scientific reports

Check for updates

OPEN

# Hierarchical reinforcement learning with central pattern generator for enabling a quadruped robot simulator to walk on a variety of terrains

Toshiki Watanabe[1,5], Akihiro Kubo[2,5], Kai Tsunoda[1,5], Tatsuya Matsuba[3], Shintaro Akatsuka[3], Yukihiro Noda[3], Hiroaki Kioka[3], Jin Izawa[3], Shin Ishii[1,2]✉ & Yutaka Nakamura[4]

**We present a data-driven deep reinforcement learning (DRL) method for the optimization of a hierarchically structured control policy that includes the central pattern generator. This method, which is as a whole referred to as the hierarchical reinforcement learning with the central pattern generator (HRL-CPG), is then evaluated with the expectation of its applicability in real robot controls. We observed that stable gait motions were gained in a reasonably small number of trials and errors. Thus, it can be deduced that our HRL-CPG can be a candidate DRL method that enables dynamical systems such as real or realistic robots to adapt to a variety of environments within a moderate physical time.**

Autonomous mobile robots are anticipated to play a crucial role in addressing labor shortages in the near-future society, particularly in factories specializing in high-mix, low-volume production and in personal logistics. This issue is expected to be particularly acute in developed countries. These robots must possess navigation capabilities not only in structured environments such as factories and flat roads but also in unstructured environments common in daily life, including slopes, steps, and gravel roads. In this context, quadruped robots are gaining significant attention for their potential to transport heavy loads while overcoming slopes and steps[1–4]. Due to the environment-dependent nature of efficient gait patterns, there is a need for mechanisms that can autonomously and efficiently optimize gait patterns to adapt to new environments.

Reinforcement Learning (RL) is a machine learning framework that enables robots to autonomously learn control policies through trial and error[5]. It differs from Model Predictive Control (MPC), which is an optimization-based control approach using an accurate target model[6,7]. With advancements in deep learning, Deep Reinforcement Learning (DRL) has been successfully applied to various tasks[8–10]. The basic principle of DRL is straightforward: represent the control policies and their value functions using deep neural networks[11]. This approach can also be applied to model-based RL, where the model is explicitly approximated, offering methods similar to MPC[12–15]. However, learning policies and value functions for combinations of high degrees of freedom in state and control requires substantial trial and error, which is why RL and DRL are primarily used for tasks executed on computers, such as in robot simulators[16,17].

Unlike computer simulations, conducting a large number of operations with an actual robot in each environment is often impractical. This is due to the necessity for human intervention, such as resetting the robot to its initial state and managing any breakdowns[18]. To address this issue and improve the efficiency of robotic operations, apart from advancements in reinforcement learning (RL) algorithms[19–22], two main strategies have emerged.

The first approach, known as sim2real, involves training a robot in a simulated environment and then transferring the learned behaviors to a real-world robot[23–26]. This method takes advantage of the flexibility and safety of simulation to improve the real robot's performance. However, this technique is challenging because it requires highly accurate simulators that can mimic complex real-world phenomena such as contact interactions and material deformations. Effective calibration techniques[27–30] are essential to bridge the gap between the dynamics and kinematics of simulated models and real-world conditions, and domain randomization techniques

[1]Kyoto University, Kyoto, Japan. [2]Advanced Telecommunications Research Institute, Kyoto, Japan. [3]AISIN Corporation, Kariya, Japan. [4]Guardian Robotics Project, RIKEN, Wako, Japan. [5]Toshiki Watanabe, Akihiro Kubo and Kai Tsunoda contributed equally to this work. ✉email: ishii@i.kyoto-u.ac.jp

are widely employed to provide robots with diverse experiences[31,32]. These issues must be addressed to ensure that policies trained in a simulator can perform reliably when deployed on physical robots.

An alternative approach is to incorporate prior knowledge into the control policy function, effectively narrowing down the search space[33–37]. One typical method for achieving this is through the use of central pattern generator (CPG)-based methods, which are inspired by the locomotion mechanisms observed in animals[38,39]. CPG-based methods are distinguished by their ability to achieve robust locomotion that effectively adapts to environmental changes and disturbances, utilizing the characteristic entrainment provided by CPGs[40–42].

For instance, Nakamura et al.[35] illustrated how the integration of reinforcement learning (RL) with central pattern generators (CPGs) can produce a well-suited gait for simplified biped robot simulations, adaptable to diverse terrains. The pursuit of strategies for developing multiple gaits continues to progress, as evidenced by recent works[43–45]. A notable contribution in this field is by Bellegarda et al.[46], who utilized an innovative domain randomization technique. Through extensive trial and error in simulated environments, they achieved robust walking capabilities for robots navigating unstructured spaces populated with obstacles. These advances mark significant steps towards more adaptable and resilient robotic locomotion.

