Generic Neural Locomotion Control Framework for Legged Robots

Mathias Thor, Tomas Kulvicius, and Poramate Manoonpong

***Abstract*—In this paper, we present a generic locomotion control framework for legged robots and a strategy for control policy optimization. The framework is based on neural control and black-box optimization. The neural control combines a central pattern generator (CPG) and a radial basis function (RBF) network to create a CPG-RBF network. The control network acts as a neural basis to produce arbitrary rhythmic trajectories for the joints of robots. The main features of the CPG-RBF network are: 1) it is generic, since it can be applied to legged robots with different morphologies; 2) it has few control parameters, resulting in fast learning; 3) it is scalable, both in terms of policy/trajectory complexity and the number of legs that can be controlled using similar trajectories; 4) it does not rely heavily on sensory feedback to generate locomotion and is thus less prone to sensory faults; and 5) once trained, it is simple, minimal, and intuitive to use and analyze. These features will lead to an easy-to-use framework with fast convergence and the ability to encode complex locomotion control policies. In this work, we show that the framework can successfully be applied to three different simulated legged robots with varying morphologies, and even broken joints, to learn locomotion control policies. We also show that after learning, the control policies can also be successfully transferred to a real-world robot without any modifications. We, furthermore, show the scalability of the framework by implementing it as a central controller for all legs of a robot and as a decentralized controller for individual legs and leg pairs. By investigating the correlation between robot morphology and encoding type, we are able to present a strategy for control policy optimization. Finally, we show how sensory feedback can be integrated into the CPG-RBF network to enable online adaptation.**

***Index Terms*—Legged robots, Neurorobotics, Generic control, Locomotion control, Learning, Policy optimization**

Manuscript received July 29, 2018; revised August 26, 2019. The work was supported in part by the Horizon 2020 Framework Programme (FETPROACT-01-2016 FET Proactive: emerging themes and communities) under Grant 732266 (Plan4Act) [P.M., Project WP-PI], the NSFC-DFG Collaborative Research Program of the National Natural Science of Foundation of China (grant no. 51861135306, P.M., project co-PI), and a startup grant on Bio-inspired Robotics from VISTEC [P.M., Project PI].

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. This includes 10 multimedia MP4 format video clips, which show different robots using the proposed framework, and supplementary material in which we demonstrate how sensory feedback can be integrated into the CPG-RBF network. This material is 91.5 MB in size.

M. Thor is with the Embodied Artificial Intelligence and Neurorobotics Laboratory, SDU-Biorobotics section, The Mærsk Mc-Kinney Møller Institute, The University of Southern Denmark, Odense 5230, Denmark (e-mail: mathias@mmmi.sdu.dk).

T. Kulvicius is with the Department of Computational Neuroscience, University of Göettingen, Göettingen, 37077, Germany.

P. Manoonpong is with the College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China, and with the Embodied Artificial Intelligence and Neurorobotics Laboratory, SDU-Biorobotics section, The Mærsk Mc-Kinney Møller Institute, The University of Southern Denmark, Odense 5230, Denmark, and with the School of Information Science & Technology, Vidyasirimedhi Institute of Science & Technology, Rayong 21210, Thailand (e-mail: poma@nuaa.edu.cn).

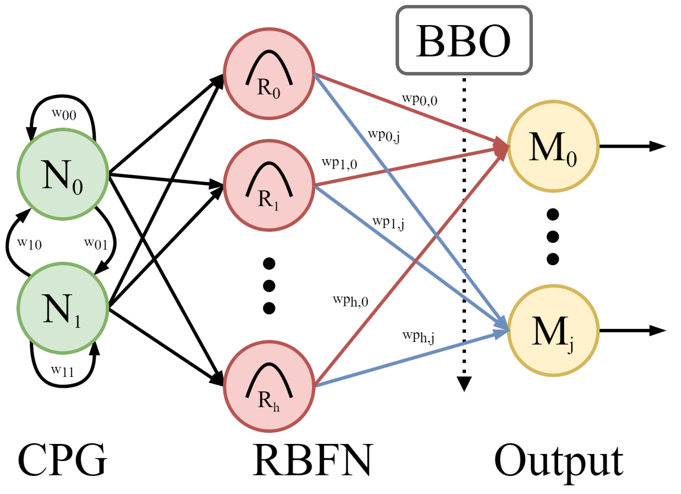


Fig. 1. CPG-RBF network, combining a CPG with an RBF network. The policy is encoded in the synaptic weights, $wp_{j,k}$, connecting the RBF neurons, R_h, to the motor neurons, M_j. These weights are optimized using black-box optimization (BBO) as indicated by the dashed arrow. The weights $w_{0,0}$, $w_{0,1}$, $w_{1,0}$, and $w_{1,1}$ of the CPG are fixed such that its outputs oscillate at a certain frequency (i.e., here approximately 0.3 Hz (low walking frequency)).

I. INTRODUCTION

DESIGNING locomotion controllers is challenging due to the varying morphology of different robots. While one type of legged robot requires a particular set of joint trajectories, another might need something different. The fact that legged robots often have many degrees of freedom, and that manual tuning of parameters in order to achieve good performance is difficult, calls for an automated approach. Good locomotion performance can be determined by: 1) fast straight walking (i.e., distance traveled or speed); 2) less body oscillation (i.e., stability); 3) less slip during a stance phase (i.e., slippage); and 4) no collision between legs. By using machine learning, it is possible to let the legged robot learn trajectories from interacting with the environment. Optimization is typically done in simulation since the robot needs many interactions with the environment, depending on the learning algorithm and controller complexity [1], [2], [3], [4].

Many of today's locomotion controllers are problematic since they either rely on imitation learning for simple control without optimization [5], [6] or learning where the neural control is either too simple [2], [1] or too complex [3], [4]. A controller that is too simple does not allow for complex locomotion control policies to be learned, while a controller that is too complex and relies on large complex neural networks suffers from slow learning (see Section II for more details). Complex controllers also make it hard to analyze the learned locomotion control policy and extend the framework with additional behaviors. Finally, most controllers rely heavily on sensory feedback which introduces a possible breaking point in cases of sensory fault. Sensory feedback is important for controlling legged robots and essential for adapting to difficult terrain, where legged robots are especially useful. However, when the sensory feedback is tightly integrated into a controller, it is difficult and in some cases, impossible to deal with sensory failures.

To overcome these problems, we need a control framework that is simple and yet able to encode complex locomotion control policies. A simple control framework implies that it is easy to analyze and has few policy parameters, resulting in fast learning. One area where fast learning is especially important is research concerning optimization directly in the real world [7], [8]. Finally, the framework also needs to be generic and scalable to support robots with different morphologies and the integration of additional behaviors. A modular setup will make the framework less prone to sensory faults since behavior modules can be added and removed online.

For these reasons, we present a comprehensible, generic, and flexible/scalable neural locomotion control framework with different control encodings and control policy optimization for locomotion generation of legged robots with different morphologies. The framework, which is the main contribution of this work, is based on neural control and black-box optimization (BBO). The neural control called the CPG-RBF network (acting as a neural basis) combines a central pattern generator (CPG) with a radial basis function (RBF) network (see Fig. 1). CPGs are one of the most popular methods for locomotion generation in legged robots (for reviews see [9], [10]). They are, however, unable to take full advantage of the robot's morphology since the rhythmic CPG output by default is wave-shaped and cannot be easily reshaped using the CPG alone [9]. To overcome this, we use the RBF network, which is often used for function approximation and can, therefore, be set up to reshape the CPG output. The RBF network can either amplify or reduce a specific part of the CPG signal. Thus, it is possible to train and produce arbitrary shaped rhythmic trajectories for the joints of a legged robot. The trajectories are encoded in the synaptic weights connecting the RBF network to the motor outputs (blue and red connections in Fig. 1). Possessing only one hidden layer, RBF networks have a faster convergence rate in comparison to multi-layer perceptrons like those presented in [3], [4]. Moreover, interpreting the functionality of each neuron in the hidden layer is straightforward. This is because each neuron's activation encodes the joint position at a particular phase in the stepping cycle. Thus, the simplicity of the CPG-RBF network makes it easy to overview and analyze. One way to analyze the network is by visualizing the trajectories using only the network weights, which may be useful when comparing to biology. Furthermore, the simplicity makes the network scalable and expandable with additional behavior modules.

The CPG-RBF network was presented for the first time in [11], where the basics of the neural controller were explained. In this work, we significantly elaborate on the previous study and demonstrate that the framework can also be applied to three robots with different morphologies. We also elaborate on the scalability of the framework, since not only can it be implemented as a central controller for all legs of a robot, but also in a decentralized way to control individual legs or leg pairs (i.e., front, hind, and middle legs). Through experiments, we aim to investigate how various control strategies relate to different legged robot morphologies. In this way, we propose a strategy for control policy optimization. More specifically, we investigate three different control strategies or controller encodings: indirect encoding where all legs of the robot are controlled by the same CPG-RBF network, semi-indirect encoding where all leg pairs are controlled with the same CPG-RBF networks, and direct encoding where each leg has its own CPG-RBF network.

The main objective of this work is to explore the use of BBO to quickly tune CPG-RBF networks that can generate feedforward joint trajectories for robots with different morphologies. In other words, to provide motor primitives as a basis for robot locomotion. However, to also show that the CPG-RBF network is extendable with sensory feedback, we present a preliminary study in the supplementary material where sensory feedback from an inertial measurement unit is integrated into the network. This integration enables the robot to adapt its body posture online. In the discussion (Section VIII), we consider other modular extensions of the framework, such as extending it with different sensor-driven behaviors for leg movement adaptation to navigate on uneven, unpredictable terrain (e.g., terrain with holes and obstacles). Such extensions are possible due to the scalability, modularity, and simplicity of the CPG-RBF network.

