{z}|\mathbf{s}_t, t)\, \mathrm{d}\mathbf{z}\right] \geq$$
$$\mathbb{E}_{p_{\theta_{\text{old}}}(\tau)}\left[\sum_{t=1}^{T} Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t, t)\, \mathbb{E}_{q(\mathbf{z}|\mathbf{a}_t, \mathbf{s}_t)}\left[\log p_{\theta}(\mathbf{a}_t, \mathbf{z}|\mathbf{s}_t, t)\right]\right], \tag{7}$$

where the distribution $q(\mathbf{z}|\mathbf{a}_t, \mathbf{s}_t) = \frac{p_{\theta_{\text{old}}}(\mathbf{a}_t, \mathbf{z}|\mathbf{s}_t, t)}{p_{\theta_{\text{old}}}(\mathbf{a}_t|\mathbf{s}_t, t)}$ is given by the posterior of the latent variables given the old policy parameters $\boldsymbol{\theta}_{\text{old}}$. In this lower bound, the EM-algorithm is applied twice. First, to derive the policy update by weighted maximum likelihood estimates. Second, we use EM to update the joint distribution $p_{\theta}(\mathbf{a}_t, \mathbf{z}|\mathbf{s}_t, t)$ instead of the marginal.

While this lower bound can be used for any latent variable model, we will discuss our specific case of estimating projection parameters in more detail in the following section.

## IV. The PePP$_{\text{C}}$Er Algorithm

In this section, we will describe the "**P**olicy **Se**arch with **P**robabilistic **P**rinciple **C**omponent **E**xplo**r**ation" Algorithm (PePP$_{\text{C}}$Er) for policy search in low-dimensional latent spaces. We will first start with a short recap of Probabilistic PCA, explain the relevant probability distributions for the PePP$_{\text{C}}$Er algorithm and derive the EM update equations.

### A. Revisiting Probabilistic PCA

Probabilistic Principal Component Analysis (PPCA) is the probabilistic formulation of the PCA algorithm for performing linear dimensionality reduction. PPCA relates a $d$-dimensional data point $\mathbf{x} \in \mathbb{R}^d$ to a low-dimensional latent variable $\mathbf{z} \in \mathbb{R}^n$ through a linear Gaussian model

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{8}$$

where the latent variable $\mathbf{z} \in \mathbb{R}^n$ is Gaussian distributed according to $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The transformation matrix $\mathbf{W} \in \mathbb{R}^{d \times n}$ maps each low-dimensional vector $\mathbf{z}$ to the high dimensional space. The matrix $\mathbf{W}$ spans a low-dimensional subspace and $\boldsymbol{\mu} \in \mathbb{R}^d$ is the mean of the high-dimensional distribution. A high dimensional isotropic noise $\boldsymbol{\epsilon} \in \mathbb{R}^d$ with zero mean and $\sigma^2\mathbf{I}$ variance is added to this projection. The parameters of this model are given by $\boldsymbol{\mu}, \sigma^2$ and $\mathbf{W}$ and can efficiently be estimated using an EM algorithm (see [10] for details). However, PPCA is a unsupervised learning method while policy search is supervised.

### *B. Deriving the Update Equations for PePPEr*

Building on the insights from PPCA, we can decompose a stochastic policy into a low-dimensional distribution and projection parameters for generating the required high-dimensional action. More specifically, we can write

$$\mathbf{a} = \mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) + \mathbf{M}\boldsymbol{\phi} + \mathbf{E}\boldsymbol{\phi}, \tag{9}$$