This study aims to develop a mechanism enabling a quadruped robot to adapt to new environments with a minimal number of trials and errors in the target environment. To achieve this, it is essential to easily design initial gait parameters and employ sample-efficient algorithms that adjust those parameters to the environment. A CPG controller is characterized by three key factors: the timing of leg movements (phase), movement patterns (waveform), and the speed of joint motions (frequency). These factors can be empirically derived, for example, by referencing model animals[42,47]. However, when implementing a CPG controller with neural oscillators, determining initial parameters that approximately reproduce the given features is not easy, due to the unknown influence of each parameter on the output, and the expected performance relies heavily on expert intuition[48]. Therefore, we adopt phase oscillators, whose parameters directly correspond to the three properties above. This study proposes a framework for tuning these interdependent parameters through RL.

We evaluate the performance of the proposed method using a quadruped robot simulator on a physics engine Bullet[49]. The aim of the control task evolves walking without deviating from an aisle of a certain width. The performance was tested on five types of floor conditions: even surface, up and down slopes with different friction values. The experimental results showed that suitable gait patterns were obtained within $5.0 \times 10^5$ control steps (corresponding to walking $3.0 \times 10^4$ steps and $1.0 \times 10^4$ seconds for 1,500 trials) on average.

This paper is organized as follows. Section 2 provides a concise explanation of the control method based on CPGs and RL, which form the foundation of the proposed approach. Section 3 describes the proposed method, in particular, the definition of the policy and the learning algorithm. In Section 4, demonstrating the effectiveness and limitations of our approach are presented by performing experiments using a quadruped robot simulator. Section 5 concludes our study with discussions.

## Backgrounds

In the spinal cord in vertebrates and invertebrates, there is a hypothetical neural circuit referred to as the central pattern generator (CPG). Referring to this mechanism, studies have been conducted on rhythmic movements such as locomotion and swimming[38,41,42]. The use of CPGs is advantageous even for artificial systems because the induced entrainment to low-dimensional limit-cycles would make the motions robust against the perturbations. For example, even when an external perturbation (e.g., misalignment of landing timing) is applied, the motion immediately recovers to the periodic one thanks to the interactions between the oscillators.

Figure 1 (upper left) shows the block diagram of the control using a CPG. In this study, the robot is assumed to change its state according to the dynamics

$$\frac{d\mathrm{s}_r}{dt} = f_r(\mathrm{s}_r, \mathrm{a}_r), \tag{1}$$

where $\mathrm{s}_r$ and $\mathrm{a}_r$ denote a state and control signal of the robot, respectively. $\mathrm{s}_r$ consists of physical quantities such as posture and position. In this study, sensor observations are used instead of state variables, and for simplicity, partial observability is not considered.

In CPG-based control, control signals to the robot, $\mathrm{a}_r$ are output from the CPG controller. A CPG is a dynamic system that changes its state $\mathrm{s}_p$ according to the following equation

$$\frac{d\mathrm{s}_p}{dt} = f_p(\mathrm{s}_p, \mathrm{a}_p). \tag{2}$$

Here, $\mathrm{a}_p$ is the feedback signal to adjust the phase of the oscillator in the CPG. The controller outputs the control signal to the robot $\mathrm{a}_r$ as

$$\mathrm{a}_r = g_p(\mathrm{s}_p). \tag{3}$$

In this study, the output of the CPG represents the target posture of the robot. That is, one oscillator corresponds to one joint, and the output of the oscillator is input to the robot as the target joint angle. After the robot receives the target angle, a PID control with a higher control frequency is employed to drive the low-level actuator of each joint.

Figure 1 (bottom) shows the temporal change of the state of CPG. The horizontal and vertical axes are the time and phase, respectively. The phase is converted into the target joint angle of each joint by a waveform modulation function. The upper right figure indicates the target pose of a leg when the phase is given.