In summary, the main features of the CPG-RBF network are: it is simple, minimal, and intuitive to use and analyze; it has few control parameters, resulting in fast learning; it is scalable both in terms of policy/trajectory complexity, controller encoding, and extendable; finally, it is generic since it can be applied to legged robots with different morphologies. In this work, we: 1) demonstrate that the CPG-RBF network can be applied to locomotion generation for legged robots with different morphologies; 2) analyze how the controller encoding of the CPG-RBF network is related to the morphology of the legged system; 3) show that the learned controller network can be deployed directly to a real-world robot and used for damage compensation; 4) show that it is possible to integrate sensory feedback into the CPG-RBF network; and 5) discuss the framework potential in terms of real-world learning, encoding switching, policies for different behaviors, and extension with different sensory feedback. To this end, the proposed framework can learn to drive three different legged robots for fast locomotion, including one with broken joints.

II. RELATED WORK

In [2], Oliveira et al. described an approach used to evolve a locomotion controller for a simple quadruped robot. The controller is based on CPGs such that it has seven free parameters for describing the CPG amplitude and offset of the front and hind legs as well as the overall swing frequency. These parameters are learned using a genetic algorithm (GA) to find the optimal combination for minimizing body vibration while maximizing the velocity and stability margin. The GA is a population-based algorithm that uses a technique inspired by evolution, such as inheritance, mutation, selection, and crossover [12]. The presented optimization of quadruped robot locomotion is promising and straightforward. However, the parameter space is narrow and does not allow for complex control policies and the aforementioned good locomotion performance. This is mainly because it is not possible to optimize the shape of the CPG output signal; only its uniform amplitude and offset. Finally, the authors only demonstrate their approach on one single quadruped robot with eight joints (i.e., two joints per leg) and in a simulated environment.

In [5], Nakanishi et al. introduced a locomotion controller for biped locomotion using dynamic movement primitives (DMPs) as a CPG. In their approach, they allow rhythmic DMPs to learn demonstrated human trajectories using locally weighted regression. Moreover, they propose an adaptation algorithm for the walking frequency, based on phase resetting and entrainment, to make the controller more robust against external perturbations and environmental changes. The algorithm uses foot contact information to update the CPG frequency and phase resetting. The results show that the DMPs can learn the demonstrated trajectories, which could be seen as prior knowledge and a basis for further improvement using reinforcement learning (RL). However, this was not explored and the trajectories are thus only as good as the ones provided by the human demonstrator. Furthermore, the controller was only demonstrated on a single robot.

In [6], Rosado et al. also presented a biped locomotion controller using rhythmic DMPs learned in task space, from a single demonstration. They use three DMPs for each leg, representing the x, y, and z dimensions, respectively. Their work is related to [5], but instead of exploring phase resetting to generate adaptive walking, they adapt the parameters of DMPs related to task variables. These parameters include amplitude, frequency, and offset, which directly relate to step length, hip height, foot clearance, and forward velocity. As in [5], they did not further improve the trajectory of the DMPs after learning the demonstrated trajectory and the controller was only used on a single robot.

Recently, Hwangbo et al. [3] described a novel method for training a neural network policy in simulation and subsequently transferring it to the legged quadruped robot called ANYmal. They achieve high efficiency and realism in simulation by combining classical models representing well-known parts of the robot with learning methods that can handle complex dynamics which are often hard to model (e.g., actuator dynamics, control signal delays, and low-level controller dynamics). They learn these mappings in an end-to-end manner using self-supervised learning with a deep neural network. With the learned dynamics, they train controllers using an RL-based optimization method and subsequently deploy them directly in the physical robot. The controller implementing the locomotion control policy is a four-layer neural network (one input layer, two hidden layers, and one output layer) with multiple inputs (e.g., robot states, joint states, and command history). The two hidden layers consist of 256 and 128 neurons, respectively. Finally, 12 output neurons (outputting joint positions) drive the 12 joints of ANYmal. Their novel approach to dealing with the reality gap and training networks for locomotion generation is very promising. The positive results in terms of locomotion performance show that the study, without doubt, makes a significant contribution to the field of data-driven methods for locomotion control. However, the learned controller network is hard to comprehend, consisting of millions of weights which have to be learned by the RL-based method. The complexity of the neural networks makes it hard to identify how and why the controller uses different sensors, and if they are even needed for basic locomotion generation. The fact that it relies heavily on sensory feedback and command history as inputs also makes it prone to sensory fault, since in this case, the controller may not function properly. Furthermore, hundreds of millions of samples are needed to train the network, resulting in long training sessions; nine days of simulated time for locomotion control policy and 79 days of simulated time for the recovery from fall control policy. Simulated time is the time used by the robot inside the simulated environment and therefore independent of the CPU. Thus, when applying the proposed locomotion controller one also needs to either use their learned dynamics or learn new ones themselves since other available simulation frameworks, such as Gazebo [13] and V-REP [14] (used by many robotics communities) are too slow.

In [4], Clune et al. presented a similar controller network as [3] for a simulated quadruped robot. This smaller network has three layers with 20 neurons per layer and also relies heavily on sensory feedback. Based on sensory feedback, the network generates new positions for the 12 joints of the quadruped robot. The network thus suffers from the same drawbacks as in [3]. However, the main contribution of their work is the investigation of different controller encodings and how those relate to the regularity of the system. In their work, they used HyperNEAT (NeuroEvolution of Augmenting Topologies) as indirect encoding and FT-NEAT for direct encoding (for a detailed explanation see [4]). The authors found that the performance of indirect encoding improves with the regularity of the problem but also that the bias of indirect encoding toward regularity hurt its performance on problems containing some irregularity (where direct encoding performs the best). Additionally, their findings indicate that it is advantageous to shift from indirect to direct encoding during training (when training with indirect encoding converges) because, in this way, it is possible to make subtle adjustments to regular patterns to account for problem irregularities [4]. With regard to legged robots, the authors modulated the regularity of the quadruped robot by changing the number of faulty joints (i.e., constant uniform noise is added to the position commands of the faulty joints). Thus, the results are limited regarding generic locomotion control, since it is only carried out for one single robot without any changes to the morphology. In this study, we investigate how the controller encoding relates to the morphology regularity of different robots.

III. CPG-RBF NETWORK

In this section, we explain the CPG-RBF network and its different components, starting with the CPG. The CPG-RBF network is inspired by rhythmic DMPs [15], [16] and therefore, at the end of this section, we compare the two.

A. *Central Pattern Generator (CPG)*

A CPG is a group of interconnected neurons located at the spinal cord of vertebrates and in the thoracic ganglia of invertebrates. The term *central* means that a CPG can be activated to generate a motor pattern without the requirement of sensory feedback. CPGs play a central role in elucidating locomotion mechanisms and other rhythmic movements such as breathing [17], [18]. Different CPG models with varying complexity have been proposed: conceptual biological models [19], detailed biophysical models [20], connectionist models [21], and abstract models [9], [10]. In the domain of robot control, most previous research has employed abstract CPG models using coupled oscillators to generate basic periodic movement patterns [22]. From the control perspective, CPGs have various interesting properties such as robustness against perturbations, easy and smooth modulation of the frequency, suitability for distributed implementation, and the fact that they use few control parameters. Concerning the latter, we especially explore the distributed implementation properties of different controller encodings, as explained in Section IV.

For the CPG-RBF network, we use the abstract neural SO(2)-oscillator based CPG model [23] (see CPG in Fig. 1). The SO(2)-based CPG is a versatile recurrent neural network consisting of two fully-connected standard additive discrete-time neurons N_0 and N_1, both using a sigmoid transfer function. The SO(2) oscillator can exhibit various dynamical behaviors (e.g., periodic patterns with different frequencies, chaotic patterns, and hysteresis effects [24], [25], [26]) by changing its synaptic weights through manual control or sensory feedback. These dynamical network behaviors can subsequently be exploited for complex locomotion (e.g., chaotic leg movement for self-untrapping of a leg from a hole in the ground [27], walking at different frequencies [28]).

The outputs of the two neurons in the SO(2) oscillator are given by

$$o_i(t+1) = \tanh\left(\sum_{j=0}^{N} w_{ij}(t) o_j(t)\right), \qquad (1)$$

where o_i is the output from neuron i, N is the number of neurons, and w_{ij} is the synaptic weight from neuron i to j. The two neurons produce rhythmic outputs with a phase shift of $\pi/2$.

As proven by Pasemann et al. [23], the network produces a quasi-periodic output when the weights are chosen as

$$\begin{pmatrix} w_{00}(t) & w_{01}(t) \\ w_{10}(t) & w_{11}(t) \end{pmatrix} = \alpha \cdot \begin{pmatrix} \cos\varphi(t) & \sin\varphi(t) \\ -\sin\varphi(t) & \cos\varphi(t) \end{pmatrix}, \qquad (2)$$

with $0 < \varphi(t) < \pi$ as the frequency-determining parameter. Parameter α determines the amplitude and the nonlinearity of the output oscillations. For the CPG-RBF network, $\alpha = 1.01$ and $\varphi = 0.01\pi$ are used to obtain harmonic oscillation at a fixed frequency of ≈ 0.30 Hz. The two outputs from the CPG can be seen in Fig. 2. Note that the frequency can also be learned [22], [29], but for the purpose of this study, it remain fixed.