where $\mathbf{W}$ is a projection matrix. The terms $\mathbf{M}\boldsymbol{\phi}$ and $\mathbf{E}\boldsymbol{\phi}$ are again the mean and the Gaussian noise term. The term $\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}$ with $\mathbf{Z} \sim \mathcal{N}_{p,n}(\mathbf{0}, \mathbf{I})$ generates an exploration noise in a low-dimensional latent space, which is then projected into the high-dimensional space of actions via $\mathbf{W}$. Due to the projection from the latent space to the original high dimensional state, the uncorrelated explorative action from the latent space becomes a correlated action in the high dimensional space. Hence, the projection matrix $\mathbf{W}$ can be understood as a matrix that defines synergies in the action space that are used for correlated exploration. Both, the mean $\mathbf{M}$ of the policy and the projection matrix $\mathbf{W}$ are learned by the policy search algorithm. Given the model in Equation 9, we can derive the expectation of our probability distribution $p(\mathbf{a})$ in a straight-forward fashion

$$\mathbb{E}[\mathbf{a}] = \underbrace{\mathbb{E}\left[\mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)\right]}_{0} + \mathbf{M}\boldsymbol{\phi} + \underbrace{\mathbb{E}[\mathbf{E}\boldsymbol{\phi}]}_{0} = \mathbf{M}\boldsymbol{\phi}. \tag{10}$$

Similarly, we can also use the properties of matrix-variate normal distributions [13] to get the covariance

$$\begin{aligned}\operatorname{cov}(\mathbf{a}) &= \mathbb{E}\left[(\mathbf{a} - \mathbb{E}[\mathbf{a}])(\mathbf{a} - \mathbb{E}[\mathbf{a}])^{\mathrm{T}}\right] \\ &= \mathbb{E}\left[\mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\mathbf{W}^{\mathrm{T}}\right] + \mathbb{E}\left[\mathbf{E}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{E}^{\mathrm{T}}\right] \\ &= \operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\left(\mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}\right),\end{aligned} \tag{11}$$

where $\operatorname{tr}(\cdot)$ denotes the trace of a matrix. From Equation 10 and Equation 11 it follows that the prior distribution over actions is

$$\mathrm{p}(\mathbf{a}) = \mathcal{N}\left(\mathbf{M}\boldsymbol{\phi}, \operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\left(\mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}\right)\right). \tag{12}$$

Now, in order to apply EM, we have to determine the posterior distribution $p(\mathbf{Z}|\mathbf{a})$ over matrices $\mathbf{Z}$. The posterior distribution can be simplified by treating $\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}$ as a latent variable. Since the result of this product is a vector, we can use Bayes theorem for Gaussian variables [15, p.93] to derive the posterior distribution $\mathrm{p}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}\right)$. Given both distributions

$$\mathrm{p}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) = \mathcal{N}\left(\mathbf{0}, \operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbf{I}\right) \tag{13}$$

and

$$\mathrm{p}\left(\mathbf{a}|\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) = \mathcal{N}\left(\mathbf{W}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right) + \mathbf{M}\boldsymbol{\phi}, \sigma^2\operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\mathbf{I}\right), \tag{14}$$

the posterior distribution can be written as

$$\mathrm{p}_{\theta_{\mathrm{old}}}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}\right) = \mathcal{N}\left(\mathbf{C}\mathbf{W}^{\mathrm{T}}(\mathbf{a} - \mathbf{M}\boldsymbol{\phi}), \mathbf{C}\sigma^2\operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\right), \tag{15}$$

where $\mathbf{C} = \left(\sigma^2\mathbf{I} + \mathbf{W}^{\mathrm{T}}\mathbf{W}\right)^{-1}$. Given this posterior distribution, we can now determine the equations of the expectation step

$$\mathbb{E}_{p_{\theta_{\mathrm{old}}}(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a})}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right] = \mathbf{C}\mathbf{W}^{\mathrm{T}}(\mathbf{a} - \mathbf{M}\boldsymbol{\phi}), \tag{16}$$

$$\begin{aligned}\mathbb{E}_{p_{\theta_{\mathrm{old}}}(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a})}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right] &= \mathbf{C}\sigma^2\operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right) \\ &+ \mathbb{E}_{p_{\theta_{\mathrm{old}}}(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a})}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]\mathbb{E}_{p_{\theta_{\mathrm{old}}}(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a})}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]^{\mathrm{T}}.\end{aligned} \tag{17}$$