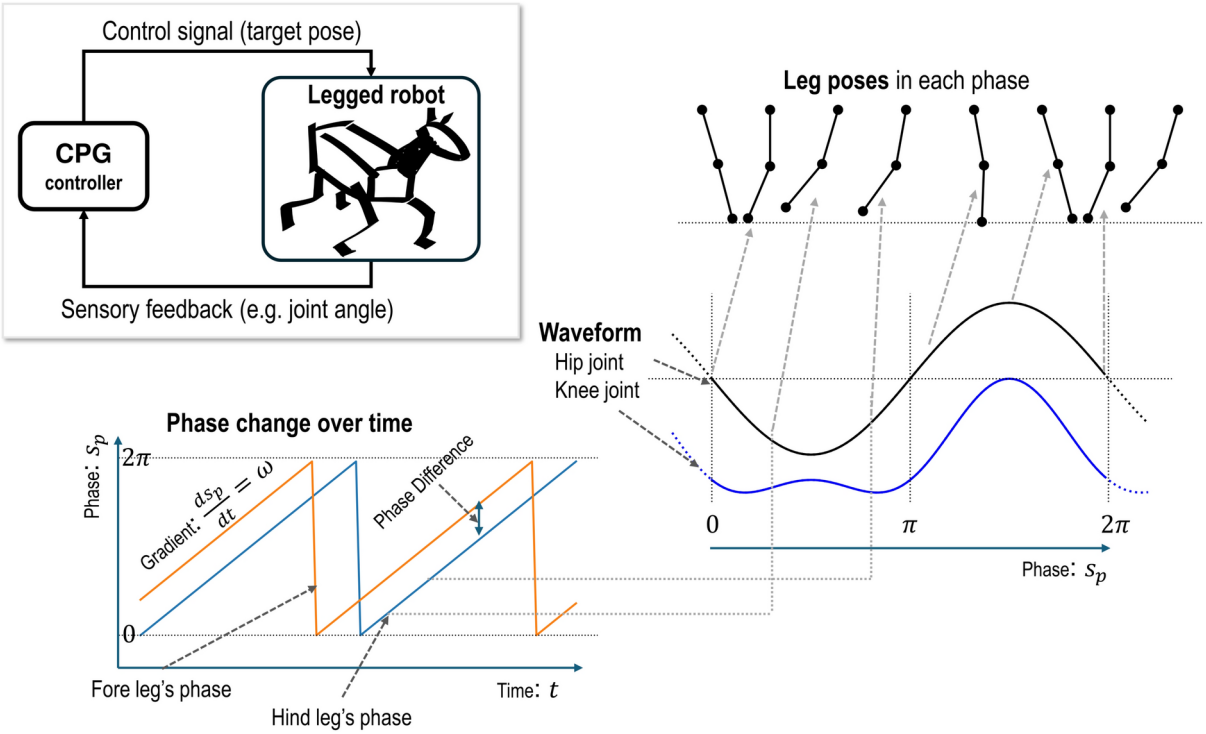



**Fig. 1**. Schematic diagram of the applied phase oscillator network (PON)

The gait of the robot changes depending on the following factors. 1) the phase relationship between joints, 2) the length of the gait cycle, and 3) the waveform. By adjusting these characteristics, the robot can modify its gait to adapt to its environment.

## Oscillator model

In this study, we employ a phase oscillator model presented by Kuramoto[50]; because the Kuramoto model is simpler than other oscillator models, such as that presented by Matsuoka[51]. Although both the Kuramoto and Matsuoka models have been used to generate the locomotive motions of animal-like architects[35,42,52–54], the parameters that determine the three aforementioned characteristics are clear in the former model. With the usage of the former model, our HRL-CPG, which will be introduced in the next section, becomes simpler.

In a phase oscillator network (PON), there are multiple oscillators (phasers), each of which has a characteristic frequency, and their phases are affected by the interaction expressed as a sinusoidal function between the oscillators. Let $s_p^{(i)}$ be the phase of the $i$-th oscillator, $i = 1, \cdots, N$, where $N$ is the number of oscillators, then their network PON is expressed by

$$\frac{ds_p^{(i)}}{dt} = \omega^{(i)} + \frac{K}{N}\sum_{j=1}^{N} \sin\left(s_p^{(j)} - s_p^{(i)}\right) + a_p^{(i)}, \tag{4}$$

for $i = 1, \cdots, N$, where $\omega^{(i)}$ corresponds to the characteristic frequency of the $i$-th oscillator, and $K$ is the coupling strength. $a_p^{(i)}$ denotes the external input to the $i$-the oscillator that coordinates the state of the oscillator with the external system.

Figure 2 shows the behavior of a PON with two oscillators. (A) shows the temporal change of the states of the oscillators, i.e., phases $s_p^{(i)}$. When an intermittent impulsive external input is applied, a phase shift occurs between the two oscillators. However, they then behave as if the phase difference converges to 0 due to the entrainment. (B) shows the temporal change of the output. The output of the oscillator is defined by a periodic function with the internal state as input, for which a trigonometric function is used. In this example, the waveform function is defined as $\mu^{(i)} = \sin(s_p^{(i)})$.