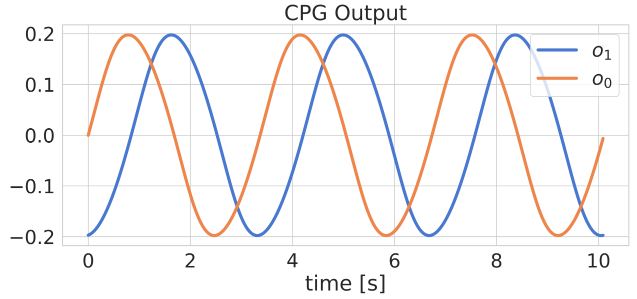


Fig. 2. Harmonic CPG outputs oscillating at a frequency of ≈ 0.30 Hz. o_0 is the output from neuron N_0 while o_1 is the output from neuron N_1.

B. *Radial Basis Function Network (RBF network)*

An RBF network is an artificial neural network that uses radial basis activation functions [30]. An RBF network consists of only three layers; an input layer which in our case is the CPG, a hidden layer, and an output layer which in our case consists of the motor neurons (see Fig. 1). The activation functions of the neurons in the hidden layer are radial basis functions; more specifically, two-dimensional Gaussian functions. The transfer function of the hidden neuron is thus given as

$$R_h = e^{-\left(\frac{(o_0-\mu_{h,0})^2+(o_1-\mu_{h,1})^2}{\sigma^2_{RBF}}\right)}, \qquad (3)$$

where, $\mu_{h,0}$ and $\mu_{h,1}$ are two means of RBF neuron R_h, σ^2_{RBF} is the common variance for the two means, and R_h is the response of the RBF neuron when receiving input o_0 and o_1 from the CPG. The two means are set manually so that the RBF neurons or kernels are equally distributed along one period of the CPG output signals. This is achieved by

$$\mu_{h,n} = o_n\left(\frac{(h-1)\cdot T}{H-1}\right), \qquad (4)$$

where n is an index for the CPG output, T is the period of the CPG signal ($T \approx 1/0.30\,Hz$), and H is the total number of RBF neurons. The advantage of equally distributing the means along one period of the CPG output signal is that in doing so it is possible to modify discrete parts of the CPG signal shape, and the centers of the kernels do not need to be learned. For example, when using $H = 20$, it is possible to make the j^{th} joint move either more or less at the center of its trajectory by altering the synaptic weight $wp_{10,j}$ from the tenth RBF neuron R_{10} to motor neuron M_j.

The number of RBF neurons, H, directly relates to the complexity of the output trajectory. While a high number of neurons enables complex trajectories that can approximate almost any functions, a small number of neurons can only produce simple trajectories. However, a small number of neurons also means fewer policy parameters and thus a faster convergence rate. While the number of neurons controls the complexity of the output trajectory, their variance, σ^2_{RBF}, controls its smoothness. A high variance results in a very smooth trajectory, while a smaller allows a more high-frequency output trajectory. This creates a trade-off, and in the case of locomotion control, $H = 20$ and $\sigma^2_{RBF} = 0.04$ are chosen, allowing the learning of smooth complex control policies at acceptable convergence rates. From empirical study, we found that a smaller σ^2_{RBF} resulted in jerky movements while a higher one resulted in too simple trajectories and low returns (the learned shapes were periodic patterns with almost symmetrical ascending and descending slopes). For H, we found that a smaller number of RBF neurons cannot generate complex trajectories (e.g., periodic patterns with asymmetrical ascending and descending slopes and different speeds as shown in Fig. 8) as well as obtaining high returns. In contrast, a higher number of kernels only increases the convergence rate and not the return. Thus, $H = 20$ and $\sigma^2_{RBF} = 0.04$ are used in the experiments presented in Section VI.

C. Dynamic Movement Primitive (DMP)

When comparing the CPG-RBF network to rhythmic DMPs [15], [16], it is clear that the two are very similar. They both inhibit a timing system (i.e., the CPG for the CPG-RBF network and the canonical system for the DMP), and a shaping system (i.e., the RBF network for the CPG-RBF network and the forcing function for the DMP). However, the CPG-RBF network allows us to use only neurons as well as simply extending it with an online frequency adaptation mechanism [31], [18], [29]. Furthermore, by using a neural-based CPG, it is possible to explore the neurodynamics (chaotic patterns and hysteresis effects) of the CPG by modulating the synaptic weights [24], [25], [26].

IV. Controller Encoding

Clune et al. [4] presented for the first time, a comprehensive study on controller encodings, demonstrating that phenotype regularity enables indirect encoding to outperform direct encoding as the regularity of the problem increases (as also described in Section I). In the case of legged robots, the regularity often boils down to their morphology, i.e., the level of symmetry. However, other factors, such as motor imprecision, may also cause irregularity.

In [4], the authors investigated direct and indirect encodings using HyperNEAT and FT-NEAT, respectively (for a detailed explanation see [4]). NEAT is a method for evolving artificial neural networks using a genetic algorithm in a manner inspired by nature. NEAT utilizes the idea that it is effective to start evolution with small, simple networks, allowing them to increase in complexity over generations.

In this work, controller encoding is not provided by the type of learning algorithm but rather by the implementation of the neural controller. This is possible because the proposed framework is scalable, and not only can it be implemented as a central controller for all legs of a robot, but also decentralized to control individual legs or leg pairs (i.e., front, hind, and middle legs). We call it indirect encoding when the CPG-RBF network acts as a central controller where the same joint trajectories are learned for all legs (see Fig. 3a). Direct encoding is when the CPG-RBF network is fully decentralized, and different joint trajectories are learned for each leg, enabling them to move differently from each other (see Fig. 3c). Finally, semi-indirect encoding bridges the gap between indirect and direct, where the CPG-RBF network is partly decentralized, and only the joint trajectories for the different leg pairs are learned (see Fig. 3b). The idea behind semi-indirect encoding is to exploit the bilateral symmetry of legged robots. Note that, as the controller decentralizes further, more complex control policies can be learned due to the increased number of unique trajectories. However, this is at the cost of additional control policy parameters and thus, slower learning.

V. Learning policies with PI^{BB}

In this study, we focus on providing a generic locomotion control framework rather than comparing different optimization approaches. Therefore, we employ the state-of-the-art learning mechanism PI^{BB} [32], which is a probability-based BBO approach. It follows a direct policy search in order to improve the policy parameters with respect to a reward function. PI^{BB} is a BBO variant of the RL-based method called "Policy Improvement with Path Integrals" (PI^2) [33] with constant exploration and without temporal averaging. In particular, it can be seen as a special case of covariance matrix adaptation evolution strategy (CMA-ES) [34] without covariance matrix updating. The modifications introduced in PI^{BB} result in a simpler algorithm providing both better convergence speed and final return (accumulated rewards) when compared to PI^2. Furthermore, PI^{BB} is robust with no matrix inversions and can be used in model-free learning with easy-to-construct reward function requirements [35]. PI^{BB} has no open parameters except for exploration noise [32], and is faster than gradient-based RL approaches by one order of magnitude [33].

PI^{BB} was selected not only for its simplicity but also because PI^2 has been successfully used in other continuous, high-dimensional action spaces [33], [35], [32]. In particular, it has especially demonstrated impressive results when using DMPs as the underlying parameterized control policy. Since PI^2 is similar to PI^{BB} [32] and the CPG-RBF network is comparable to DMPs, PI^{BB} is appropriate for the task at hand.

The pseudocode for PI^{BB} can be seen in Algorithm 1, which also illustrates its simplicity. Here, it can be seen that the mechanism executes K roll-outs, all with different Gaussian noise, ϵ_k, added to the control policy parameters, $wp_{k,j}$. The results from the K roll-outs are K returns, R_k, describing how well the policy, with added exploration noise, performed according to a certain reward function. Finally, the probability for each roll-out is calculated and used in cost-weighted averaging to update the policy parameters.

In the case of the CPG-RBF network, the control policy parameters $wp_{k,j}$ are given as the synaptic weights from the RBF neurons to the motor neurons (see $wp_{k,j}$ in Fig. 1).

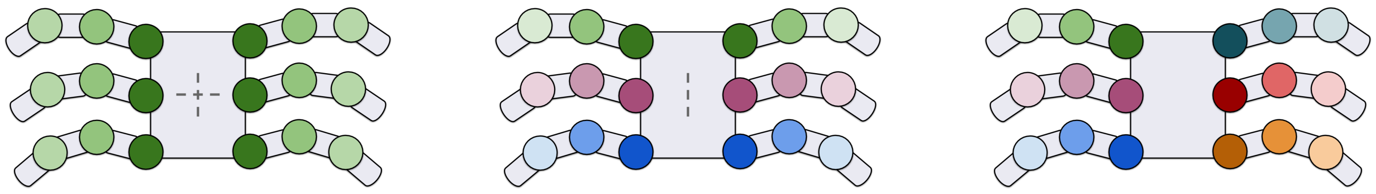

(a) Indirect encoding (b) Semi-indirect encoding (c) Direct encoding

Fig. 3. Examples of the three types of controller encoding on a hexapod robot with three joints (colored circles) in each leg. Each color indicates a different trajectory or set of CPG-RBF network weights. The dashed lines in the center of the robot indicate symmetry. For example, a vertical line means that the controller is using the same joint trajectories for the left and right legs while the crossed lines mean that all legs are using the same trajectories.

