J Intell Robot Syst (2016) 83:393–408
DOI 10.1007/s10846-016-0347-y

Contextual Policy Search for Linear and Nonlinear Generalization of a Humanoid Walking Controller

Abbas Abdolmaleki · Nuno Lau · Luis Paulo Reis · Jan Peters · Gerhard Neumann

Received: 14 June 2015 / Accepted: 27 January 2016 / Published online: 15 February 2016

Abstract We investigate learning of flexible robot locomotion controllers, i.e., the controllers should be applicable for multiple contexts, for example different walking speeds, various slopes of the terrain or other physical properties of the robot. In our experiments, contexts are desired walking linear speed of the gait. Current approaches for learning control parameters of biped locomotion controllers are typically only applicable for a single context. They can be used for a particular context, for example to learn a gait with highest speed, lowest energy consumption or a combination of both. The question of our research is, how can we obtain a flexible walking controller that controls the robot (near) optimally for many different contexts? We achieve the desired flexibility of the controller by applying the recently developed contextual relative entropy policy search(REPS) method which generalizes the robot walking controller for different contexts, where a context is described by a real valued vector. In this paper we also extend the contextual REPS algorithm to learn a non-linear policy instead of a linear policy over the contexts which call it RBF-REPS as it uses Radial Basis Functions. In order to validate our method, we perform three simulation experiments including a walking experiment using a simulated NAO humanoid robot. The robot learns a policy to choose the controller parameters for a continuous set of forward walking speeds.

Keywords Learning humanoids robot locomotions · Generalizing robot skills · Stochastic search · Contextual relative entropy policy search · Nonlinear policies · Nao robot

1 Introduction

In the humanoid robotics research field, one of the major tasks is to design a locomotion controller for the biped robot that be optimally applicable for multiple locomotion tasks such as walking with different

A. Abdolmaleki (✉) · N. Lau
DETI / IEETA, University of Aveiro, Aveiro, Portugal
e-mail: abbas.a@ua.pt

N. Lau
e-mail: nunolau@ua.pt

A. Abdolmaleki · L. P. Reis
DSI, University of Minho, Guimarães, Portugal

L. P. Reis
LIACC, University of Porto, Porto, Portugal
e-mail: lpreis@dsi.uminho.pt

J. Peters
IAS, TU Darmstadt, Darmstadt, Germany
e-mail: peters@ias.tu-darmstadt.de

J. Peters
MPI for Intelligent Systems, Stuttgart, Germany

G. Neumann
CLAS, TU Darmstadt, Darmstadt, Germany
e-mail: neumann@ias.tu-darmstadt.de

desired speeds. We characterize a task with a context vector $\boldsymbol{s}$. The context describes all variables which do not change during the execution of the task but might change from task to task. For example throughout this paper, we refer to context as a combination of following three items:

1. The walking objective such as the walking speed.
2. The physical properties of the robot such as the mass
3. The environmental states such as the friction, slope and gravity.

There have been several parametrized walking controllers proposed in the literature [4, 8, 9, 11, 13, 15] that potentially, with appropriately set parameters, can control the robot for different contexts. One of the key problems is now to find the optimal gait controller parameters for a given context. There are many examples of optimizing the control parameters of biped locomotion controllers, however, these examples usually optimize a parameter set for a single context, for example, optimize a gait for highest speed [3, 21, 22], lowest energy consumption or both [1]. Black box optimization methods such as evolutionary strategies, e.g. CMA-ES [25], have been employed extensively for gait optimization [5–7, 12, 21]. However, these algorithms fail to generalize the optimized gait to different contexts. In order to generalize the gait to, for example, different walking speeds, typically the parameters are optimized for several target contexts independently. Subsequently, regression methods are used to generalize the optimized gaits to a new, unseen context [6, 16]. Although such approaches have been used successfully, they are time consuming, ad-hoc and inefficient in terms of the number of needed training samples as optimizing the controller parameters for different contexts and the generalization between optimized parameters for different contexts, are two independent processes. Hence, we cannot reuse datapoints obtained from optimizing a gait for one context to improve and accelerate the optimisation of a gait context. Given these limitations, for the first time, we investigate the application of contextual policy search [2] to efficiently optimize and generalize the parameters of a walking controller for multiple unseen contexts such as forward walking speeds. In this paper, we present a hierarchical structure of walking controller, that is composed of two main parts, which are a lower level walking policy an upper level policy. We model the lower level walking policy as a ZMP based walking controller that models the dynamics of the robot as an inverted pendulum model augmented with a spring model [17]. The upper level walking policy sets the parameters of the lower level policy for a given context. We use contextual relative entropy policy search (REPS) [2, 20], to learn the upper-level walking policy $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ that chooses the parameters $\boldsymbol{\theta}$ of the lower level walking policy according to the desired input context s. REPS typically learns a linear upper level policy. We extend the algorithm to learn a non-linear policy by using radial basis functions and we call the the algorithm RBF-REPS. We will show that RBF-REPS outperforms standard REPS in our experiments. We also investigate using two different constraint handling methods for learning the upper level policy. Using our proposed structure of the walking controller, we successfully learn an upper level walking policy that generalizes the lower level walking policy for different forward walking speeds. The reminder of the paper is organized as follows. The next section introduces our proposed lower level walking policy. Subsequently we explain the learning method for the upper level policy which sets the parameters of our lower level walking policy given a context. The results and discussions will be presented in the experiments section and finally, we conclude the paper with a conclusion and future work.

2 Parametrized Walking Controller

In this section, we explain the parametrisation of our lower level walking policy which is a ZMP-based omnidirectional walking engine [8, 11, 13, 15].

2.1 Overview of Walking Engine

The walking engine is omnidirectional in the sense that the robot can potentially walk towards any desired direction. The origin of the robot base-frame is on the torso of the robot, with the x-axis pointing forward, y-axis pointing to the left, and z-axis pointing up. Similar to [8], we assume that the position of the center of mass (CoM) and the base-frame are identical. There are different methods to synthesize the walking motion. We use one of the popular approaches that generates Cartesian trajectories for the CoM and the feet of the robot given planned footsteps [8, 11,

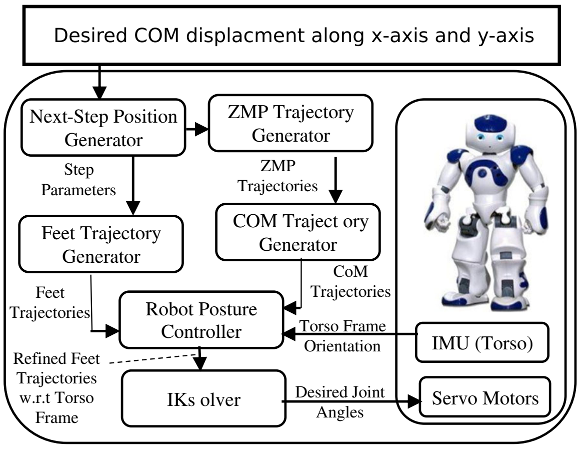


Fig. 1 This sketch shows the modules of the lower level walking policy and their relations

13]. The footstep planner gets as an input the desired CoM displacement along x-axis and y-axis for a single step and using this input, plans the final position of the feet and the final position of the CoM in the Cartesian space. Knowing the initial and end position of the CoM for one single step, the walk engine generates the CoM trajectories such that the zero moment point (ZMP) stability criterion is fulfilled, i.e., the ZMP stays within support polygon.[1] We use Bezier curves to generate the trajectory of the swing foot to reach the computed foot end position. Given the CoM trajectory (Which is same as robot base-frame trajectory in global fram) and the swing foot trajectory (in global frame) we use inverse kinematics to compute the desired joint positions. We also use a posture controller that ideally keeps the trunk of the robot upright to avoid the robot to fall. Finally, we use a PD controller to convert the desired joint positions to proper motor commands. Hence, the lower level walking policy is decomposed in several modules. Figure 1 depicts the role of each module and their interactions. Each module has a few parameters to set which are listed in Table 1. In the following, a brief description of each module is given. Please refer to [17] for details of our omnidirectional walking engine.

Next-Step Position Generator - Based on the desired CoM displacement along x-axis (x_{step}) and along y-axis (y_{step}) it computes the position of the swing foot at the end of the step by taking to account foot reachability and feet collision, please refer to [17] for more details. Please note that for direct walking which is the case of this paper, the (y_step) is zero.

[1]Support polygon is the convex hull of the feet

Table 1 Open parameters of lower level walking policy

Notation	Description
A(m)	Oscillation amplitude of the spring
$x_0(m)$	Rest position of the spring (initial height of the robot)
z_{step}(m)	Maximum height of The Swing Foot From The Ground
T(s)	Duration of the step
r_{offset}(Degree)	Torso roll Offset
γ_{tilt}	Feedback gain for The torso tilt
γ_{roll}	Feedback gain For The torso roll
x_{step}(m)	The CoM Step Size along The desired direction
ϕ_{DSP}(s)	Duration of double support phase
σ_{step}(m/s)	The factor of how fast the CoM step size increases or decreases

Swing Foot Trajectory Generator - This module is responsible to generate a trajectory for the swing foot such that it reaches its next position that is given by Next-Step Position Generator module. The idea is to smoothly move the swing foot from the current position to the final foot step position, ideally with zero lifting and landing speed. Cubic Bezier Curve is used to generate such a desired smooth trajectory. The parameter of this module to set is z_{step} which is the maximum height of the swing foot during the movement.