## Reinforcement learning

RL is a machine learning framework that allows a learning agent to make optimal decisions. The agent takes an action at the observed state, then observes a new state and a consequent reward. The objective of RL is to identify the optimal decision-making agent, i.e., the optimal policy, that maximizes the expectation of the reward summation (return) at any state.

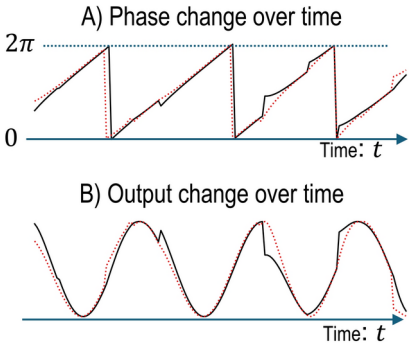



**Fig. 2**. Behavior of the PON. The dashed black line shows example behavior of one unit (phaser) in PON; the dotted line shows another. (A) State. (B) Output

Although the robot and CPG are treated as continuous-time differential dynamical systems i.e., Eqs. (1) and (2), they are approximated as discrete-time difference dynamical systems, for the sake of simplicity. That is, the state transition is defined by the following probability:

$$p(\mathrm{s}'|\mathrm{s},\mathrm{a}), \tag{5}$$

where $\mathrm{s}'$, s, and a are the destination state, the current state and the action taken, respectively.

The agent starts from an initial state $\mathrm{s}_0 \sim p_0(\mathrm{s}_0)$. At each state $\mathrm{s}_t$ at a discretized time *t*, it takes an action $\mathrm{a}_t$ sampled from a probabilistic policy $\pi(\mathrm{a}_t \mid \mathrm{s}_t)$, and receives a reward $r(\mathrm{s}_t, \mathrm{a}_t)$. Then, the system transitions to a new state $\mathrm{s}_{t+1}$ according to its dynamics $p(\mathrm{s}_{t+1}|\mathrm{s}_t, \mathrm{a}_t)$.

A trajectory of states and actions $\tau := (\mathrm{s}_0, \mathrm{a}_0, \mathrm{s}_1, \mathrm{a}_1, \cdots)$ is thus produced by its probability density

$$\rho_\pi(\tau) = p_0\left(\mathrm{s}_0\right) \prod_t \pi_t\left(\mathrm{a}_t \mid \mathrm{s}_t\right) p\left(\mathrm{s}_{t+1} \mid \mathrm{s}_t, \mathrm{a}_t\right).$$

While the standard RL often optimizes the expected return $\mathbb{E}_{\tau\sim\rho_\pi}\left[\sum_t r(\mathrm{s}_t, \mathrm{a}_t)\right]$, in this study, we consider an entropy-based regularized version, such as the one used in the soft Q-learning[55] and the soft actor-critic (SAC)[21] algorithms:

$$J(\pi) := \mathbb{E}_{\tau\sim\rho_\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left\{r\left(\mathrm{s}_t, \mathrm{a}_t\right) - \alpha \log \pi(\mathrm{a}_t|\mathrm{s}_t)\right\}\right], \tag{6}$$

where $0 \leq \gamma < 1$ is a discount factor.

In Eq. (6), the first term is the expected return and the second one is the entropy which encourages the agent to try exploratory behaviors that have a relatively low chance according to the policy without the entropy term. $\alpha$ is a temperature hyperparameter that controls the relative importance of the exploration against the exploitation in terms of return maximization. In the limit of $\alpha \to 0$, the above objective is reduced to the conventional RL objective. The details of the RL algorithm used in this study is explained in Appendix A.

One of the important challenges in applying RL to robot control problems is sample efficiency. Especially in the early stages of learning, collisions and falls are expected to occur frequently. In such cases, human intervention and even repair would be required. Bellow, we propose an efficient learning method by constraining the action space by prior knowledge and discuss its effectiveness.

## Hierarchical reinforcement learning for a CPG controller

To allow a quadruped robot to produce stable locomotions, we propose a hierarchical learning framework of the controller, inspired by the periodic motion generating mechanism in animals. We refer to the framework as the hierarchical RL with central pattern generator (HRL-CPG), because it is a hierarchical RL for the coupled dynamical system which involves a PON, a type of CPG.

By naively introducing a CPG network to the RL-based controller as in Figure 3(a), the policy becomes a non-stationary stochastic policy:

$$\pi_r(\mathrm{a}_r|\mathrm{s}_r) \sim p(\mathrm{a}_r|\mathrm{s}_p), \tag{7}$$

since it depends on the internal state of the policy, i.e., the state of the CPG network $\mathrm{s}_p$. Note that this state $\mathrm{s}_p$ depends on the history of state of robot $\mathrm{s}_r$, because of the CPG dependence on the sensory feedback from the robot. Thus, the Markov property, one of the assumptions normally used in RL, is no longer satisfied.