#### *1. Maximization Step for* $\mathbf{M}$

We use a maximum likelihood estimate to identify the value of $\mathbf{M}$ in each iteration. To this end, we calculate the derivative of the log-likelihood function w.r.t. $\mathbf{M}$,

$$\frac{\partial \ln \mathrm{p}(\mathbf{a})}{\partial \mathbf{M}} = \left(\mathbf{D}^{-1}\left(\mathbf{a}\boldsymbol{\phi}^{\mathrm{T}} - \mathbf{M}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\right), \tag{18}$$

where $\mathbf{D} = \operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\left(\mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}\right) = \mathbf{D}^{\mathrm{T}}$. After inserting this result into the EM policy search framework and set the derivative to zero, we get

$$\begin{aligned} 0 &= \mathbb{E}_{p_{\theta_{\mathrm{old}}}(\boldsymbol{\tau})}\left[\sum_{t=1}^{T}\frac{\partial \ln \mathrm{p}(\mathbf{a}_t)\, Q_t^{\pi}}{\partial \mathbf{M}}\right] \\ \Leftrightarrow \mathbf{M} &= \mathbb{E}_{p_{\theta_{\mathrm{old}}}(\boldsymbol{\tau})}\left[\sum_{t=1}^{T}\frac{\mathbf{a}_t\boldsymbol{\phi}^{\mathrm{T}}Q_t^{\pi}}{\operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\right]\left(\mathbb{E}_{p_{\theta_{\mathrm{old}}}(\boldsymbol{\tau})}\left[\sum_{t=1}^{T}\frac{\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}Q_t^{\pi}}{\operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\right]\right)^{-1}\end{aligned} \tag{19}$$

such that $\mathbf{M}$ maximizes the log-likelihood function $\ln \mathrm{p}(\mathbf{a})$.

#### *2. Maximization Step for* $\mathbf{W}$

For optimizing $\mathbf{W}$ we have to use the new lower bound given in Equation 7 and set the derivative of this term w.r.t $\mathbf{W}$ to zero. Accordingly the derivative can be written as

$$\begin{aligned}\frac{\partial \ln \mathrm{p}\left(\mathbf{a}, \mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)}{\partial \mathbf{W}} &= -\left(\sigma^2\operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\right)^{-1} \\ &\left(-\mathbf{a}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}} + \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}} + \mathbf{M}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right)\end{aligned} \tag{20}$$

from which follows that the optimal value of $\mathbf{W}$ that maximizes the log-likelihood is given by

$$\begin{aligned}\mathbf{W} = &\,\mathbb{E}_{p_{\theta_{\mathrm{old}}}(\boldsymbol{\tau})}\left[\sum_{t=1}^{T}\frac{(\mathbf{a}_t - \mathbf{M}\boldsymbol{\phi})\,\mathbb{E}_{p_{\theta_{\mathrm{old}}}(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a})}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]^{\mathrm{T}}Q_t^{\pi}}{\operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\right] \\ &\left(\mathbb{E}_{p_{\theta_{\mathrm{old}}}(\boldsymbol{\tau})}\left[\sum_{t=1}^{T}\frac{\mathbb{E}_{p_{\theta_{\mathrm{old}}}(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t)}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right]Q_t^{\pi}}{\operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)}\right]\right)^{-1}.\end{aligned} \tag{21}$$

*3. Maximization Step for $\sigma^2$*

Similarly to the estimation of $\mathbf{W}$, we can also derivate the log-likelihood of $\ln \mathrm{p}\left(\mathbf{a}, \mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)$ with respect to $\sigma^2$ in order to identify a new estimate of $\sigma^2$ with

$$\begin{aligned}\frac{\partial \ln \mathrm{p}\left(\mathbf{a}, \mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)}{\partial\sigma^2} &= -\frac{d}{2\sigma^2} + \left(2\left(\sigma^2\right)^2 \operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\right)^{-1} \\ &\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\left(\mathbf{a} - \mathbf{W}\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} - \mathbf{M}\boldsymbol{\phi}\right). \quad (22)\end{aligned}$$