COM Trajectory Generator - In this module we generate the CoM trajectories for all three Cartesian coordinates i.e., The trajectories along x.axis, y-axis and z-axis. We use a 3D-Inverted pendulum model(IPM) augmented with springs as a simplified model of the robot (Fig. 2) with the ZMP criterion as stability indicator to generate CoM trajectories from its initial position to the calculated end CoM position. The parameters of this module to be set are A, x_0, T and ϕ_{DSP} which will be explained in the Section 2.2. For more detailed explanation refer to [17].

Robot Posture Controller - This module maintains the stability of the robot by controlling the torso to have a reference roll and tilt. In our case the reference tilt of the torso is zero. However we

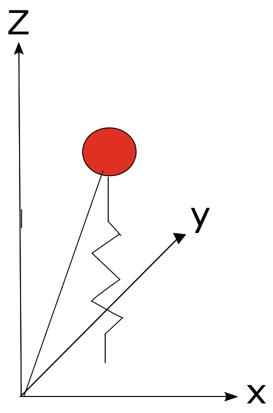


Fig. 2 We use an inverted pendulum model augmented with a spring which generates movements in the $x-z$ and $y-z$ planes

empirically found that if robot be rolled forward to some extend it can be helpful. We still need to find that how much robot should be rolled forward. Therefore the reference roll of the torso r_{offset} is one of the open parameters. This module tries to keep the tilt and roll of the robot torso same as the referenced ones using tilt and roll of the robot from sensors. The parameters γ_{tilt} and γ_{roll} are the other open parameters of this module which take care of correcting for undesired tilt and roll of the torso. Please refer to [24] for details.

Inverse Kinematics Solver - This module computes the reference joint angles based on the relative position of the feet to the torso frame. Please refer to [14] for details.

2.2 CoM Trajectory Generator

To generate the CoM trajectories, we model the robot as a 3D inverted pendulum model augmented with a spring as shown by the Fig. 2. It is a single particle system that uses an undamped spring to model the dynamics of CoM along z-axis. We generate the Cartesian trajectories of the CoM such that Zero Moment Point(ZMP) [10] always stays under the support foot. Once we have the initial and final positions of the feet and CoM, we can define the desired reference ZMP trajectory

$$\begin{bmatrix} x_{ZMP}(t) \\ y_{ZMP}(t) \end{bmatrix}.$$

We choose the reference ZMP to be at the centre of the supporting foot during the single support phase and linearly move towards to the centre of the other foot during double support phase [17]. Assuming the model in Fig. 2 the relationship between the position of the ZMP and the CoM can be obtained by

$$x_{\text{ZMP}} = x_{\text{CoM}} - \frac{\ddot{x}_{\text{CoM}} \cdot z_{\text{CoM}}}{g + \ddot{z}_{\text{CoM}}}, \tag{1}$$

$$y_{\text{ZMP}} = y_{\text{CoM}} - \frac{\ddot{y}_{\text{CoM}} \cdot z_{\text{CoM}}}{g + \ddot{z}_{\text{CoM}}}. \tag{2}$$

The variables z_{CoM} and $\ddot{z}_{\text{CoM}}$ denote the position and acceleration along z-axis. The z_{CoM} and $\ddot{z}_{\text{CoM}}$ trajectories are determined by the dynamics of the spring with the resulting movment

$$z_{\text{CoM}}(t) = x_0 + A\cos(\omega t + \phi), \quad s.t. \quad \omega = \frac{2\pi}{T}, \tag{3}$$

$$\ddot{z}_{\text{CoM}}(t) = -A\omega^2 \cos(\omega t + \phi), \quad s.t. \quad \omega = \frac{2\pi}{T}. \tag{4}$$

Similar to walking, it is a periodic motion with amplitude A which determines the desired amount of CoM displacement along z-axis around spring rest position x_0 which can be interpreted as the initial height of the torso, the period T is the duration of a single oscillation, which has the same duration of a single step and ϕ is the phase of the motion which determines the starting position of z_{CoM} at the beginning of the step. Throughout this paper we set ϕ to 0. The next problem is how to generate the x_{CoM} and y_{CoM} trajectories on-line for one step given the $z_{\text{CoM}}, \ddot{z}_{\text{CoM}}, x_{\text{ZMP}}, y_{\text{ZMP}}$ trajectories and the initial position of the x_0 and terminal position of the CoM x_n for a single step. This problem is the inverse problem of Eqs. 1 and 2. Similar to [13], we use a central difference approximation to discretize the second derivative of?the x_{CoM}, i.e.,

$$\begin{aligned} \ddot{x}_{\text{CoM}}(t) &= \frac{x_{\text{CoM}}(t+\zeta) - 2x_{\text{CoM}}(t) + x_{\text{CoM}}(t-\zeta)}{\zeta^2} \\ &= \frac{x_{\text{CoM}}(k+1) - 2x_{\text{CoM}}(k) + x_{\text{CoM}}(k-1)}{\zeta^2}, \end{aligned} \tag{5}$$

where ζ is the sampling period and $k = 1 \ldots n$ is the index of the time-discretized samples with n total number of samples which depends on sampling period T. On the other word, as we always solve for one step, we can infer that

$$n = \frac{T}{\zeta},$$

where T is duration of one step. Note that we use the same sampling period ζ to discretize the given z_{CoM}, $\ddot{z}_{CoM}$, x_{ZMP}, y_{ZMP} trajectories. Integrating (5) into (1), we obtain

$$\begin{aligned} x_{\mathrm{ZMP}}(k) &= a(k)x_{\mathrm{CoM}}(k-1) + b(k)x_{\mathrm{CoM}}(k) \\ &\quad + c(k)x_{\mathrm{CoM}}(k+1), \\ \text{s.t. } \forall \quad k &= 1 \ldots n \quad c(k) = a(k) = \frac{-z_{\mathrm{CoM}}(k)}{(g - \ddot{z}_{\mathrm{CoM}}(k))\,T^2}, \\ &\qquad a(1) = 0, \\ b(k) &= 1 - 2a(k), \quad c(n) = 0. \end{aligned} \tag{6}$$

This Equation is a tridiagonal system for n unknowns $x_{\mathrm{CoM}}(k)$, $k = 1 \ldots n$. For such a system, the Thomas algorithm [13] can find the solutions efficiently in $O(n)$ operations and can hence be used for real-time gait generation. To solve this system, we first rearrange the coefficients. New coefficients are denoted with primes:

$$c(k)' = \begin{cases} 0, & k = 1 \\ \frac{c(k)}{b(k) - a(k-1)c(k-1)'} & k = 2, 3, \ldots, n-1 \end{cases} \tag{7}$$

$$x_{ZMP}(k)' = \begin{cases} x_0, & k = 1 \\ \frac{x_{ZMP}(k) - a(k)x_{ZMP}(k-1)'}{b(k) - a(k)c(k-1)'} & k = 2, 3, \ldots, n-1 \end{cases} \tag{8}$$

The solution is then obtained by back substitution, i.e.,

$$x_{\mathrm{CoM}}(k)' = \begin{cases} x_n, & k = n, \\ x_{\mathrm{ZMP}}(k)' - c(k)' x_{\mathrm{CoM}}(k+1), & k = n-1, n-2, \ldots, 1. \end{cases} \tag{9}$$

Using the same method, we can obtain the y_{CoM} trajectory. Table 1 lists the open parameters of the lower level policy. In the next section we discuss a method to learn an upper level policy which sets the open parameters given a context (Fig. 3).