To address this non-Markovian character, the PON is divided into two parts, i.e., the internal state and the input. The former is treated as a part of the control target along with the robot and constitutes a coupled

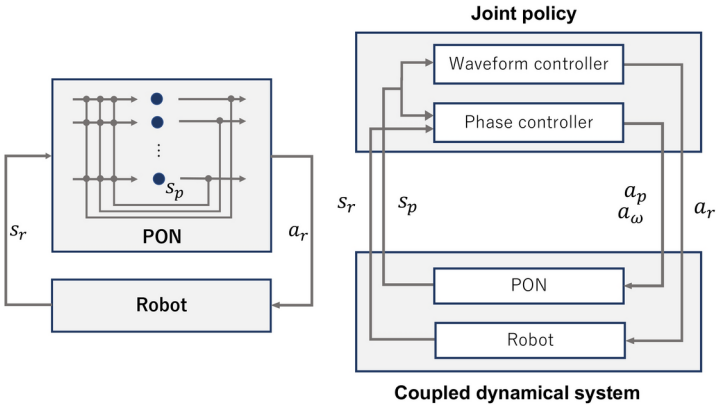



**Fig. 3**. Schematic diagrams of the CPG-based control. (**a**) Control architecture for the robot using a phase oscillator network (PON). (**b**) Control architecture for the coupled dynamical system using a feed-forward neural network

dynamical system. The RL is thus used to optimize the behavior of the coupled system that is the combination of the physical system to be controlled and the CPG network, as shown in Fig. 3b.

The latter becomes the external input of the oscillators $\mathrm{a}_p$, the third term of the right-hand side of Eq. (4). To coordinate the robot state with the state of the CPG controller and between the oscillators inside the CPG, both the sensory feedback from the robot $\mathrm{s}_r$ and the PON state $\mathrm{s}_p$ are used to determine $\mathrm{a}_p$. In other words, the coupled dynamical system defined by the state transition probability

$$p(\mathrm{s}'_r, \mathrm{s}'_p | \mathrm{s}_r, \mathrm{s}_p, \mathrm{a}_r, \mathrm{a}_p, a_\omega) \tag{8}$$

is controlled by the joint policy

$$\pi(\mathrm{a}_r, \mathrm{a}_p, a_\omega | \mathrm{s}_r, \mathrm{s}_p),$$

where $a_\omega$ is a control signal to determine the length of the gait cycle, i.e., the first term of the right-hand side of Eq. (4). Note that, the state and action of the coupled dynamical system are defined as $\mathrm{s} = [\mathrm{s}_r, \mathrm{s}_p]$ and $\mathrm{a} = [\mathrm{a}_r, \mathrm{a}_p, a_\omega]$, respectively. The above state transition probability and joint policy can be conceptually rewritten as $p(\mathrm{s}'|\mathrm{s}, \mathrm{a})$ and $\pi(\mathrm{a}|\mathrm{s})$.

In the following subsection, we describe each of the policies comprising the joint policy, and Fig. 4 visualizes their respective roles.

## Control signal

$\mathrm{a}_r$, $\mathrm{a}_p$, and $a_\omega$ are the modulation signals of the waveform, the phase control signal to the PON, and the base frequency modulation signal, respectively. While each element of $\mathrm{a}_p$, i.e., $a_p^{(i)}$, adjusts the phase of the *i*-th oscillator in an individual manner, $a_\omega$ adjusts the time length of the cycle for all the oscillators. Both of which determine the right-hand side of Eq. (4); $\mathrm{a}_r$ determines the waveform.

*Phase control*

$\mathrm{a}_p$ is the phase control signal to the PON, the third term of the right-hand side of Eq. (4), and adjusts the phase relationship among oscillators in PON. The phase control policy for the *i*-th joint is implemented as a neural network:

$$\pi_p(a_p^{(i)} | \mathrm{s}; \phi_p) := \mathcal{N}(\mu_p^{(i)}, \sigma_p^{(i)}), \tag{9}$$

where $\mu_p^{(i)}$ is the mean value of the *i*th component of output $\mu_p(\mathrm{s}; \phi_{p,\mu}) : \mathrm{s} \to \mu_p \in \mathcal{R}^N$ calculated by a neural network with the parameter $\phi_{p,\mu}$. $\sigma_p(\mathrm{s}; \phi_{p,\sigma}) : \mathrm{s} \to \sigma_p \in (0, \infty)^N$ is the variance vector where $\phi_{p,\sigma}$ is the parameter to produce the variance.