Setting the above derivative to zero leads to the following maximum-likelihood estimate of the variance:

$$\begin{aligned}\sigma^2 = \frac{1}{d}\mathbb{E}_{p_{\theta_{\text{old}}}(\boldsymbol{\tau})}\Bigg[&\sum_{t=1}^{T}\left(\operatorname{tr}\left(\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)\right)^{-1} \\ &\Big(\left(\mathbf{a}_t - \mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\left(\mathbf{a}_t - \mathbf{M}\boldsymbol{\phi}\right) \\ &-2\left(\mathbf{a}_t - \mathbf{M}\boldsymbol{\phi}\right)^{\mathrm{T}}\mathbf{W}\mathbb{E}_{p_{\theta_{\text{old}}}(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t)}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right] \\ &+\operatorname{tr}\left(\mathbb{E}_{p_{\theta_{\text{old}}}(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t)}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\mathbf{Z}\right]\mathbf{W}^{\mathrm{T}}\mathbf{W}\right)\Big)Q_t^{\pi}\Bigg] \\ &\left(\mathbb{E}_{p_{\theta_{\text{old}}}(\boldsymbol{\tau})}\left[\sum_{t=1}^{T}Q_t^{\pi}\right]\right)^{-1}. \quad (23)\end{aligned}$$

## C. Complete Algorithm

The resulting algorithm that implements all of the above steps can be found in Alg. 1. The initial values for the parameters $\sigma^2, \mathbf{W}$ and $\mathbf{M}$ can either be randomly chosen or initialized using a PPCA on a set of demonstrations. Additionally, the algorithm requires the number of latent dimensions $n$ as input. After convergence, a policy is given by a weight matrix $\mathbf{M}$ which is multiplied by the feature vector $\boldsymbol{\phi}(\mathbf{s}, t)$ to receive an action for a given state and time.

```
Input: Initialized parameters $\sigma_0^2, \mathbf{W}_0$ and $\mathbf{M}_0$ and the
       dimensionality $n$ of the low dimensional
       manifold. The function $\boldsymbol{\phi}(\mathbf{s}_t, t)$ represents the
       feature vector for the policy.
---------------------------------------------------------------
repeat
    Sampling:
    for $h=1:H$ do # Sample the H rollouts
        for $t=1:T$ do
            $\mathbf{a}_t^h = \mathbf{W}_i\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi} + \mathbf{M}_i\boldsymbol{\phi} + \mathbf{E}\boldsymbol{\phi}$
              with $\mathbf{Z} \sim \mathcal{N}_{p,n}(\mathbf{0}, \mathbf{I})$ and $\mathbf{E} \sim \mathcal{N}_{d,p}\left(\mathbf{0}, \sigma_i^2\mathbf{I}\right)$
            Execute action $\mathbf{a}_t^h$
            Observe and store reward $r_t\left(\mathbf{s}_t^h, \mathbf{a}_t^h\right)$

    Calculate weights:
    $Q^{\pi}(\mathbf{s}, \mathbf{a}, t) = \mathbb{E}\left[\sum_{\tilde{t}=t}^{T} r_{\tilde{t}}(\mathbf{s}_{\tilde{t}}, \mathbf{a}_{\tilde{t}})\,|\mathbf{s}_t = s, \mathbf{a}_t = \mathbf{a}\right]$

    Expectation:
    foreach $\mathbf{a}_t^h$ do
        Compute $\mathbb{E}_{p_{\theta_{\text{old}}}(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h)}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right]$ with (16).
        Compute $\mathbb{E}_{p_{\theta_{\text{old}}}(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}|\mathbf{a}_t^h)}\left[\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\left(\mathbf{Z}^{\mathrm{T}}\boldsymbol{\phi}\right)^{\mathrm{T}}\right]$ with (17).

    Maximization:
    Compute $\mathbf{M}_{i+1}$ with (19).
    Compute $\mathbf{W}_{i+1}$ with (21).
    Compute $\sigma_{i+1}^2$ with (23).
until $\mathbf{M}_i \approx \mathbf{M}_{i+1}$
---------------------------------------------------------------
Output: Linear weights $\mathbf{M}$ for the feature vector $\boldsymbol{\phi}$.
```