3 Learning the Upper Level Policy

Given the lower level policy, we would like to obtain an upper level policy $\pi(\theta|s)$ that sets the parameters θ (see Table 1) of the lower level walking policy given a context s. For this purpose, we use contextual REPS [2, 20]. Contextual REPS is a stochastic search algorithm that maintains a Gaussian search distribution $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ over the parameter vector $\boldsymbol{\theta}$ condition on context $\boldsymbol{s}$, which we denote as upper level policy. This policy is used to create samples $\boldsymbol{\theta}^{[i]}$ (i is the index of the sample) of the parameter vector $\boldsymbol{\theta}$ for a given context sample $\boldsymbol{s}^{[i]}$. Subsequently, the performance $R^{[i]} \sim p(R|\boldsymbol{\theta}^{[i]}, \boldsymbol{s}^{[i]})$ of $[\boldsymbol{s}^{[i]}\,\boldsymbol{\theta}^{[i]}]$ is evaluated. Using the samples $[\boldsymbol{s}^{[i]}\,\boldsymbol{\theta}^{[i]}\,R^{[i]}]$, a weight $d^{[i]}$ for each sample is computed that is used to estimate a new upper-level Gaussian policy by using a weighted

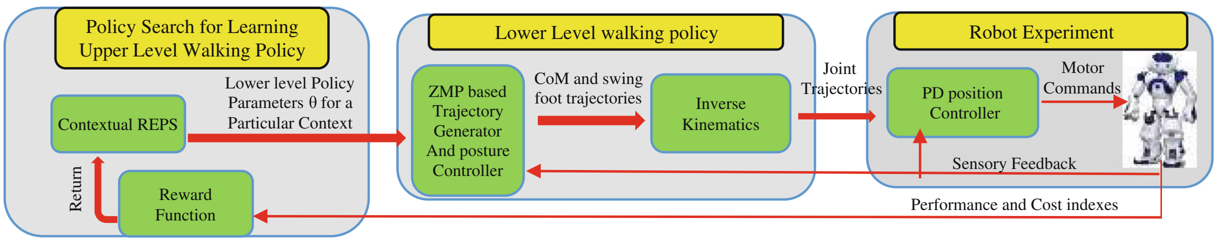

Policy Search for Learning Upper Level Walking Policy
Contextual REPS
Lower level Policy Parameters θ for a Particular Context
Return
Reward Function
Lower Level walking policy
ZMP based Trajectory Generator And posture Controller
CoM and swing foot trajectories
Inverse Kinematics
Joint Trajectories
Robot Experiment
PD position Controller
Motor Commands
Sensory Feedback
Performance and Cost indexes

Fig. 3 High level view of learning an upper level policy for the lower level walking controller. As the workflow shows, the policy search component generates parameters $\boldsymbol{\theta}$ for a particular context. The lower level walking policy uses this parameters to generate the joint trajectories. This joint trajectories are applied on the robot and based on the performance indexes, a reward value is computed. Once enough number of samples are evaluated, policy search algorithm updates the parameters of the upper level policy. This process iterates till algorithm converges to a solution

maximum likelihood (ML) estimate. Algorithm 1 shows a compact representation of the method.

Algorithm 1 Contextual policy search algorithm

```
Repeat
  Generate context samples $\{s^{[i]}\}_{k=1...N}$
    Sample parameters $\{\theta^{[i]}\}_{k=1...N}$ from $\pi(\theta|\boldsymbol{s})$ given context samples $\{s^{[i]}\}_{k=1...N}$
  Evaluate the reward $R^{[i]}$ of each sample in the sample set $\{s^{[i]}, \theta^{[i]}\}_{k=1...N}$
    Use the data set $\{R^{[i]}\}_{k=1...N}$ to compute a weight $d^{[i]}$ for each sample
  Use the data set $\{s^{[i]}, \theta^{[i]}, d^{[i]}\}_{k=1...N}$ to update the upper level policy $\pi(\theta|\boldsymbol{s})$
Until upper level policy $\pi(\theta|\boldsymbol{s})$ converges
```

3.1 Contextual Relative Entropy Policy Search

Contextual REPS is an information theoretic policy search method [2]. The main insight of using information theory is to bound the relative entropy, also called Kullback-Leibler divergence, between two subsequent policies. By limiting the relative entropy, we can control the information loss of the policy update and the policy will keep the information about what areas of the parameter space still need to be explored. Contextual REPS directly searches in the parameter space of the lower level controller by maintaining and updating a stochastic policy which is typically modelled as a multivariate Gaussian distribution. The upper-level policy $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ is conditioned on the context vector $\boldsymbol{s}$, which describes the current task. We now want to optimize the following performance criteria

$$\max_{\pi} \iint \mu(\boldsymbol{s})\pi(\boldsymbol{\theta}|\boldsymbol{s})R_{\boldsymbol{s\theta}}d\boldsymbol{\theta}d\boldsymbol{s},$$
$$\text{s.t.} \int \mu(\boldsymbol{s})\text{KL}\left(\pi(\boldsymbol{\theta}|\boldsymbol{s})||q(\boldsymbol{\theta}|\boldsymbol{s})\right)d\boldsymbol{s} \leq \varepsilon, \ \forall \boldsymbol{s}:\ 1$$
$$= \int \pi(\boldsymbol{\theta}|\boldsymbol{s})d\boldsymbol{\theta}. \qquad (10)$$

where $\mu(\boldsymbol{s})$ denotes the distribution over the context which is specified by the task and $R_{\boldsymbol{s\theta}}$ denotes the expected performance when evaluating parameter vector $\boldsymbol{\theta}$ in context $\boldsymbol{s}$. This optimization problem can be solved efficiently by the method of Lagrangian multipliers [19]. The closed form solution for policy $\pi(\boldsymbol{\theta}|\boldsymbol{s})$ is given by

$$\pi(\boldsymbol{\theta}|\boldsymbol{s}) \propto q(\boldsymbol{\theta}|\boldsymbol{s})\exp\left(R_{\boldsymbol{s\theta}}/\eta\right).$$

where η is a Lagrangian multiplier that sets the temperature of the soft-max distribution given in the previous equation. The temperature parameter η can be found efficiently by optimizing the dual function

$$g(\eta) = \eta\varepsilon + \eta\int \mu(\boldsymbol{s})\log\left(\int q(\boldsymbol{\theta}|\boldsymbol{s})\exp\left(\frac{R_{\boldsymbol{s\theta}}}{\eta}\right)d\boldsymbol{\theta}\right)d\boldsymbol{s}. \qquad (11)$$

The optimal value for η can be obtained by minimizing the dual function $g(\eta)$ such that $\eta > 0$, see [19]. However approximating the log integral in the dual function (Eq. 11) is not feasible as we would need a lot of samples $\boldsymbol{\theta}^i$ for each context $\boldsymbol{s}^i$. In order to alleviate this problem the performance criteria is reformulated. Since contextual REPS use the contexts and parameters samples $[\boldsymbol{\theta}^i, \boldsymbol{s}^i]$, it only has access to the samples from the joint distribution i.e. , $q(s, \boldsymbol{\theta})$. Therefore we optimize for the joint probabilities $p(\boldsymbol{s}, \boldsymbol{\theta})$ instead of for the policy $\pi(\boldsymbol{\theta}|\boldsymbol{s})$. Additionally, we need to enforce that $p(\boldsymbol{s}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{s}, \boldsymbol{\theta})d\boldsymbol{\theta}$ still reproduces the correct context distribution $\mu(\boldsymbol{s})$ by using the constraints $\forall \boldsymbol{s}$: $p(\boldsymbol{s}) = \mu(\boldsymbol{s})$. However now we have an infinite number of constraints. In order to avoid an infinite number of constraints, these constraints are approximately fulfilled by matching feature expectations instead of matching single probabilities, i.e.,

$$\int_{\boldsymbol{s}} p(\boldsymbol{s})\boldsymbol{\phi}(\boldsymbol{s})d\boldsymbol{s} = \hat{\boldsymbol{\phi}}.$$

where $\hat{\boldsymbol{\phi}} = \int_{\boldsymbol{s}} \mu(\boldsymbol{s})\boldsymbol{\phi}(\boldsymbol{s})d\boldsymbol{s}$ is the expected feature vector for the given context distribution $\mu(\boldsymbol{s})$ and a given feature vector $\boldsymbol{\phi}(\boldsymbol{s})$. For example, if the feature vector $\boldsymbol{\phi}(\boldsymbol{s})$ contains all linear and squared terms $[\boldsymbol{s}, \boldsymbol{s}^2]$ of the context vector $\boldsymbol{s}$, these constraints correspond to matching the first and second moment of $p(\boldsymbol{s})$ with the moments of $\mu(\boldsymbol{s})$. The resulting optimization program yields

$$\max_{p} \iint p(\boldsymbol{s}, \boldsymbol{\theta})R_{\boldsymbol{s\theta}}d\boldsymbol{s}d\boldsymbol{\theta} \quad \text{s.t. } \varepsilon \geq \text{KL}(p(\boldsymbol{s}, \boldsymbol{\theta})||\mu(\boldsymbol{s})q(\boldsymbol{\theta}|\boldsymbol{s})),$$
$$\hat{\boldsymbol{\phi}} = \iint p(\boldsymbol{s}, \boldsymbol{\theta})\boldsymbol{\phi}(\boldsymbol{s})d\boldsymbol{s}d\boldsymbol{\theta}, \quad 1 = \iint p(\boldsymbol{s}, \boldsymbol{\theta})d\boldsymbol{s}d\boldsymbol{\theta}. \qquad (12)$$

The solution for $p(\boldsymbol{s}, \boldsymbol{\theta})$ is now given by

$$p(\boldsymbol{s}, \boldsymbol{\theta}) \propto q(\boldsymbol{\theta}|\boldsymbol{s})\mu(\boldsymbol{s})\exp\left((R_{\boldsymbol{s\theta}} - V(\boldsymbol{s}))/\eta\right),$$