*Base frequency modulation*

We assume that the characteristic frequency of each oscillator $\omega^{(i)}$, i.e., the first term of the right-hand side of Eq. (4), is common for all the oscillators, but is modulated as

$$\omega^{(i)}(t) := \omega_0 + \omega_1 a_\omega(t) \qquad (i = 1, \cdots, N), \tag{10}$$

where $\omega_0$ is the pre-determined basic frequency and $\omega_1 > 0$ is a scale factor for the frequency modulation.

$a_\omega(t)$ is the base frequency modulation signal that depends on the state s at time *t*, and is determined by the following procedure. First, we sample a $\tilde{a}_\omega \in \mathcal{R}$ from the standard normal distribution $\mathcal{N}(\mu_\omega, \sigma_\omega)$, which is

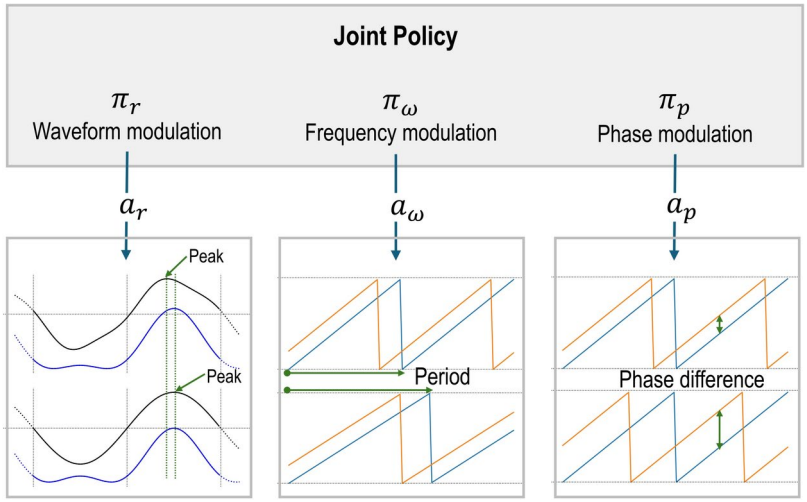



**Fig. 4**. Conceptual diagram illustrating the roles of the combined policies (waveform, frequency, and phase modulation) of the joint policy. Each policy's approach to signal generation and the ways in which it may change through learning are depicted as waveform examples for two joints (top row: before learning, bottom row: after learning). The waveform modulation policy outputs the target angles according to each joint's phase, and learning this policy can, for example, shift the peak timing of the periodic waveform. In contrast, the frequency and phase modulation policies produce a phase signal; the former adapts the signal's period through learning, while the latter adjusts the phase differences among the joints

defined in the same way as in Eq. (9), based on the neural network's outputs, $\mu_\omega$ and $\sigma_\omega$, with the parameter $\phi_\omega$ . Then we apply the hyperbolic tangent function to $\tilde{a}_\omega$ so that it takes a value between -1 and 1:

$$a_\omega := \tanh(\tilde{a}_\omega), \quad \tilde{a}_\omega \sim \mathcal{N}(\mu_\omega, \sigma_\omega).$$

This implementation is similar to that in[21] and the distribution of the random variable $a_\omega$ determined by this process is denoted as $\pi_\omega(a_\omega|\mathrm{s};\phi_\omega)$.

*Waveform modulation*
Considering the waveform should have an oscillatory nature, the waveform of the *i*th joint is defined as:

$$\mathcal{W}^{(i)}(s_p^{(i)}) = \phi_r^{(i,1)} + \sum_{j=1}^{K}\left(\phi_r^{(i,2j)}\sin\left(2\pi j s_p^{(i)} + \delta^{(i)}\right) + \phi_r^{(i,2j+1)}\cos\left(2\pi j s_p^{(i)} + \delta^{(i)}\right)\right) \tag{11}$$

where $\phi_r^{(\cdot)}$ is the parameter adjusted by RL. $\delta^{(i)}$ is a constant set to emit a gait in which the two diagonal legs are swung out at the same timing; it was set to 0 for the left foreleg and right hind leg, and $\pi$ for the right foreleg and left hind leg.

The waveform modulation policy is given by

$$\pi_r(a_r|\mathrm{s}_p;\phi_r) := \mathcal{N}\left(\mu_r(\mathrm{s}_p), \sigma_r(\mathrm{s}_p)\right), \tag{12}$$

where $\mu_r$ and $\sigma_r$ are the mean and variance, respectively. The *i*-th element of mean and variance are calculated as

$$\mu_r^{(i)}(\mathrm{s}_p) := \mathcal{W}^{(i)}(s_p^{(i)}) \tag{13}$$

and

$$\sigma_r^{(i)}(\mathrm{s}_p) := e^{\phi_r^{(i,0)}}, \tag{14}$$

respectively.