Algorithm 1 PI^{BB}

```
while cost not converged do
    // Execute $K$ roll-outs
    for each $k \in K$ do
        // Sample in parameter space
        $\epsilon_k = \mathcal{N}(0, \sigma^2_{PI^{BB}})$
        // Execute policy and record final return (R)
        $R_k = \texttt{execCPGRBFN}(wp_{k,j} + \epsilon_k)$
    end for
    // Calculate trajectory cost
    for each $k \in K$ do
        $S_k = e^{-\lambda \cdot \frac{R_k - \min_k(R_k)}{\max_k(R_k) - \min_k(R_k)}}$
    end for
    // Calculate probability for each roll-out
    for each $k \in K$ do
        $P_k = \frac{S_k}{\sum_{k=1}^{K} S_k}$
    end for
    // Cost-weighted averaging
    $\delta wp_{k,j} = \sum_{k=1}^{K} (P_k \cdot \epsilon_k)$
    // Update policy parameters
    $wp_{k,j} \leftarrow wp_{k,j} + \delta wp_{k,j}$
    // Decay exploration noise
    $\sigma^2_{PI^{BB}} = \gamma \cdot \sigma^2_{PI^{BB}}$
end while
```

Noise is given as Gaussian noise with an exploration noise of $\sigma^2_{PI^{BB}} = 0.015$, while K, representing the number of roll-outs, is set to 8. Both the amount of exploration noise and number of roll-outs are empirically chosen to promote fast and stable learning. The exploration noise is further linearly decayed during learning using a decay constant of $\gamma = 0.995$. This adds an additional free parameter to the PI^{BB} algorithm. Decaying the exploration noise allows for large weight changes at the beginning of the learning process and small changes or fine-tuning toward the end. Finally, we use the reward function

$$R_k = w_1 \cdot distance - (w_2 \cdot instability + \left(\frac{1 - collision}{\max_{dist}}\right)^{w_3} + w_4 \cdot slippage), \tag{5}$$

where $w_1 = 3$, $w_2 = 1$, $w_3 = 20$, $w_4 = 1$ and $\max_{dist}$ is the distance between the legs from where the collision sub-reward is set to zero (i.e., having no influence). It is a function of four factors or sub-rewards: distance, instability, collision, and slippage. Distance is a measure of movement along the x-axis, and therefore, rewards fast straight walking. Instability is a measure of how stable the robot is during movement. It is calculated as the sum of variance in height (z-axis), absolute mean yaw or heading direction (0° is straight), absolute mean pitch, and absolute mean roll. A pitch and roll equal to 0° means that the robot is parallel with the floor. The three measurements are summed throughout an entire roll-out. Instability, therefore, penalizes movement that is not in the walking direction. Collision is a measure of the extent to which each leg of the robot collides its other legs. It is calculated as the exponential inverse distance between the legs of the robot, thus penalizing legs that are colliding or very close to each other. Finally, slippage is a measure of the extent to which each leg of the robot slips on the ground. It is calculated as foot movement during ground contact versus no foot movement during ground contact. A slippage return of 1 means that one or more legs are slipping all the time they are in contact with the ground. Note that different weights (w_1 to w_4) are empirically assigned to the four sub-rewards to make them equal in magnitude and range. Furthermore, instability and collision are thresholded at 1.5 and 1, respectively. This is to avoid the negative returns being too large. For (5), the walking distance can be seen as the dominating reward.

VI. Experiments

A. Robot platforms

The legged robot platforms shown in Fig. 4 (Alpha, Laikago, and MORF) were all used to assess the performance of the generic locomotion control framework. All these were simulated using a general-purpose robot simulation framework called V-REP [14]. The simulated environment offers real-world parameters for a large number of physical properties, making it realistic and accurate.

1) Alpha: Alpha is a hexapod robot developed at the University of Southern Denmark (SDU). It is designed to mimic a dung beetle such that the kinematics and dimensions match on a scale of 1:22. The morphology of the robot is irregular, since the front, middle, and hind legs are different from each other and thus have different ranges of freedom (see Fig. 5). Each leg consists of three segments and three joints (J_1, J_2, and J_3). Furthermore, an additional joint is placed in the back of the robot but remains fixed for simplicity. The simulated model of the Alpha robot is shown in Fig. 4.

2) Laikago: Laikago is a quadruped robot developed by Unitree in China [36]. It is designed as a research platform

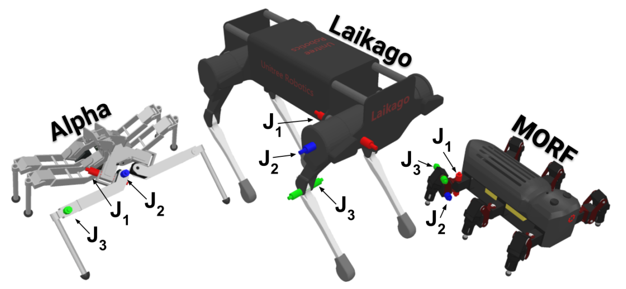


Fig. 4. The simulated Alpha, Laikago, and MORF robots. All three robots are equipped with three joints per leg (J_1, J_2, and J_3).

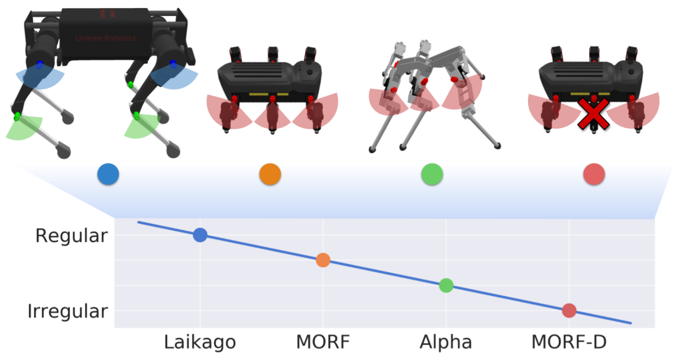


Fig. 5. The regularity of Laikago, MORF, Alpha, and MORF-D. MORF-D is MORF when damaged, which in this work means a broken middle leg (as indicated by the cross). The transparent areas illustrate the range of freedom for the robots such that the legs are not colliding with the body or other legs.

with four identical legs, each consisting of three joints (J_1, J_2, and J_3). Its morphology is regular, and all four legs have similar ranges of freedom (see Fig. 5). The simulated model of Laikago (provided by Unitree) is shown in Fig. 4.

3) MORF: The Modular Robot Framework (MORF) has been developed at SDU [37]. It is designed as a modular research platform for studies on legged locomotion. In this study, we used MORF in a compact hexapod configuration. In this configuration, MORF is regular with respect to leg morphology, since all legs are identical. However, it is irregular with respect to the position of the legs, i.e., the range of freedom for the middle leg is different from that of the front and hind legs which can move further forward or backward (see Fig. 5). Each leg consists of three joints (J_1, J_2, and J_3). The simulated model of the MORF robot is shown in Fig. 4.

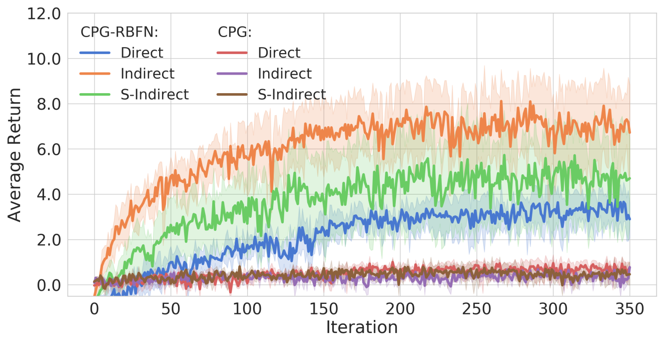
B. Experimental setup

Four experiments were performed to assess the performance of the generic locomotion control framework. In all experiments, PI^{BB} was used with the same variance of $\sigma^2_{PI^{BB}} = 0.015$ and number of roll-outs $K = 8$ (as explained in Section V). In each roll-out, the robots were simulated for five seconds, i.e., 40 seconds of simulated time in total per iteration. As mentioned previously, simulated time is the time used by the robot inside the simulated environment and is therefore independent of the CPU. From each roll-out, a return was calculated using the reward function presented in Section V. The same reward function was used for all legged robots with no prior knowledge given, except for the initial static poses shown in Fig. 4 and leg phase relationships (trot gait for Laikago and tripod for the hexapod robots). Note that while the leg phase relationship is fixed for indirect encoding, this is not the case for semi-indirect and direct encoding due to them being decentralized. The initial CPG-RBF network weights or policies were randomly initialized using Gaussian noise with twice the exploration noise of PI^{BB}. A total of 350 iterations were executed for each robot using direct, indirect, and semi-indirect encodings. Each experiment were repeated five times from where the average iteration return and its standard error was computed.