Where $V(\boldsymbol{s}) = \boldsymbol{\phi}(\boldsymbol{s})^T \boldsymbol{w}$ is a context dependent baseline which is subtracted from the return $R_{\boldsymbol{s}\boldsymbol{\theta}}$. The parameters $\boldsymbol{w}$ and η are again Lagrangian multipliers that can be obtain by optimizing the dual function, given as

$$g(\eta, \boldsymbol{w}) = \eta\varepsilon + \hat{\boldsymbol{\phi}}^T \boldsymbol{w} + \eta \log \left(\iint \mu(s) q(\boldsymbol{\theta}|\boldsymbol{s}) \times \exp\left(\frac{R_{\boldsymbol{s}\boldsymbol{\theta}} - \boldsymbol{\phi}(\boldsymbol{s})^T \boldsymbol{w}}{\eta} \right) d\boldsymbol{\theta} d\boldsymbol{s} \right). \quad (13)$$

This policy update results in a weight $d^i = p(\boldsymbol{s}^i, \boldsymbol{\theta}^i)$ for each sample $[\boldsymbol{s}^i, \boldsymbol{\theta}^i]$ which we can use to estimate a new upper-level policy $\pi(\theta|s)$ by using a weighted maximum likelihood estimate. In the simplest case, $\pi(\theta|s)$ is modeled as linear Gaussian policy, i.e., $\pi(\theta|s) = \mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{A}^T \varphi(s), \Sigma\right)$, where $\varphi(s)$ is an arbitrary feature function. In this case, the parameters A can be obtained by the weighted maximum likelihood

$$\boldsymbol{A} = (\boldsymbol{\Phi}^T \boldsymbol{D} \Phi)^{-1} \boldsymbol{\Phi}^T DU, \quad (14)$$

where $\Phi = [\varphi^{[1]}, ..., \varphi^{[N]}]$ contains the feature vector for all samples, $\boldsymbol{U} = [\theta^{[1]}, ..., \theta^{[N]}]$ contains all the sample parameters and $\boldsymbol{D}$ is the diagonal weighting matrix containing the weightings $d^{[i]}$. The covariance matrix $\boldsymbol{\Sigma}$ is obtained by

$$\boldsymbol{\Sigma} = \frac{\sum_{i=1}^{N} d^{[i]} (\boldsymbol{\theta}^{[i]} - \boldsymbol{A}^T \varphi(\boldsymbol{s}^{[i]}) (\boldsymbol{\theta}^{[i]} - \boldsymbol{A}^T \varphi(\boldsymbol{s}^{[i]})^T}{Z},$$
$$Z = \frac{\left(\sum_{i=1}^{N} d^{[i]}\right)^2 - \sum_{i=1}^{N} \left(d^{[i]}\right)^2}{\sum_{i=1}^{N} \left(d^{[i]}\right)}. \quad (15)$$

Typically $\boldsymbol{\varphi}(\boldsymbol{s}) = [1 \;\; \boldsymbol{s}]$ and $\boldsymbol{\phi}(\boldsymbol{s}) = [\boldsymbol{s}, \boldsymbol{s}^2]$, which results in linear generalization over contexts. In order to achieve non-linear generalization over contexts, we extend the algorithm by using normalized radial basis features (RBF) as feature function:

$$\phi_j(\boldsymbol{s}) = \varphi_j(\boldsymbol{s}) = \frac{\psi_j(\boldsymbol{s})}{\sum_{j=1}^{K} \psi_j(\boldsymbol{s}_j)}, \quad \psi_j(\boldsymbol{s}) = \exp\left(-\frac{(\boldsymbol{s} - c_j)^2}{2\sigma^2} \right) \quad (16)$$

Where j is the index of the element in the vector and the centers c_j are equally spaced in the range of the context space based on the desired number of RBFs K and σ^2 is the bandwidth of the RBF. We call the algorithm with this new setup RBF-REPS. In the next section, we will show the effectiveness of RBF-REPS for non-linear tasks.

4 Experiments

In this section, we evaluate standard-REPS and RBF-REPS for learning an upper level policy for different tasks including a complex walking task. In the first sets of experiments we use three toy tasks to only evaluate our proposed method to learn non-linear policies using RBF-REPS against standard REPS. We use a simple standard sin function with one parameter to show that RBF-REPS can learn non-linear policies while standard REPS can not. Furthermore we use a 3-link planar robot that has to reach a given point in task space and we call it planar reaching task. The resulting policy has 15 parameters, but we also use the same planar robot to reach a given hole on the ground that has 18 parameters and we call it planar hole reaching tasks. Both, the reaching task and hole reaching task use dynamic motor primitives (DMPs) [18] for parametrizing the lower level policies. Dynamic motor primitives recently has been used extensively to model the joints trajectory in robotics. For details on DMPs please refer to [18]. Finally, we use a simulated NAO robot (Fig. 4) with the described walking controller to test and verify our proposed approach. The NAO robot is a kid size humanoid robot with 21 degree of freedom.The simulation of the NAO is done by Robocup soccer simulator, rcssserver3d, which is the official simulator released by the Robocup community [23]. The open parameters of the algorithm that need to be

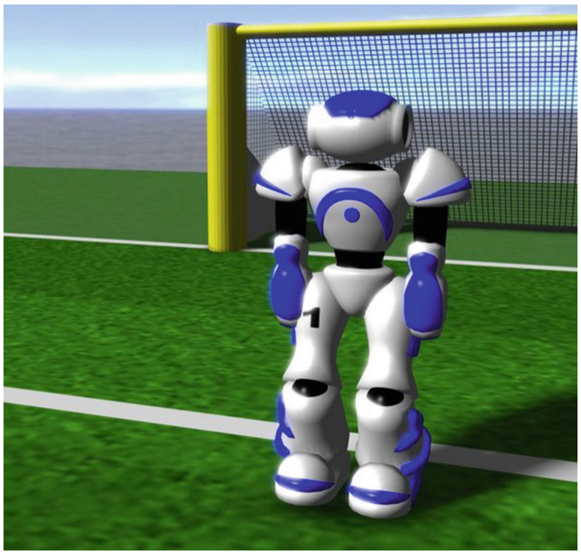
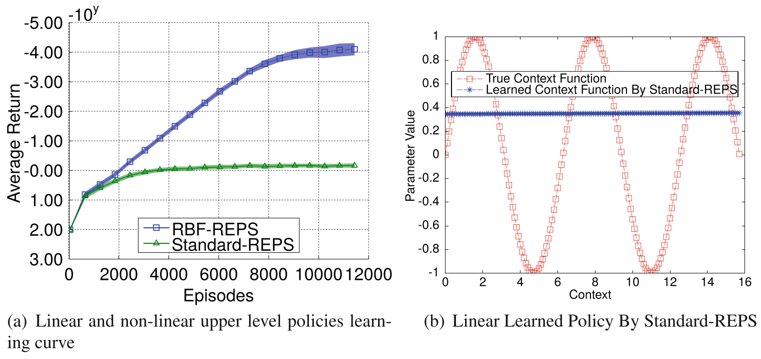
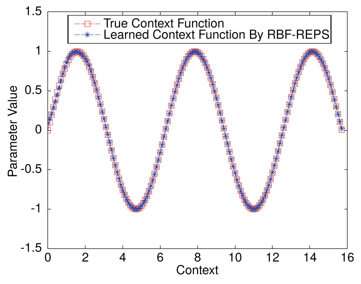


Fig. 4 The simulated Nao robot used for walking experiments



(a) Linear and non-linear upper level policies learning curve (b) Linear Learned Policy By Standard-REPS



(c) Non-Linear Learned Policy By RBF-REPS

Fig. 5 Standard sin function over contexts **a** This figure shows the learning curves of learning a linear and a non-linear upper level policies for sin task **b** This figure shows the learned linear upper level policy using linear features(standard REPS) over context [0.0 5PI]. The x axis is the desired context, and the y axis is the learned parameter for the given context. **c** This figure shows the learned non-linear upper level policy using RBFs(RBF-REPS) over context [0.0 5PI]. The x axis is the desired context, and the y axis is the learned parameter for the given context

specified by the user are ε (which will be 0.5 throughout all experiments), the number of last L samples to keep and number of N new samples to generate at each iteration. We also need to define the distribution $\mu(\boldsymbol{s})$ over contexts which, in this paper can be either a uniform or a Gaussian distribution. In case of RBF-REPS we need to specify the number of RBFs K as well as the bandwidth σ^2 value. In order to define the σ^2 for each context, we use the following simple heuristic:

$$\sigma^2 = \frac{\max(\text{context}) - \min(\text{context})}{K}$$

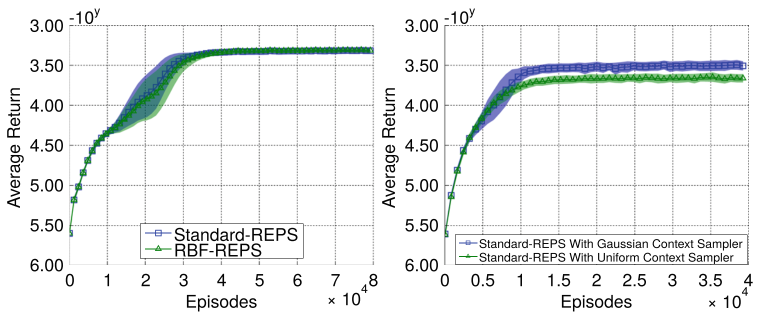
We run five trials for each experiment and show the mean and standard deviation of the results in the figures. The learning curves has been shown in log space in order to present the differences between learning curves clearly. The shaded area in the figures represents the standard deviation of the results while the solid lines are the mean of the trials.