## Joint policy

The joint policy is defined as the product of the above three policies:

$$\pi(\mathrm{a}|\mathrm{s}) := \pi_r(\mathrm{a}_r|\mathrm{s}_p;\phi_r)\pi_p(\mathrm{a}_p|\mathrm{s};\phi_p)\pi_\omega(a_\omega|\mathrm{s};\phi_\omega). \tag{15}$$

HRL-CPG optimizes the joint policy for the coupled dynamical system, according to a single objective function, Eq. (6). This optimization problem can be seen approximately as an MDP.

Each policy has different temporal and structural impacts on different scales. For example, base frequency modulation changes the overall speed of the gait, while waveform modulation changes the shape of the entire cycle. The proposed method conducts simultaneous optimization of multiple policies at different scales.

## SAC algorithm for HRL-CPG

Here, we describe an SAC-based optimization for tuning the parameters in our HRL-CPG. There are two critic neural networks, both with the same architecture but different parameters: one network, $Q_\theta(\mathrm{s},\mathrm{a})$, with parameter $\theta$, is used for approximating the soft Q-function, while the other network uses an exponential moving average of $\theta$ (denoted as $\bar{\theta}$) to approximate the soft V-function. The network $Q_\theta$ is trained to minimize the following loss functions:

$$J_Q(\theta) := \mathop{\mathbb{E}}_{(\mathrm{s},\mathrm{a},\mathrm{s}')\sim\mathcal{D}} \left[ \left( Q_\theta(\mathrm{s},\mathrm{a}) - \left( r\left(\mathrm{s},\mathrm{a},\mathrm{s}'\right) + \gamma \hat{V}\left(\mathrm{s}'\right)\right)\right)^2 \right]. \tag{16}$$

The soft V-function here is computed as

$$\hat{V}(\mathrm{s}) := \mathop{\mathbb{E}}_{\mathrm{a}\sim\pi_\phi(\cdot|\mathrm{s})} \left[ Q_{\bar{\theta}}(\mathrm{s},\mathrm{a}) + H(\mathrm{s},\mathrm{a}) \right], \tag{17}$$

where

$$H(\mathrm{s},\mathrm{a}) := -\alpha_p \log \pi_p\left(\mathrm{a}_p \mid \mathrm{s}_p,\mathrm{s}_r\right) - \alpha_\omega \log \pi_\omega\left(a_\omega \mid \mathrm{s}_p,\mathrm{s}_r\right) - \alpha_r \log \pi_r\left(\mathrm{a}_r \mid \mathrm{s}_p\right). \tag{18}$$

To follow the description in "Joint policy", the actor is a set of three policy networks, whose parameters are given by $\phi := (\phi_r, \phi_p, \phi_\omega)$. They are trained to decrease the loss function

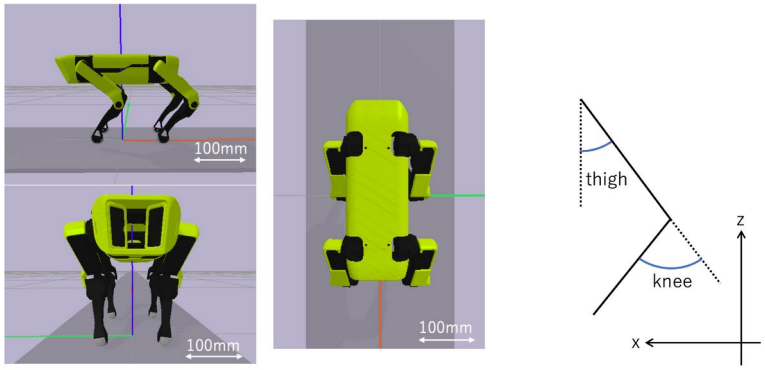
$$J_\pi(\phi) := - \mathop{\mathbb{E}}_{\mathrm{s}\sim\mathcal{D},\mathrm{a}\sim\pi_\phi(\cdot|\mathrm{s})} \left[ Q_\theta(\mathrm{s},\mathrm{a}) + H(\mathrm{s},\mathrm{a}) \right]. \tag{19}$$

In our implementation, we set different pre-defined parameter values for $\alpha_p$ and $\alpha_r$. Their settings are described in "Main results".