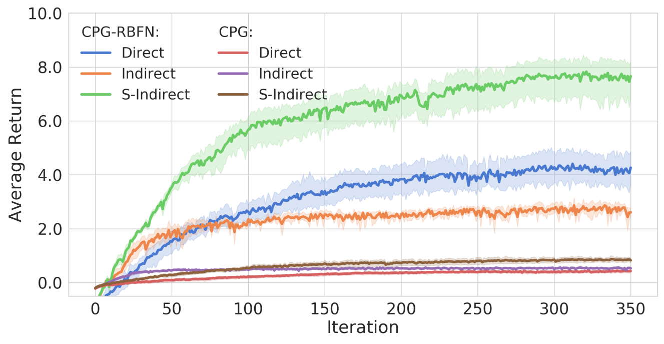
1) Learning locomotion control policy for forward walking: In the first experiment, locomotion control policies for the Laikago, Alpha, and MORF robots were learned in simulation. The purpose of this experiment was to verify that the CPG-RBF network is generic and investigate how the controller encoding relates to the morphology in terms of learning speed and maximum performance. To show the importance of the trajectory shape, we further compared the CPG-RBF network with a CPG without the RBF network (i.e., a fixed wave-shaped output). Both the amplitude and offset of the CPG were learned using PI^{BB} as for the CPG-RBF network. Like the CPG-RBF network, the CPG had no prior knowledge, except the initial poses shown in Fig. 4 and leg phase relationships (trot gait for Laikago and tripod for the hexapod robots). Note that while the CPG-RBF network starts learning from a static position with no movement, the CPG alone starts with its standard oscillating output, already enabling it to walk from iteration 0. The reason for this is that the shape of the CPG cannot be changed; thus, we start with some reasonable parameters and optimize those to get the best performance. In the CPG-RBF network, we start from a static position to avoid the introduction of a bias in the trajectory shape.

2) Learning locomotion control policy for damage compensation: In the second experiment, locomotion control policies for MORF with a broken middle leg (MORF-D) were learned in simulation. The purpose of this experiment is to show that the CPG-RBF network can also be applied to a damaged robot and investigate how the controller encoding should be changed to deal with such highly irregular cases (see MORF-D in Fig. 5). To see how the CPG-RBF network can make use of prior knowledge and how fast it is able to adapt after impairment, we also included an experiment in which the converged locomotion control policy from learning on the fully functional version of MORF was used as the starting policy. Both prior knowledge and the new policy were learned using semi-indirect encoding.

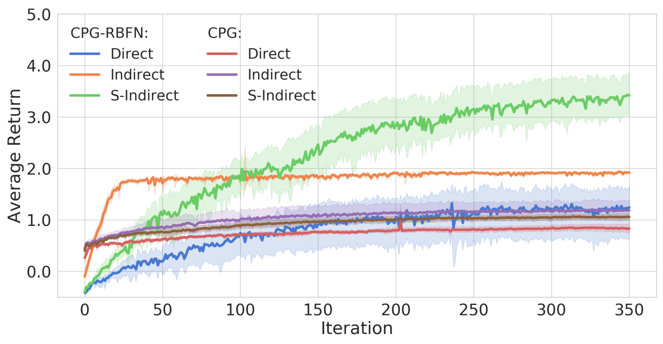
3) Deploying locomotion control policies on a physical robot: Finally, to show that the learned locomotion control policy works not only in simulation but also in the real world, a test was performed using the physical MORF. During continuous walking, MORF initially walked with six fully functional legs using the indirect encoded control policy. After one meter of walking the middle leg was virtually broken, setting it into a static position where it could not touch the ground. To enable MORF to continue walking, the control policy was swapped to the semi-indirect control policy learned



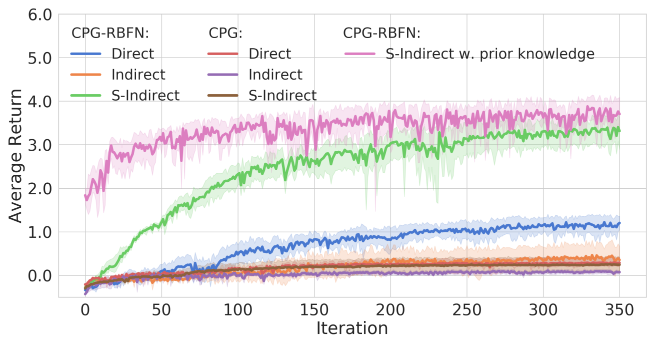
(a) Laikago



(b) Alpha



(c) MORF



(d) MORF-D

Fig. 6. Mean returns with standard error for learning the weights of the CPG-RBF network and the amplitude and offset of a CPG without the RBF network when using direct, indirect, and semi-indirect (s-indirect) encodings on (a) Laikago, (b) Alpha, (c) MORF, and (d) MORF-D. In (d), the mean return for learning the weights of the CPG-RBF network when using semi-indirect encoding and prior knowledge from learning using MORF with fully functioning legs is also presented. Note that one iteration consists of eight roll-outs, and each roll-out represents five seconds of simulated time. This means that one iteration takes 40 seconds of simulated time.

for MORF-D. After one meter of walking with a broken middle leg, it was again enabled, and the initial control policy swapped back in. The purpose of this experiment was to show that the policy can be successfully swapped online and the learned control policies for normal walking and leg damage work in the real world.

Note that the fourth experiment is placed in the supplementary material. In this experiment, we show that the CPG-RBF network can integrate sensory feedback from an inertial measurement unit to learn a body posture behavior that can stabilize MORF on uneven terrain.

VII. Results

A. *Learning locomotion control policy for forward walking*

The average return plots from learning the locomotion control policies of Laikago, Alpha, and MORF are shown in Figs. 6a, 6b, and 6c, respectively. The average sub-returns describing velocity (calculated from the distance), instability, collision, and slippage (as explained in (5), Section V) of the converged locomotion control policies are shown in Fig. 7. The figure also shows the cost of transport (COT) calculated as $\frac{P}{m \cdot g \cdot v}$, where m is the weight of the entire robot in kg, g is the gravity of earth ($9.82 m/s^2$), v is the walking velocity of the robot in m/s, and P is the power given as the joint torque in $N \cdot m$ times the angular joint velocity in rad/s. It should be noted that COT is a dimensionless measurement that quantifies the energy efficiency of transporting the legged robot from start to finish positions (i.e., the energy efficiency of the generated locomotion). Video clips of the three robots using the CPG-RBF network and the three encoding types with weights from different iterations can be found in the supplementary materials (videos 1.0, 2.0, and 3.0) or at https://youtu.be/vQ_vdE_fAfw (Laikago), https://youtu.be/-XrN7BBBywQ (Alpha), and https://youtu.be/7JOPKMk97lM (MORF). The learned joint trajectories for MORF when using semi-indirect encoding are shown in Fig. 8. The remaining learned joint trajectories using the three encoding types on all three robots for every iteration can be found as video clips in the supplementary materials (videos 1.1, 2.1, and 3.1) or at https://youtu.be/y-ntLATy9OQ (Laikago), https://youtu.be/UqmXiyjUQ2Y (Alpha), and https://youtu.be/RfCQQf6uvkA (MORF).

The results show that our generic framework can be used to learn walking patterns for robots with different morphology in a minimum simulated learning time of 13 minutes (MORF with indirect encoding, Fig. 6c) and a maximum simulated learning time of 133 minutes (MORF-D with semi-indirect encoding, Fig. 6d).

The convergence, when using semi-indirect or direct encoding is slower in comparison to indirect encoding (see Fig. 6). This is expected, since the number of policy parameters

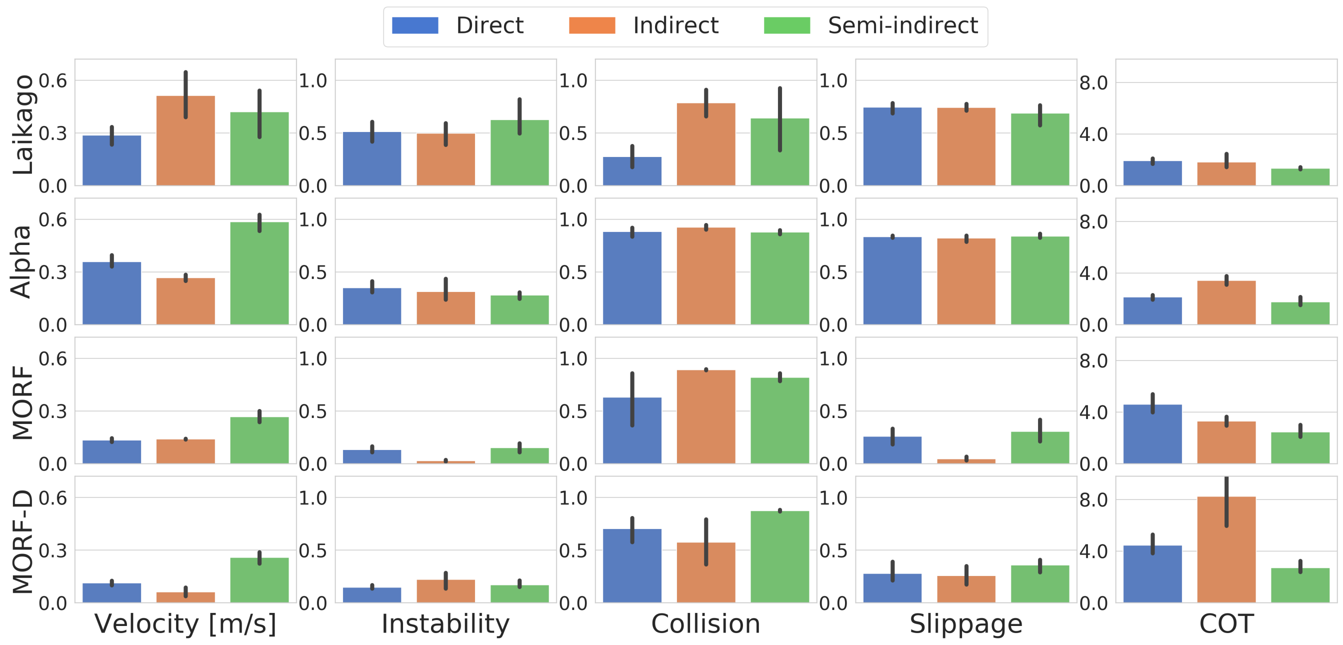


Fig. 7. Mean sub-returns and cost of transport (COT) with standard error for the learned policies when using direct, indirect, and semi-indirect (s-indirect) encodings on Laikago, Alpha, MORF, and MORF-D (five repetitions for each combination of encoding type and robot). Each column of bar plots shows the sub-returns for forward velocity (calculated from the distance), instability, collision, and slippage, respectively. The velocity is in m/s while the instability, collision, and slippage are measured as explained in (5). The last column shows the COT; a dimensionless measurement that quantifies the energy efficiency of transporting the legged robot from start to finish positions.