4.1 Sinus Function Task

We consider learning a non-linear function over a one dimensional context $\boldsymbol{s}$ where the optimal parameter $\boldsymbol{\theta}$ for a given context $\boldsymbol{s}$ is $\sin(\boldsymbol{s})$ i.e., $\boldsymbol{\theta}^* = \sin(\boldsymbol{s})$. We use the following context-dependent reward function

$$R_{\boldsymbol{s}\boldsymbol{\theta}} = (\boldsymbol{\theta} - \sin(\boldsymbol{s}))^2$$

In each iteration, we generated 40 new samples. we always keep the last $L = 2000$ samples. We use a



(a) Linear and non-linear upper level policies learning curve (b) Uniform and Gaussian contex samplers comparison

Fig. 6 Evaluation on the Reaching task: **a** This figure shows the learning curves of learning a linear and a non-linear RBF based upper level policy for reaching task **b** This figure compares the performance of uniform and Gaussian contexts samplers

Gaussian context sampler. Figure 5 shows the learning curves and the learned upper level policies by these two algorithms. As the results show , contextual RBF-REPS successfully learns this non-linear function but standard contextual REPS only linearly approximate it and fails to complete the learning task.

4.2 Planar Reaching Task and Hole Reaching Task

In this setup, we used a 3-link planar robot with DMPs [18] as a lower level policy. Each link had a length of $1m$. The robot is modeled as a decoupled linear dynamical system. We used 5 basis functions per degree of freedom for the DMPs. In each iteration, we generated 40 new samples. We always keep the last $L = 2000$ samples.

Planar Reaching Task For completing the reaching task, the robot has to reach a via-point at time step 50 with its end-effector and at the final time step $T = 100$ the point $\boldsymbol{v}_{100} = [5, 0]$. We varied the

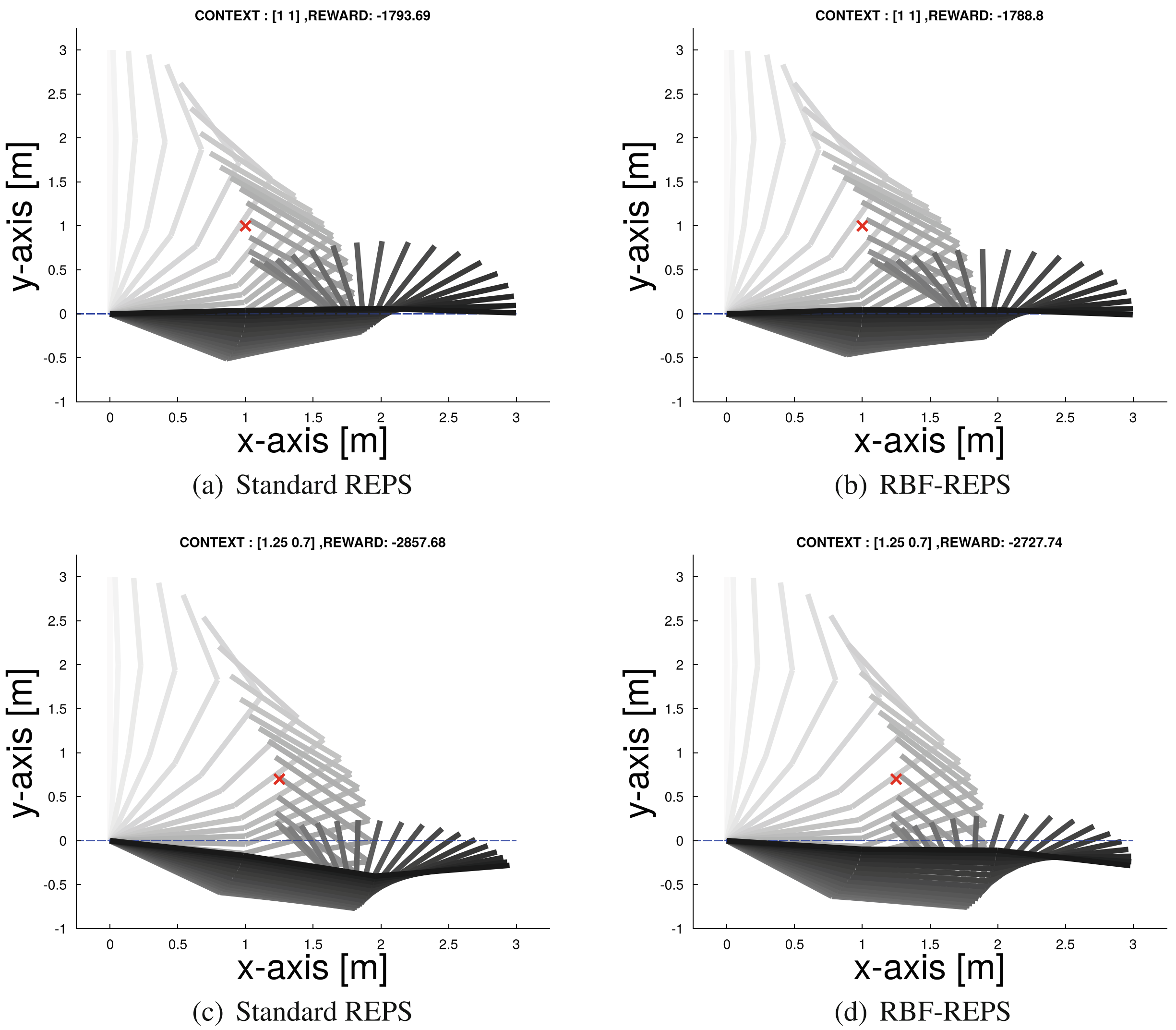


Fig. 7 The planar reaching task used for our comparisons. A 3-link robot has to reach a via-point at time step 50 . The via-point is indicated by the red cross. The postures of the resulting motion are shown as overlay, where darker postures indicate a posture which is close in time to the via-point. In title of each figure you can see the context and gained reward by each algorithm. In this task both algorithm performed similarly good as we saw in the learning curve, however considering the gained rewards, RBF-REPS slightly outperforms standard-REPS

first via-point from -0.5 to -1.5 in both dimensions. The reward was given by a context-dependent quadratic cost term for the two via-points as well as quadratic costs for high accelerations. The goal attractor of DMPs for reaching the final state was assumed to be known. Hence, our parameter vector had 15 dimensions. Figure 7 shows the setup. First, we use this setup to evaluate Gaussian and Uniform context samplers using standard REPS. Figure 6b shows that Gaussian context sampler outperform the uniform context sampler, hence thorough all experiments we use a Gaussian context sampler. We also compared standard-REPS and RBF-REPS with 2 RBFs. Figure 6a shows that these two algorithms perform similarly good on this task, however, RBF-REPS outperforms standard-REPS slightly, the reason is that this task is simple and a linear policy still can have a good performance. Figure 7 shows the learned policies for two different contexts by RBF-REPS and Standard-REPS.

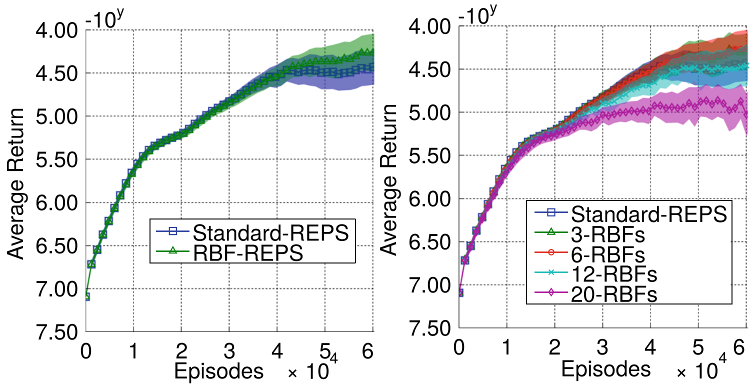
Planar Hole Reaching Task For completing the hole reaching task at final time step $T = 100$ the robot end effector has to reach the bottom of a hole (30 cm wide and 1 m deep) centering at a point varying from 1.4 to 2 without any collision with the ground or the hole wall. The reward was given by a quadratic cost term for the desired final point, quadratic costs for high accelerations and quadratic costs for collisions with the environment. Note that this performance function is discontinuous due to the quadratic costs for collisions . The DMPs goal attractor for reaching the final state in this task is unknown and need to be learned. Hence, our parameter vector had 18 dimensions.The learning setup is shown in Fig. 9 . The results shown in Fig. 8a shows that RBF-REPS with 6 RBFs clearly outperforms standard-REPS because of non-linearity of this task. We also evaluate the performance of RBF-REPS with different number of RBFs K in Fig. 8b where RBF-REPS with 6 RBFs has the best performance and RBF-REPS with 20 RBFs has the worse performance. Figure 9 shows the learned policies for two different contexts by RBF-REPS and Standard-REPS. The result shows that RBF-REPS successfully complete the task in both given contexts while standard-REPS fails to complete the task for the second context and it collides with the wall.