**Algorithm 1:** Policy Search with Probabilistic Principle Component Exploration in the Action Space (PePP$_C$Er)

## V. Experiments

The PePP$_C$Er Algorithm has been evaluated on a simulated and a real-world robot task. In this section, we will describe the experimental setup of these evaluations and present the achieved results in comparison to PoWER and the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [16] algorithm.

### A. Learning Inverse Kinematics

In our first experiment, we will focus on learning inverse kinematics. A range of methods exist for analytically or numerically solving the inverse kinematics problem. However, various researchers have also looked at inverse kinematics from a machine learning point of view [17]. In our experiment, we use a simulated robot with $d$ hinge-joints and $d + 1$ segments. The goal of the simulated robot is to track the position of a sphere that is moving on a circle. Setting $d$ to values higher than two results in a redundant system with more DOF than required to accomplish the task. To learn inverse kinematics, we set the reward function to

$$r_t(\mathbf{s}_t, \mathbf{a}_t) = e^{-\mathcal{D}}, \quad (24)$$

where $\mathcal{D}$ is the distance of the end-effector to the target, when action $\mathbf{a}_t$ is executed. Then, we use PePP$_C$Er to determine a suitable policy for the task. During the optimization process, PePP$_C$Er uncovers the redundancies of the system by determining the low-dimensional latent space of joint angle configurations that lead to touching the target. The latent space models the co-articulation of different links. An example result of a learned policy can be found in Fig. 2. As can be seen in the figure, a 20 linked robot arm successfully tracks the target along a circular path.

We ran the explained setup with different specifications of policy search algorithms resulting in the graph depicted in Fig. 3. The graph depicts the sum

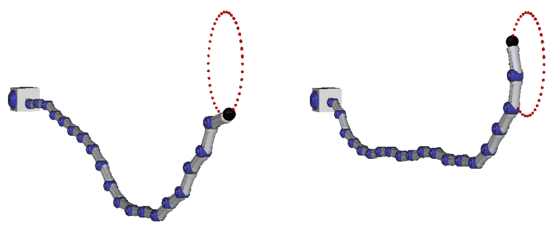

Fig. 2: A tentacle-like robot with 20 links tracks a target along a circular path.

of distances of the end-effector to the target positions. For a balance evaluation, we compared to two different implementations of the PoWER algorithm. In one implementation the $\sigma^2$ was static, while in the other implementation an automatic adaptation of a diagonal covariance matrix was performed. This feature was also implemented in the PePP$_C$Er algorithm, which results in a slightly different update equation for $\sigma^2$. In each iteration 30 samples were drawn and executed on a simulated 20-linked robot. As features we used 19 time-dependent Gaussians, so we had to estimate 380 parameters for 50 time steps. We repeated each experiment 10 times and calculated the mean (bold lines) and standard deviation of the results (light colors around the mean). The figure shows that PePP$_C$Er outperforms CMA-ES and PoWER. In particular in the early iterations both policy search methods perform comparatively well. At the same time, we can see that both PoWER implementations start to stagnate at around 50 iterations. PePP$_C$Er continues to reduce the distance to the targets.