increases with the number of unique trajectories that need to be learned. For Laikago, which is the most regular robot of the three (see Laikago in Fig. 5), the converged policy return when using semi-indirect or direct encoding is smaller than the one obtained with indirect encoding (see Fig. 6a). Alpha, the most irregular robot among the three (see Alpha in Fig. 5), benefits significantly from semi-indirect encoding as can be seen from the larger converged policy return (see Fig. 6b). Unlike the two other robots, Alpha achieves a higher converged policy return using direct encoding, compared to indirect encoding. Finally, MORF, which is less regular than Laikago and more regular than Alpha (see MORF in Fig. 5), also benefits significantly from semi-indirect encoding (see Fig. 6c). For all three robots, semi-indirect encoding results in the most energy efficient locomotion (see COT in Fig. 7).

When comparing the standard error in the average return of the three robots, it is clear that Laikago has the largest standard error. The reason for this is that the predefined trot gait and the morphology of Laikago are not as stable as the tripod gait and the morphology of MORF and Alpha. This makes it possible for Laikago to roll over on its back, causing a large negative return value, which is also indicated in Fig. 7, whereas Laikago has a relatively high instability for all encodings. The general standard error is otherwise low, indicating consistent learning, even though it uses a stochastic learning mechanism. Fig. 7 and the videos of the Laikago and Alpha robots also show that some foot slippage was still present even though it was penalized in the reward function.

The results in Fig. 6 also show that when using a CPG without the RBF network, whereby only the amplitude and offset are learned, it is not possible for the robots to reach high performance. This can be observed from the overall low returns that, only in some cases, are able to compete with those of the CPG-RBF network. The CPG achieves the highest returns when used on MORF, but scores close to zero when used on the other robots. This is because Alpha is too irregular to obtain good solutions when using CPG alone and the fixed wave-shaped CPG output makes it hard for Laikago to have a flat, stable, and slip-free stance phase. In summary, the shape of the movement trajectory is crucial for the generation of locomotion patterns, especially for robots with highly irregular morphologies.

B. Learning locomotion control policy for damage compensation

The average return plot resulting from learning locomotion control policies on MORF with a broken middle leg (MORF-D) is shown in Fig. 6d. The video clip of MORF-D when using the CPG-RBF network and the three encoding types with weights from different iterations can be found in the supplementary material (video 4.0) or at https://youtu.be/1l3hD68Tx88. The video of the learned joint trajectories using the three encoding types for all iterations can likewise be found in the supplementary material (video 4.1) or at https://youtu.be/pk56YJ7NzM0.

The results show that our generic framework can be used to learn walking patterns for MORF-D. In this configuration, MORF-D is considered irregular (see Fig. 5). The fixed gait of indirect encoding resulted in MORF-D almost being unable to move as can be seen from the low returns. The ability to change the gait greatly benefits MORF-D, and therefore, both semi-indirect and direct encoding perform a lot better in terms

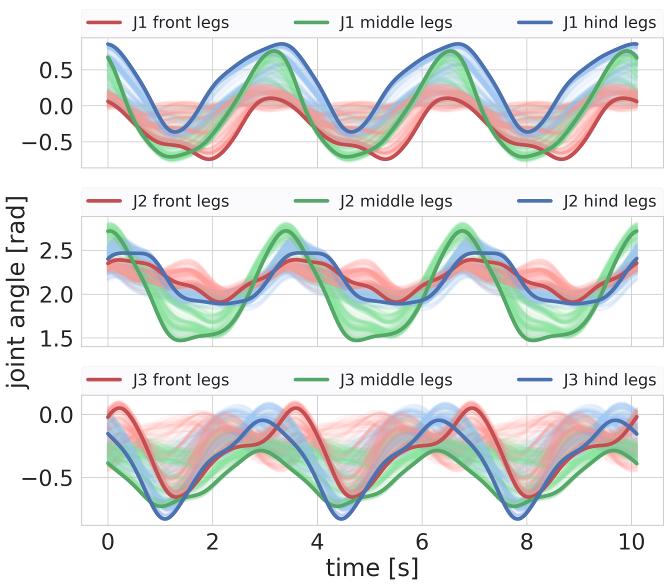


Fig. 8. The learned joint trajectory policies for MORF when using semi-indirect encoding. The solid lines show the trajectories of the three joints (J_1, J_2, and J_3) in the front, middle, and hind legs at iteration 350 and the transparent lines show the intermediate joint trajectories starting from the first iteration. The remaining learned joint trajectories when using the three encoding types on all three robots for every iteration can be found as video clips in the supplementary materials. Note that the figure shows the trajectories before applying the leg phase relationships (trot gait for Laikago and tripod for the hexapod robots).

of convergence speed and return. This is also why, when using the CPG without the RBF network, MORF-D gets returns close to zero. When using prior knowledge from MORF with fully functional legs, the convergence rate is much faster and only takes around 13 minutes of simulated time instead of 133 minutes.

C. Deploying locomotion control policies on a physical robot

In the third experiment, the learned control policy was deployed on the physical MORF and swapped online for damage compensation. Snapshots from the experiment can be seen in Fig. 9 and a video clip is available in the supplementary material (video 5.0) or at https://youtu.be/OIYxm5DvPOA.

The results show that the learned control policies successfully transfer to a physical robot platform without any further modifications. They also show that the control policy can be successfully and smoothly swapped online to deal with damages.

VIII. Discussion and Conclusion

In this paper, we have presented a generic framework for learning complex locomotion control policies for three different legged robots, including one with a damaged leg, using identical reward functions and no prior knowledge except the initial static poses and leg phase relationships. The framework is based on a neural controller called the CPG-RBF network, consisting of a combination of a CPG and an RBF network together with the optimization algorithm PI^{BB} for learning the weights of the CPG-RBF network.

The CPG-RBF network is scalable in terms of complexity and can be decentralized such that the number of unique joint trajectories to be learned can be varied. This provides a foundation for three different types of controller encoding: indirect where all legs use identical joint trajectories; semi-indirect where leg pairs (e.g., front, middle, and hind legs) use identical joint trajectories; and direct where all legs use unique joint trajectories. The results show that the choice of encoding relates to the morphology of the legged robot. Regular robots benefit more from indirect encoding than irregular robots, which gain greater benefit from semi-indirect and direct encoding. For example, Laikago, which is very regular, does not benefit from having different joint trajectories, as can be observed from the low return of semi-indirect and direct encoding. On the other hand, Alpha is very irregular, and therefore, benefits from using semi-indirect encoding. The fact that it is more irregular than the other two robots is evidenced by the high return of direct encoding and the relatively slower convergence of indirect encoding. Finally, MORF, which is between Laikago and Alpha with respect to regularity, benefits significantly from semi-indirect encoding. However, MORF is still relatively regular, which is also one reason why indirect encoding results in a higher return than direct. This finding is consistent with that found in [4]. The semi-indirect controllers achieved the highest return for all robots, except Laikago, at reasonable convergence rates. We believe this is due to the bilateral symmetry of legged robots. In terms of convergence speed, indirect controllers were the fastest while direct were the slowest. This is due to the number of control parameters increasing with the number of unique joint trajectories to learn. Thus, for systems that are not too irregular and where convergence speed is essential, indirect encoding should be used. This could, for example, be considered for real-world optimization [7], [8].

The only prior knowledge given to the robots is the predefined phase relations between the legs (i.e., gait) and the initial static poses. However, for semi-indirect and direct encodings, this gait may change during learning. With semi-indirect encoding, the phase between ipsilateral legs (from front to rear in forward walking) can change since the same trajectories, with a predefined phase shift, are learned for contralateral leg pairs (e.g., front, middle, and hind legs). With direct encoding, the phase between all legs can change as each leg learns its own trajectories. The results show that both Alpha and MORF learned semi-indirect and direct encoded locomotion policies that minimize the use of the middle legs (see supplemental videos 2.0 and 3.0). The reason for this is that distance is dominant in the reward function, and thus the speed of a trotting gait is preferred over the stability of a tripod gait.