4.3 Robot Walking Task

In this section we learn an upper level policy for a one dimension context which is the linear speed of the gait, ranging from 0.1 $\frac{m}{s}$ to 0.6 $\frac{m}{s}$ and the direction of the walk stays always straight starting from a complete stop. We use the controller explained in Section 2 with 10 open parameters as lower level walking policy. Figure 3 illustrates our approach for learning an upper level policy for the robot walking lower level policy.

4.3.1 Used Learning Setup

In this section, we explain the settings of the learner to learn an upper level policy for the lower



(a) Linear and non-linear upper level policies learning curve (b) Evaluation of RBF-REPS with different number of RBFs

Fig. 8 Evaluations on hole reaching task: **a** This figure shows the learning curves of learning a linear and a non-linear RBF based upper level policy for the hole reaching task **b** This figure compares the performance of RBF-REPS with different number of RBFs

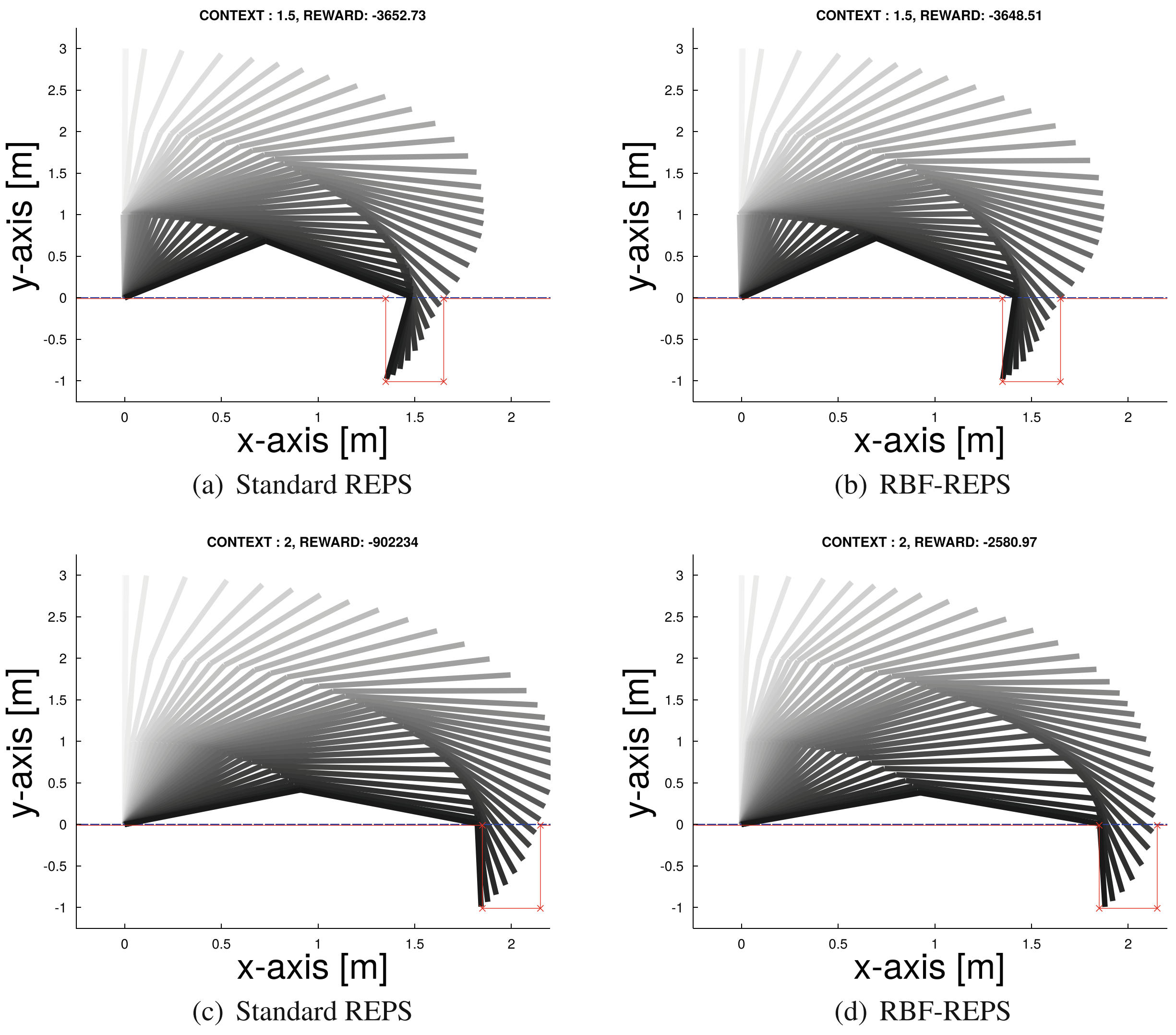


Fig. 9 The planar hole reaching task used for our comparisons. A 3-link robot has to reach the bottom of a hole (30 cm wide and 1 m deep) at time step 100 centering at a point varying from 1.4 to 2 without any collision with the ground or the hole wall. The *red lines* show the ground and the hole. The postures of the resulting motion are shown as overlay, where darker postures indicate a posture which is close in time to the bottom of the hole. In the title of each figure, you can see the given context value and gained reward by each algorithm. In this task, RBF-REPS considerably outperforms the standard-REPS. As you can see in the second row, the learned policy by standard-REPS for the given context 2 collides with wall while the one learned by RBF-REPS completes the task successfully (Please see the resulting rewards in the title of the figures)

level walking policy. There are constraints on the lower level walking policy parameters as follows

	A	x_0	z_{step}	T	r_{offset}
UpperBound	1	0.22	0.15	3	5
LowerBound	0	0.16	0.02	0.1	0
	γ_{tilt}	γ_{roll}	x_{step}	ϕ_{DSP}	σ_{step}
UpperBound	1	1	0.5	1	0.5
LowerBound	0	0	0.05	0	0

We initialize the Gaussian upper level policy with the mean

$$\frac{(UpperBound + LowerBound)}{2},$$

and a diagonal covariance matrix with diagonal elements

$$(UpperBound - LowerBound),$$

for all 10 parameters (Table 1). We also use a Gaussian distribution as a context sampler. The experiments start with 100 samples and continue with 30 samples at each iteration. We keep the last $L = 600$ samples. Due to the noise in robot's actuators, sensors and simulation uncertainties, we obtain noisy rewards. In order to remedy this problem, we use a re-sampling technique by evaluating each sample three times. We use the average of these three rewards as the expected reward of the corresponding sample. We also evaluate both standard-REPS and RBF-REPS with 5 RBFs on this task.

4.3.2 Constraint Handling

REPS is not a constraint optimizer so we need to have precautions to satisfy the constraints over the lower level walking parameters. A typical solution is adding a penalty term in the reward function to discourage the policy to search in infeasible parameters areas. Another solution is to transform the parameter space in a way that the parameters are always feasible and the constraints are satisfied. To do so, we use a sigmoid function for transforming the θ sampled from the upper level policy to a new θ' such that the constraints are satisfied

$$\theta'_j = \left(\frac{1}{1+\exp(-\boldsymbol{\theta}_j)}\right)(\mathrm{UB}_j - \mathrm{LB}_j) + \mathrm{LB}_j \tag{17}$$

UB and LB are the upper bound and lower bound vectors of the parameter space respectively. We will show that sigmoid based constraint handling method outperforms the penalty based one in our used setup.