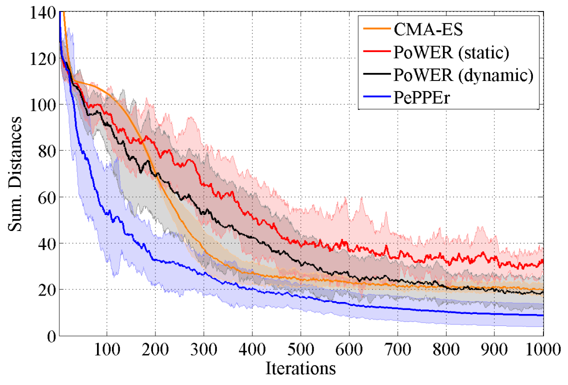

Fig. 3: Comparison between PePP$_C$Er, PoWER and CMA-ES on the inverse kinematics task with a 20-linked robot. In each iteration we executed 30 different joint configurations on the simulated robot. For the static PoWER we set $\sigma = 15$. For the dynamic PoWER and PePP$_C$Er we computed the diagonal covariance matrix.

In the above experiment, the dimensionality of the latent space was set to $n = 5$. In order to evaluate the effect of this parameter on the results, we repeated the evaluation of PePP$_C$Er with varying values for $n$ in an inverse kinematics task for a 12-linked robot, as can be seen in Fig. 4. In the depicted graph, we can see a bump in the average distance at around 5 and 9 dimensions. This is an interesting phenomenon of latent space policy search: too small a value for $n$ restricts the search space, too high a value for $n$ diminishes the effect of dimensionality reduction. In our specific example, the best value for $n$ seems to be 4 or 5.

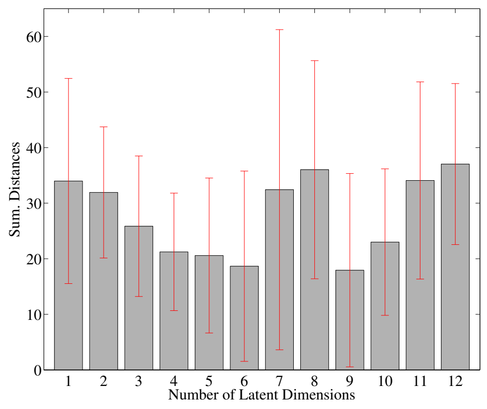

Fig. 4: The mean sum of distances and the standard deviation between the 450th and 500th iteration for an 12-linked robot. Five solutions were learned by PePP$_C$Er for different values of the dimensionality $n$ of the latent space.

### B. *Learning to Stand on One Leg*

We also performed a learning task on a real robot. More specifically, we used PePP$_C$Er to learn policies for standing on one leg. The task of *standing on one leg* is a synergistic motor skill that requires the co-articulation of different body parts for successful execution. It is often used in biomechanical studies on synergies and low-dimensional control in humans, such as in [18]. In our experiment, we set the episodic reward of the robot proportional to the height of the right leg after execution of the policy. Furthermore, we have to consider in our reward function, that the head and the right foot of the robot should not move a lot. Hence, the reward function can be written as

$$R(h, rf, lf) = \exp\{\alpha \cdot h + \beta \cdot rf - \gamma \cdot lf - \lambda_{\text{MAX}}\}, \quad (25)$$

where $\alpha, \beta, \gamma \in \mathbb{R}^+$, $h$ is the height of the head, $rf$ the height of the right foot and $lf$ the height of the left foot in the final position. The constant $\lambda_{\text{MAX}}$ is the maximal possible value of the first part of the sum. The height of the head is responsible for a low reward if the robot falls

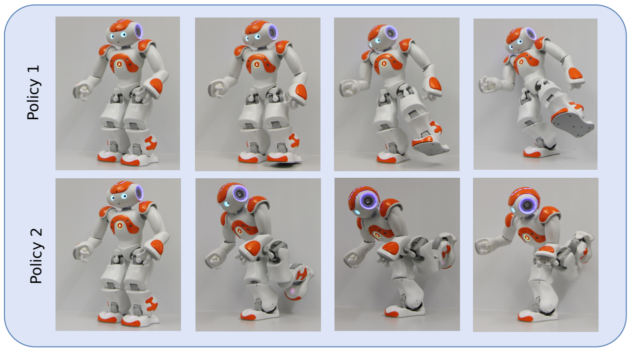

Fig. 5: Two different policies for standing on one leg learned using latent space policy search. Only 100 samples were needed to learn policy 1.

during learning. As features, time-dependent Gaussians were used in this experiment.Actions were represented by the change in the 25 robot joint angles between two consecutive time steps.