The use of the identical reward function shows that our approach is not sensitive and, therefore, does not need precise tuning of the reward weights. However, the locomotion behaviors of Laikago and Alpha can be improved by adjusting the reward functions to better take into account slippage and stability. This is because Laikago and Alpha can cover long distances as a result of their long legs (see Fig. 7). Thus, the stability and slippage penalization terms have less impact on the total return since they are heavily dominated by the

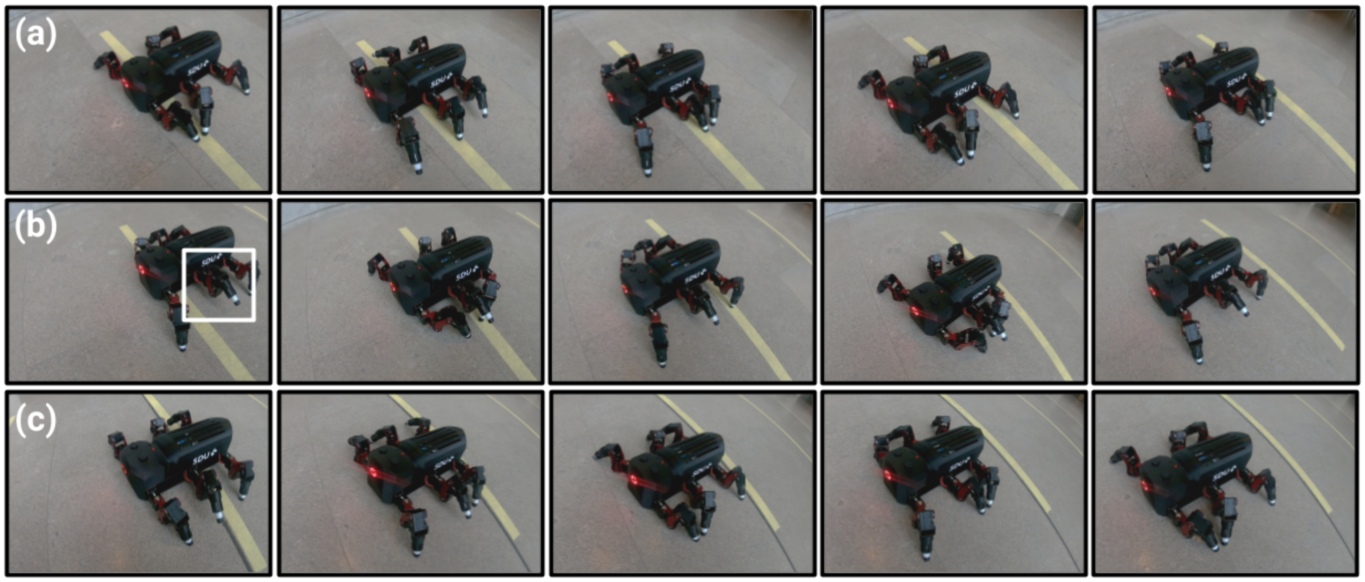


Fig. 9. Learned locomotion control policy from simulation applied directly to the physical MORF. (a) MORF is fully functional for the first meter of walking. (b) after one meter of walking the middle leg is virtually broken (as indicated by the white box) and the locomotion control policy is successfully and smoothly swapped to one that enables it to continue walking. (c) finally, after two meters of walking, the middle leg recovers, and the initial locomotion control policy is swapped back in.

distance. Another strategy for improving the locomotion for uneven terrain is shown in the supplementary material. Here sensory feedback from an inertial measurement unit is integrated into the CPG-RBF network for body posture adaptation and walking stability enhancement of MORF during walking on uneven terrain.

We have demonstrated that the proposed framework can learn complex locomotion control policies with a minimum simulated learning time of 13 minutes and a maximum simulated learning time of 133 minutes. The fast convergence is due to the low number of parameters needed by the CPG-RBF network. An approach similar to the CPG-RBF network is presented in [38], where arbitrary rhythmic signals are generated as a weighted linear combination of several CPGs with different amplitudes, frequencies, and phase shifts. In comparison, the CPG-RBF network uses fewer parameters, which is likewise true when also comparing to [3] and [4]. This means that the CPG-RBF network is able to learn faster and the network is easier to comprehend. Moreover, the CPG-RBF network consists of two decoupled components, i.e., it is more modular with a single CPG controlling the frequency of the rhythmic signal and the RBF network subsequently reshaping it. Without the RBF network, the CPG, with its fixed wave-shaped output, is not able to achieve the same high returns as the CPG-RBF network. This shows that the shape of the motion trajectories plays a crucial role in the robot's locomotion performance.

The simplicity of the CPG-RBF network makes it easy to analyze since it is straightforward to interpret the function of each synaptic weight or policy parameter. Nevertheless, the CPG-RBF network is still able to produce complex locomotion patterns, showing that a neural controller does not need to be large and complex to encode motor primitives for complex locomotion behaviors. These features can be seen from both the videos of robots in simulation and the learned joint trajectories. For example, in the video of Alpha, it can be seen that by using semi-indirect encoding, a highly complex gait was learned, where the middle legs delay their movement in the middle of the swing phase to make room for the front and hind legs.

The learned controller successfully transfers to a physical, real-world legged robot without any modification. When used on the physical robot, the results show that it can walk and even compensate for damage by swapping the locomotion control policy online. It thereby displays behaviors similar to those presented in [1] where an intelligent trial and error algorithm is used for adapting to robot damage using a pre-computed behavior-performance map that predicts the performance of thousands of control policies with different locomotion behaviors. The main difference between the frameworks is that in [1] many locomotion control policies are learned at random and put into a behavior-performance map for handling many different types of damage to a legged robot, whereas in our case, the policy is learned directly for a particular legged robot with specific damage. Another difference is the complexity of the control policies. In [1], the control policies are given by a periodic square signal, parametrized by its amplitude, phase, and duty cycle (the duty cycle is the proportion of one period in which the joint is in its higher position). In comparison, it is clear that the CPG-RBF network can encode more complex and smooth locomotion control policies since it is not only parametrized by the amplitude, phase, and duty cycle, but the entire shape is encoded. Therefore, one possible improvement to the framework in [1] would be to use the CPG-RBF network instead of the simple parametrized square signal control.

To summarize, in this work, we demonstrate that: 1) the proposed framework is simple, minimal, and intuitive to use and analyze; 2) it has few control parameters, resulting in fast learning – down to 13 minutes of simulated time; 3) the CPG-RBF network can be applied to locomotion generation for legged robots with different morphologies; 4) there is a correlation between controller encoding and the morphology of the legged system; 5) the learned locomotion policy can be deployed directly to a real-world robot as well as for damage compensation; and 6) it is possible to integrate sensory feedback into the CPG-RBF network. These features lead to a scalable and easy-to-use framework with fast convergence and the ability to encode complex locomotion control policies.

The proposed locomotion control framework has the potential to facilitate many exciting future studies. Firstly, since our framework can learn a control policy for a legged robot within 13 minutes of simulated time, it would be interesting to investigate whether it is possible to make the framework adapt the locomotion control policy online for continuous autonomous lifelong learning. In this case, one could deal with damage without the need for a pre-computed map, or an intelligent trial and error algorithm as in [1]. For example, by continuously estimating the return from the reward function (5)[1], a sudden decrease in the return could trigger relearning. Such a decrease could be the result of damage to the robot (e.g., a broken leg) or a change in walking environment.

In future work, we also plan to thoroughly investigate the meta parameters of the CPG-RBF network (H and σ^2_{RBF}) to see if they are robot and task specific. We will also apply, e.g., an information theory approach [41], [42] for meta parameter learning to speed up the overall learning process such that we can transfer it to a real robot, allowing it to efficiently learn and adapt its locomotion behavior online toward continuous lifelong autonomous learning. An alternative way of increasing the performance of the CPG-RBF network and rate of convergence is to make use of controller encoding switching like HybrID (switching from FT-NEAT to HyperNEAT) in [4]. The approach would begin by learning with indirect encoding and then switch to semi-indirect encoding once indirect encoding converges. In this way, the complexity of the control policy remains the same.

In the supplementary material, we show how the CPG-RBF network can integrate sensory feedback and become a closed-loop network that can adapt to the environment. In future work, we will introduce other types of sensory feedback modules to further shape the joint trajectories online and improve the adaptability to uneven and unpredictable terrains. One example would be to use foot contact force [43] or joint position feedback [39], [40] to modulate the trajectory in order to stop lowering a leg if it steps on an obstacle or keep lowering it if stepping into a hole [44]. By having the sensory feedback as extensions and employing a sensor fault detection algorithm like the one presented in [45], it is possible to stop using faulty sensors, thereby preventing them from affecting the trajectory of the original CPG-RBF network. Subsequently, it would then be possible to re-train the network to rely on other sensors if redundancy is present.

In this work, the policies were learned to optimize locomotion speed and stability. In future work, we plan to include other factors such as energy efficiency. We also intend to consider learning locomotion patterns for more complex environments such as uneven ground, stairs, slopes, and pipes.

[1]For continuous learning on the real robot, one can use different sensors to calculate the reward function. For example, distance can be estimated from a vision sensor (like RealSense T265 which provides visual odometric feedback), instability can be estimated from an inertial measurement unit, collision can be estimated from joint position sensors [39], [40], and slippage can be estimated from joint position sensors or foot contact sensors with forward models [29].

Acknowledgment

The authors would like to thank Peter Billeschou from the University of Southern Denmark for providing the Alpha robot simulation.

References

[1] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, p. 503, May 2015.

[2] M. Oliveira, L. Costa, A. Rocha, C. Santos, and M. Ferreira, "Multi-objective optimization of a quadruped robot locomotion using a genetic algorithm," in *Soft Computing in Industrial Applications*, A. Gaspar-Cunha, R. Takahashi, G. Schaefer, and L. Costa, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 427–436.

[3] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019. [Online]. Available: https://robotics.sciencemag.org/content/4/26/eaau5872

[4] J. Clune, K. O. Stanley, R. T. Pennock, and C. Ofria, "On the performance of indirect encoding across the continuum of regularity," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 3, pp. 346–367, June 2011.

[5] Jun Nakanishi, Jun Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "A framework for learning biped locomotion with dynamical movement primitives," in *4th IEEE/RAS International Conference on Humanoid Robots, 2004.*, vol. 2, Nov 2004, pp. 925–940.