4.3.3 Reward Function

For all experiments, contextual REPS aims at maximizing the following reward function for a given context s:

$$R(s,\theta) = -\alpha(\dot{x}-s)^2 - \beta(\dot{y})^2 - \sigma(1_{fall}) - \lambda(1_p) \tag{18}$$

Where s is the desired direct linear speed along the x-axis in robot torso coordinate system as was explained in Section 2. The $\dot{x}$ and $\dot{y}$ are the achieved speeds along the x-axis and y-axis respectively using the parameter sets θ. As we want the robot walks straight we punish for speed $\dot{y}$ along y-axis. The variable 1_{fall} is a flag that is 1 when the robot falls and zero otherwise. 1_p is set based on the constraint handling method that we want to use, it will be set to 1 if we use the penalty based method and is set to 0 if we use sigmoid based constraint handling method. We empirically set the coefficients of the reward function as follows:

α	β	σ	λ
10	1	100	1000

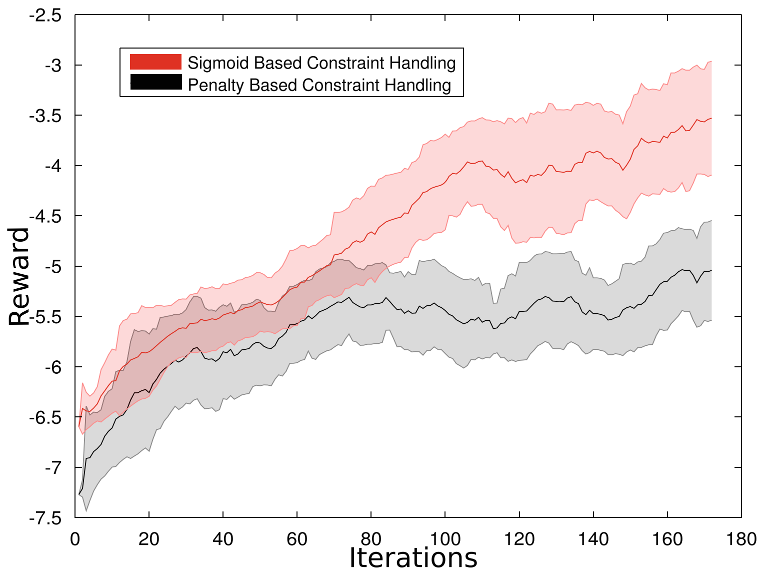
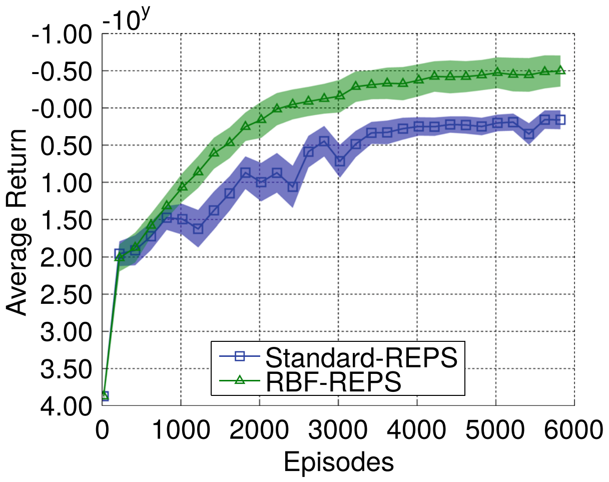


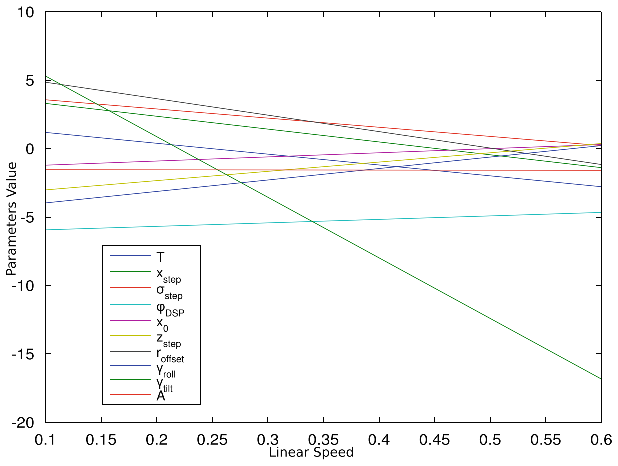
Fig. 10 This figure shows a comparison between sigmoid based and penalty based constraint handling methods for learning an upper level policy for different forward walk speeds using standard REPS. The results show that sigmoid based constraint handling method considerably outperforms the penalty based constraint handling method

4.3.4 Results

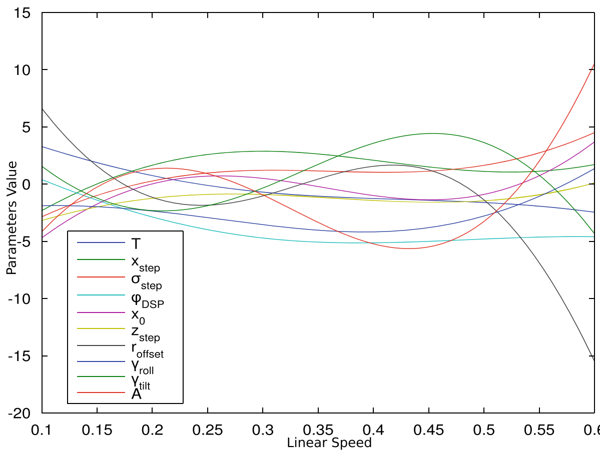
Evaluating Different Constraint Handling Methods We compare the sigmoid based and penalty based constraint handling methods. We use two different learning setups. The first learning set-up uses a penalty based constraint handling and the second one uses a sigmoid based constraint handling method. Fore both evaluations standard-REPS is used. The other settings are the same for both learning settings as was explained before. Figure 10 shows that the learning setup with sigmoid based constraint handling outperforms the one with the penalty based constraint handling. As you see in the Fig. 10, the penalty based method starts with a worse reward which makes sense, because at the beginning the algorithm



(a) Linear and non-linear upper level policies learning curve



(b) Linear upper level policy



(c) non-linear upper level policy

Fig. 11 **a** This figure shows the learning curves of learning a linear and a non-linear upper level policies for forward walking task **b** This figure shows the learned linear upper level policy for forward walking task using linear features over the desired speed [0.1 0.6]. The x axis is the desired context which is the linear speed in this task, and the y axis is the learned walk parameters for the given context (Table 1) **c** This figure shows the learned non-linear upper level policy for forward walking task using RBF features over speed [0.1 0.6] sometimes samples from the infeasible areas so it gets punished. It takes several iteration to find the feasible search areas. One of the issues with the contextual REPS is that the search distribution after each update shrinks due to the expectation reward maximization. And this leads to premature convergence which is the case in the penalty based constraint handling learning setup. Because once it finds the feasible area, its search distribution variance is too small to continue the exploration to find better solutions and it converges prematurely. On the other hand, the sigmoid based learning setup is searching from the beginning in the feasible search area, hence, it converges to a reasonable solution before its search distribution collapses. Given this results we choose the sigmoid based constraint handling method for the other experiments.

Standard-REPS and RBF-REPS We learn an upper level policy for the walking task using both standard-REPS and RBF-REPS. Figure 11a shows that both learning setups successfully learn the task. However RBF-REPS outperforms the Standard-REPS considerably due to non-linearity of the task. Figure 11b and c show the learned generalization policies for standard-REPS and RBF-REPS respectively. In order to show the significance of the results we chose 11 different forward speeds and used the final learned linear and non-linear policies to generate the controller parameters given the desired forward speed. Table 2 shows the final reward achieved using linear and non-linear polices for each speed. As you can see for almost all the query speeds, non-linear policy learned by RBF-REPS achieves considerably better reward without a single robot fall. However for example for query forward speeds 0.4, 0.55 and 0.6 using linear policies robot falls. Using non-linear policy robot can complete all the tasks without falling.

Table 2 Comparing the performance of the linear and non-linear policies for walking task

Speed ($\frac{m}{s}$)	Reward achieved by linear policy	Reward achieved by non-linear policy
0.1	−1.057	−0.166
0.15	−0.800	−0.011
0.2	−0.043	−0.0196
0.25	−0.139	−0.077
0.3	−0.633	−0.026
0.35	−0.353	−0.546
0.4	−3338.082	−0.220
0.45	−0.168	−0.206
0.5	−0.77	−0.366
0.55	−3336.321	−1.221
0.6	−3339.157	−0.911

Implementation Issues In this section we discuss a few implementation issues. One of the important issue is that the covariance matrix (Eq. 15) must be positive semidefinite to be used by the multivariate Gaussian policy. We regularize the covariance matrix by adding a small diagonal matrix in each iteration such that the eigen values of the covariance matrix always stay nonnegative. The second issue is optimising the dual function (Eq. 11). In order to optimise the dual function we use fmincon tool in matlab. However one could use any non-linear optimiser package. The last and possibly the most important issue is about choosing the number of samples to use. The number of samples to use heavily depends on the number of parameters we need to estimate for mean and covariance matrix. Therefore the more parameters we have to estimate the more samples we need. Those parameters depends on the dimension of the problem and the number of Radial basis functions we choose.