The goal in robot learning is to learn from few trials. We therefore restricted the maximum number of samples (executions on the robot) to 600 samples. For automation and repeatability purposes, learning was performed in a physics-based simulator. However, we want to stress that, given the relatively small number of trials needed by PePP$_{\mathrm{C}}$Er to learn a policy, we can also perform learning directly on the real robot. Fig. 5 shows two learned policies acquired using PePP$_{\mathrm{C}}$Er. Learning started from random initializations and did not require any demonstrations. Policy 1 was learned using a sample size of 20 samples and 5 iterations, i.e., 100 execution on the robot in total. We can see, that it results in a smooth and stable motor skill. Policy 2 required 600 evaluations in total and allows the robot to lift the leg even higher.

## VI. CONCLUSIONS

In this paper we presented a novel policy search algorithm for robotics applications. The PePP$_{\mathrm{C}}$Er algorithm determines the correlations between different joints of the robot and uses the information for fast and efficient reinforcement learning. The presented method combines policy search and dimensionality reduction in a natural way and has been derived from basic principles. Applications on a simulated and a real robot indicate that the approach can be employed to learn new motor skills for complex, redundant robots using a relatively small number of trials on the robot. In our future work we want to combine the introduced approach with imitation learning, in order to start in a good region of the search space. Additionally, we want to investigate methods for identifying the dimensionality of the current task.

## Acknowledgment

The research leading to these results has received funding from the European Union under grant agreement #270327 (CompLACS).

## References

[1] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Proceedings of the International Symposium on Experimental Robotics*, 2004, pp. 363–372.

[2] K. Muelling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *International Journal of Robotics Research*, no. 3, pp. 263–279, 2013.

[3] J. Zico Kolter and A. Y. Ng, "The stanford littledog: A learning and rapid replanning approach to quadruped locomotion," *Int. J. Rob. Res.*, vol. 30, no. 2, pp. 150–174, Feb. 2011.

[4] J. Kober and J. Peters, "Reinforcement learning in robotics: a survey," in *Reinforcement Learning*. Springer Berlin Heidelberg, 2012, pp. 579–610.

[5] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, no. 12, pp. 1–142, 2013.

[6] J. Z. Kolter and A. Y. Ng, "Learning omnidirectional path following using dimensionality reduction," in *in Proceedings of Robotics: Science and Systems*, 2007.

[7] S. Bitzer, M. Howard, and S. Vijayakumar, "Using dimensionality reduction to exploit constraints in reinforcement learning," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Oct 2010, pp. 3219–3225.

[8] C. L. Nehaniv and K. Dautenhahn, "Imitation in animals and artifacts," K. Dautenhahn and C. L. Nehaniv, Eds. Cambridge, MA, USA: MIT Press, 2002, ch. The Correspondence Problem, pp. 41–61.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[10] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society, Series B*, vol. 61, pp. 611–622, 1999.

[11] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1-2, pp. 171–203, 2011.

[12] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[13] A. K. Gupta and D. K. Nagar, *Matrix variate distributions*. CRC Press, 2000, vol. 104.

[14] C. Daniel, G. Neumann, and J. Peters, "Learning concurrent motor skills in versatile solution spaces," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 3591–3597.

[15] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer New York, 2006, vol. 1.

[16] N. Hansen, S. Muller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)." *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[17] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 522–531, Aug. 2004.

[18] G. Torres-Oviedo and L. H. Ting, "Subject-specific muscle synergies in human balance control are consistent across different biomechanical contexts," *Journal of neurophysiology*, vol. 103, no. 6, pp. 3084–3098, 2010.