[6] J. Rosado, F. Silva, and V. Santos, "Adaptation of robot locomotion patterns with dynamic movement primitives," in *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, April 2015, pp. 23–28.

[7] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," 2020.

[8] A. Marjaninejad, D. Urbina-Meléndez, B. Cohn, and F. Valero-Cuevas, "Autonomous functional movements in a tendon-driven limb via limited experience," *Nature Machine Intelligence*, vol. 1, pp. 144–154, 03 2019.

[9] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642 – 653, 2008.

[10] J. Yu, M. Tan, J. Chen, and J. Zhang, "A Survey on CPG-Inspired Control Models and System Implementation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 441–456, March 2014.

[11] M. Pitchai, X. Xiong, M. Thor, P. Billeschou, P. L. Mailänder, B. Leung, T. Kulvicius, and P. Manoonpong, "CPG driven RBF network control with reinforcement learning for gait optimization of a dung beetle-like robot," in *Artificial Neural Networks and Machine Learning – ICANN 2019: Theoretical Neural Computation*, I. V. Tetko, V. Kůrková, P. Karpov, and F. Theis, Eds. Cham: Springer International Publishing, 2019, pp. 698–710.

[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley, 1989.

[13] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sep. 2004, pp. 2149–2154 vol.3.

[14] M. F. E. Rohmer, S. P. N. Singh, "V-REP: A Versatile and Scalable Robot Simulation Framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[15] S. Schaal, *Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics*. Tokyo: Springer Tokyo, 2006, pp. 261–280.

[16] A. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, 2013.

[17] S. Aoi, P. Manoonpong, Y. Ambe, F. Matsuno, and F. Wörgötter, "Adaptive control strategies for interlimb coordination in legged robots: A review," *Frontiers in Neurorobotics*, vol. 11, p. 39, 2017.
[18] T. Nachstedt, C. Tetzlaff, and P. Manoonpong, "Fast dynamical coupling enhances frequency adaptation of oscillators for robotic locomotion control," *Frontiers in Neurorobotics*, vol. 11, p. 14, 2017.
[19] T. G. Brown, "On the nature of the fundamental activity of the nervous centres; together with an analysis of the conditioning of rhythmic activity in progression, and a theory of the evolution of function in the nervous system," *The Journal of Physiology*, vol. 48, no. 1, pp. 18–46, 1914.
[20] J. Hellgren, S. Grillner, and A. Lansner, "Computer simulation of the segmental neural network generating locomotion in lamprey by using populations of network interneurons," *Biological Cybernetics*, vol. 68, no. 1, pp. 1–13, Nov 1992.
[21] Ö. Ekeberg, "A combined neuronal and mechanical model of fish swimming," *Biological Cybernetics*, vol. 69, no. 5, pp. 363–374, Oct 1993.
[22] M. Thor and P. Manoonpong, "A Fast Online Frequency Adaptation Mechanism for CPG-Based Robot Motion Control," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3324–3331, Oct 2019.
[23] F. Pasemann, M. Hild, and K. Zahedi, "SO(2)-Networks as Neural Oscillators," in *Proc. of Computational Methods in Neural Modeling*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 144–151.
[24] F. Pasemann and N. Stollenwerk, "Attractor switching by neural control of chaotic neurodynamics," *Network: Computation in Neural Systems*, vol. 9, no. 4, pp. 549–561, 1998, pMID: 10221579.
[25] F. Pasemann, "Complex dynamics and the structure of small neural networks," *Network: Computation in Neural Systems*, vol. 13, no. 2, pp. 195–216, 2002, pMID: 12061420.
[26] S. Steingrube, M. Timme, F. Wörgötter, and P. Manoonpong, "Self-organized adaptation of a simple neural circuit enables complex robot behaviour," *Nature Physics*, vol. 6, p. 224, jan 2010.
[27] P. Manoonpong, U. Parlitz, and F. Wörgötter, "Neural control and adaptive neural forward models for insect-like, energy-efficient, and adaptable locomotion of walking machines," *Frontiers in Neural Circuits*, vol. 7, p. 12, 2013.
[28] M. Thor and P. Manoonpong, "A fast online frequency adaptation mechanism for CPG-based robot motion control," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3324–3331, Oct 2019.
[29] M. Thor and P. Manoonpong, "Error-based learning mechanism for fast online adaptation in robot motor control," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2019.
[30] D. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional interpolation and adaptive networks," *Royal Signals and Radar Establishment MALVER (United Kingdom)*, vol. RSRE-MEMO-4148, 03 1988.
[31] L. Righetti, J. Buchli, and A. J. Ijspeert, "Dynamic Hebbian learning in adaptive frequency oscillators," *Physica D: Nonlinear Phenomena*, vol. 216, no. 2, pp. 269 – 281, 2006.
[32] F. Stulp and O. Sigaud, "Policy improvement methods: Between black-box optimization and episodic reinforcement learning," *Archive ouverte HAL*, p. 34, 10 2012.
[33] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *J. Mach. Learn. Res.*, vol. 11, pp. 3137–3181, Dec. 2010.
[34] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, Jun. 2001.
[35] S. Chatterjee, T. Nachstedt, M. Tamosiunaite, F. Wörgötter, Y. Enomoto, R. Ariizumi, F. Matsuno, and P. Manoonpong, "Learning and chaining of motor primitives for goal-directed locomotion of a snake-like robot with screw-drive units," *International Journal of Advanced Robotic Systems*, vol. 12, no. 12, p. 176, 2015.
[36] Unitree. Laikago. [Online]. Available: http://www.unitree.cc/
[37] M. Thor, J. C. Larsen, and P. Manoonpong, "MORF – Modular Robot Framework," in *Proc. of The 2nd Int. Youth Conf. of Bionic Engineering (IYCBE2018).* Frontiers, Nov. 2018, pp. 21–23.
[38] L. Righetti and Auke Jan Ijspeert, "Programmable central pattern generators: an application to biped locomotion control," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 1585–1590.
[39] J. Faigl and P. Čížek, "Adaptive locomotion control of hexapod walking robot for traversing rough terrains with position feedback only," *Robotics and Autonomous Systems*, vol. 116, pp. 136 – 147, 2019.
[40] M. Palankar and L. Palmer, "A force threshold-based position controller for legged locomotion," *Autonomous Robots*, vol. 38, 03 2014.
[41] S. Dasgupta, F. Wörgötter, and P. Manoonpong, "Information dynamics based self-adaptive reservoir for delay temporal memory tasks," *Evolving Systems*, vol. 4, pp. 235–249, 05 2013.
[42] S. Herzog, C. Tetzlaff, and F. Wörgötter, "Evolving artificial neural networks with feedback," *Neural Networks*, vol. 123, pp. 153 – 162, 2020.
[43] X. Xiong, F. Wörgötter, and P. Manoonpong, "Neuromechanical control for hexapedal robot walking on challenging surfaces and surface classification," *Robotics and Autonomous Systems*, vol. 62, no. 12, pp. 1777 – 1789, 2014.
[44] P. Manoonpong, U. Parlitz, and F. Wörgötter, "Neural control and adaptive neural forward models for insect-like, energy-efficient, and adaptable locomotion of walking machines," *Frontiers in Neural Circuits*, vol. 7, p. 12, 2013.
[45] I. Hashlamon and K. Erbatur, "Joint sensor fault detection and recovery based on virtual sensor for walking legged robots," in *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, June 2014, pp. 1210–1214.

Mathias Thor received an M.Sc. degree in Robot Systems from the University of Southern Denmark, Odense, Denmark, in 2019. He is currently pursuing a Ph.D. degree with SDU Embodied Systems for Robotics and Learning at the University of Southern Denmark. His current research interests include neural locomotion control of walking machines, learning/plasticity, dynamic simulations, and design of legged robotic systems including their software interface.

Tomas Kulvicius received his Ph.D. degree in Computer Science (2010) from the University of Göttingen, Germany. For his Ph.D. thesis, he investigated the development of receptive fields in closed loop learning systems. From 2010 to 2015, Tomas was a researcher at the University of Göttingen where he worked on the trajectory generation and motion control of robotic manipulators. From 2015 to 2017, he was appointed as an Assistant Professor at the Centre for Bio Robotics, University of Southern Denmark. Currently he is a Research Assistant at the University of Göttingen, Germany. His research interests include the modeling of closed-loop behavioral systems, robotics, artificial intelligence, machine learning algorithms, movement generation, and trajectory planning.

Poramate Manoonpong received a Ph.D. degree in Electrical Engineering and Computer Science from the University of Siegen, Siegen, Germany, in 2006. He was the Emmy Noether Research Group Leader for Neural Control, Memory, and Learning for Complex Behaviors in Multisensory-Motor Robotic Systems with the Bernstein Center for Computational Neuroscience, Georg-August Universität Göttingen, Göttingen, Germany, from 2011 to 2014. Currently, he is a professor at the College of Mechanical and Electrical Engineering at Nanjing University of Aeronautics and Astronautics (NUAA), China. He is also an invited Professor at the School of Information Science & Technology, at Vidyasirimedhi Institute of Science & Technology (VISTEC), Thailand, and serves as an Associate Professor of Embodied AI & Robotics at the University of Southern Denmark (SDU), Denmark. His current research interests include: embodied AI, machine learning for robotics, the neural locomotion control of walking machines, biomechanics, dynamics of recurrent neural networks, learning/plasticity, embodied cognitive systems, prosthetic and orthopedic devices, exoskeletons, brain-machine interface, human-machine interaction, and service/inspection robots.