5 Conclusion and Future Work

In this paper, we studied generalizing a ZMP based walking controller for different contexts. In order to do so we used contextual Relative Entropy Policy Search. Using REPS, we successfully learned an upper level policy that generalizes the lower level walking policy for different forward speeds. We also studied different settings of REPS. We showed that using sigmoid function for constraint handling outperforms the the penalty based method. We also extended the REPS algorithm by using RBF features resulting in RBF-REPS and showed that it enables the algorithm to learn non-linear upper level policies which outperforms standard REPS. For the future work, we are going to test our method on the real robots. We will also investigate using more complex upper level policy representations such as neural networks and Gaussian processes.

Acknowledgments The first author is supported by FCT under grant SFRH/BD/81155/2011. The work was also partially funded by the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 and by FCT – Portuguese Foundation for Science and Technology under projects PEst-OE/EEI/UI0027/2013 and UID/CEC/00127/2013 (IEETA and LIACC). The work was also funded by project EuRoC, reference 608849 from call FP7-2013-NMP-ICT-FOF.

References

1. Kormushev, P., Ugurlu, B., Calinon, S., Tsagarakis, N.G., Caldwell, D.G.: Bipedal walking energy minimization by reinforcement learning with evolving policy parameterization. In: Proceedings of the International Conference on Robot Systems (2011)
2. Kupcsik, A.G., Deisenroth, M.P., Peters, J., Neumann, G.: Data-efficient generalization of robot skills with contextual policy search. In: Proceedings of the National Conference on Artificial Intelligence (AAAI) (2013)
3. Shafii, N., Khorsandian, A., Abdolmaleki, A., Jozi, B.: An optimized gait generator based on fourier series towards fast and robust biped locomotion involving arms swing. In: Proceedings of the International Conference on Automation and Logistics (2009)
4. Harada, K., Kajita, S., Kaneko, K., Hirukawa, H.: An analytical method for real-time gait planning for humanoid robots, International Journal of Humanoid Robotics (2006)
5. Gong, D., Yan, J., Zuo, G.: A review of gait optimization based on evolutionary computation, Applied Computational Intelligence and Soft Computing (2010)
6. Wang, J.M., Fleet, D.J., Hertzmann, A.: Optimizing walking controllers. In: ACM Transactions on Graphics (TOG) (2009)
7. Seungmoon, S., Hartmut, G.: Regulating speed and generating large speed transitions in a neuromuscular human walking model. In: Proceedings of the International Conference on Robotics and Automation (ICRA) (2012)
8. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K.: Biped walking pattern generation by using preview control of zero-moment point. In: Proceedings of the International Conference on Robotics and Automation (ICRA) (2003)
9. Shafii, N., Abdolmaleki, A., Ferreira, R., Lau, N., Reis, L.P.: Omnidirectional Walking and Active Balance for Soccer Humanoid Robot, in Progress in Artificial Intelligence (2013)
10. Vukobratovic, M., Stokic, D., Borovac, B., Surla, D.: Biped Locomotion: Dynamics, Stability, Control and Application. Springer, Berlin Heidelberg New York (1990)
11. Harada, K., Kajita, S., Kaneko, K., Hirukawa, H.: An analytical method for real-time gait planning for humanoid robots. International Journal of Humanoid Robotics (2006)
12. Srinivasan, M., Ruina, A.: Computer optimization of a minimal biped model discovers walking and running. Nature (2005)
13. Kagami, S., Nishivaki, K., Inaba, M., Inoue, H.: A Fast Dynamically Equilibrated Walking Trajectory Generation Method of Humanoid Robot. Autonomous Robots (2002)
14. Kofinas, N., Orfanoudakis, E., Lagoudakis, M.G.: Complete analytical inverse kinematics for NAO. In: Autonomous Robot Systems (Robotica) (2013)
15. Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., Hirukawa, H.: The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. Intelligent Robots and Systems (2001)
16. Cord, N., Rfer, T., Laue, T.: Gait optimization on a humanoid robot using particle swarm optimization. In: Proceedings of the Second Workshop on Humanoid Soccer Robots (2007)
17. Abdolmaleki, A., Shafii, N., Reis, L.P., Lau, N., Peters, J., Neumann, G.: Omnidirectional walking with a compliant inverted pendulum model. In: Advances in Artificial Intelligence–IBERAMIA (2014)
18. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Learning attractor landscapes for learning motor primitives. In: Neural Information Processing Systems (NIPS) (2002)
19. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
20. Peters, J., Mlling, K., Altun, Y.: Relative entropy policy search. In: AAAI (2010)
21. MacAlpine, P., Barrett, S., Urieli, D., Vu, V., Stone, P.: Design and optimization of an omnidirectional humanoid walk: a winning approach at the RoboCup 2011 3D simulation competition. In: AAAI (2012)
22. Shafii, N., Lau, N., Reis, L.P.: Learning to Walk Fast: Optimized Hip Height Movement for Simulated and Real Humanoid Robots. Journal of Intelligent and Robotic Systems (2015)
23. Xu, Y., Vatankhah, H.: Simspark: an open source robot simulator developed by the RoboCup community. In: RoboCup 2013 (2014)
24. Glaser, S., Dorer, K.: Trunk controlled motion framework. In: Proceedings of the 8th Workshop on Humanoid Soccer Robots, IEEE-RAS International Conference on Humanoid Robots (2013)
25. Hansen, N., Mller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). In: Evolutionary Computation (2003)

Abbas Abdolmaleki obtained B.Sc. (2009) and M.Sc. (2011) in Computer Engineering field of Artificial Intelligence from the University of Isfahan (Iran). He is currently a Ph.D. student at the University of Minho, Aveiro and Porto (Portugal), and his thesis topic is on information theoretic stochastic search. He is also a researcher at the Institute of Electronics and Telematics Engineering of Aveiro. He has worked on simulated rescue robots and simulated humanoid robot and achieved different ranks in Robocup competitions including 2 world championships (rescue simulation). His main research interests include stochastic search for black box optimization, policy search for robotics and multi agent systems.

Nuno Lau is Assistant Professor at Aveiro University, Portugal. He got his Electrical Engineering Degree from Oporto University in 1993, a DEA degree in Biomedical Engineering from Claude Bernard University, Lyon, France, in 1994 and the PhD from Aveiro University in 2003. His research interests include Intelligent Robotics, Artificial Intelligence, Multi-Agent Systems and Simulation. Nuno Lau is one of the team leaders of FC Portugal team, that has been 3 times World champion of RoboCup (simulation leagues). He is also a member of the CAMBADA Middle-Size League team which has also been RoboCup World Champion. He has lectured courses at Phd and MSc levels on Intelligent Robotics, Distributed Artificial Intelligence, Computer Architecture, Programming, etc. Nuno Lau is the author of more than one hundred publications in international conferences and journals. He is currently the President of Portuguese Robotics Society and Principal Investigator of the Intelligent Robotics and Systems group at Aveiro University.

Luis Paulo Reis is an Associate Professor at the University of Minho in Portugal and member of the Directive Board of LIACC - Artificial Intelligence and Computer Science Laboratory where he coordinates the Human-Machine Intelligent Cooperation Research Group. During the last 25 years he has lectured courses at the University on Artificial Intelligence, Intelligent Robotics, Simulation and Modelling and Computer Programming. He was principal investigator of more than 10 national/international research projects in those areas. He won more than 50 scientific awards including wining more than 15 RoboCup international competitions and best papers at conferences such as ICEIS, Robotica, IEEE ICARSC and ICAART. He supervised 16 PhD and 95 MSc theses to completion. He organized more than 50 scientific events and belonged to the Program Committee of more than 200 scientific events. He is the author of more than 300 publications in international conferences and journals.

Jan Peters is a full professor (W3) for Intelligent Autonomous Systems at the Computer Science Department of the Technische Universitaet Darmstadt and at the same time a senior research scientist and group leader at the Max-Planck Institute for Intelligent Systems, where he heads the interdepartmental Robot Learning Group. Jan Peters has received the Dick Volz Best 2007 US PhD Thesis Runner-Up Award, the 2012 Robotics: Science & Systems - Early Career Spotlight, the 2013 INNS Young Investigator Award, and the IEEE Robotics & Automation Societys 2013 Early Career Award. In 2015, he was awarded an ERC Starting Grant. Jan Peters has studied Computer Science, Electrical, Mechanical and Control Engineering at TU Munich and FernUni Hagen in Germany, at the National University of Singapore (NUS) and the University of Southern California (USC), and been an visiting researcher at the ATR Telecomunications Research Center in Japan. He has received four Master's degrees in these disciplines as well as a Computer Science PhD from USC.

Gerhard Neumann is an Assistant Professor at the TU Darmstadt since September 2014 and head of the Computational Learning for Autonomous Systems (CLAS) group. His research topics are policy search for robotics, hierarchical reinforcement learning, non-parametric inference and multi-agent reinforcement learning. Before becoming assistant professor, he joined the IAS group as Post-Doc in November 2011 and became a Group Leader for Machine Learning for Control in October 2013. He did his Ph.D. at the Graz University of Technology under the supervision of Wolfgang Maass. He won several best-paper awards and is principle investigator of the EU project Romans and project leader for the DFG Project LearnRobots.