The entire algorithm for optimizing the joint policy is shown in Algorithm 1. Note that $\mathrm{s}_t$ stands for the state of the coupled dynamical system, Eq. (8), at the environment step $t$. As a detailed implementation, we followed that of the original SAC algorithm[21], such as minibatches and the double Q-network method[56]. This algorithm is a type of coordinate optimization similar to the expectation-maximization (EM) algorithm[57], well-established in the field of machine learning. That is, we perform stochastic gradient descent steps for the critic (lowering Eq. (16)) and the actor (lowering Eq. (19)) in a step-by-step manner and iteratively.

**Input:** $\theta, \phi, \lambda_Q, \lambda_{\pi_p}, \lambda_{\pi_{\mu_r}}, \lambda_{\pi_{\sigma_r}}, \omega_0, \omega_1$
  $\bar{\theta} \leftarrow \theta$: Initialize target network weights
  $\mathscr{D} \leftarrow \emptyset$: Initialize an empty buffer
  **for** each iteration **do**
    **for** each environment step **do**
      $\mathbf{a}_t \sim \pi_\phi(\cdot|\mathbf{s}_t)$: Sample action from the joint policy
      $\mathbf{s}_{t+1} \sim p(\cdot|\mathbf{s}_t, \mathbf{a}_t)$: Step forward the coupled dynamical system (8)
      $\mathscr{D} \leftarrow \mathscr{D} \cup (\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})$: Store the samples into the buffer
    **end for**
    **for** each gradient step **do**
      $\theta \leftarrow \theta - \lambda_Q \nabla_\theta J_Q(\theta)$
      $\phi_p \leftarrow \phi_p - \lambda_{\pi_p} \nabla_{\phi_p} J_\pi(\phi)$
      $\phi_\omega \leftarrow \phi_\omega - \lambda_{\pi_p} \nabla_{\phi_\omega} J_\pi(\phi)$
      $\phi_{\mu_r} \leftarrow \phi_{\mu_r} - \lambda_{\pi_{\mu_r}} \nabla_{\phi_{\mu_r}} J_\pi(\phi)$
      $\phi_{\sigma_r} \leftarrow \phi_{\sigma_r} - \lambda_{\pi_{\sigma_r}} \nabla_{\phi_{\sigma_r}} J_\pi(\phi)$
      $\bar{\theta} \leftarrow \tau\theta + (1-\tau)\bar{\theta}$: Update target network weights
    **end for**
  **end for**
**Output:** $\phi$

**Algorithm 1**. HRL-CPG Algorithm




**(a)** Side and top views **(b)** Initial pose

**Fig. 5**. Quadruped robot REX (**a**) side and top views of the quadruped robot with its scales. (**b**) The initial pose which is depicted in terms of the left foreleg

| Variable | Environment 1 | Environment 2 | Environment 3 | Environment 4 | Environment 5 |
|---|---|---|---|---|---|
| Gradient (degree) | 0.0 | -3.0 | 3.0 | 0.0 | 0.0 |
| Friction coefficient | 1.0 | 1.0 | 1.0 | 0.5 | 2.0 |

**Table 1**. Terrain configurations

## Experiments and results

In this section, we examine the applicability of our HRL-CPG framework in generating locomotion of a quadruped robot in a variety of environments.

We conducted experiments with consistent hyperparameters for the five environments, and the proposed method exhibited high performance in four out of the five environments, which suggested the stability of our RL method.

### Task description

We used Bullet Physics SDK for physics simulation and Rex model as a simulator of the quadruped robot, both of which are open-source[49,58].

*Quadruped robot*

Figure 5b shows the quadruped robot we used in the experiments. The length (head to tail) and the width (shoulder to shoulder) of the robot are about 0.35 m and 0.20 m, respectively. The height of the robot with an initial pose shown in the figure is about 0.24 m. Let *x*-, *y*-, and *z*-axes denote the proceeding direction, the left-hand-side, and the up-side of the robot simulator, respectively; they are Cartesian and fixed irrespective to the robot's posture. Each leg consists of two joints, a thigh and knee, arranged to a sagittal plane that is parallel to the *xz*-plane.

The robot's state is represented as a vector of 18 dimensions ($\mathrm{s}_r \in \mathcal{S} \subset \mathcal{R}^{18}$): angles and angular velocities of thighs and knees (16 variables), the rotation angle of the robot's torso around the *z*-axis, and the displacement of the torso along the *y*-axis.

Eight joints (the thigh and knee of each leg: $N = 8$) in total are controlled by our method. Our policy emitted the target joint angles every 20 ms. (i.e., 50 Hz control frequency). The low-level PID controller regulates each joint angle to the target angle specified by the policy, with a control frequency of 1,000 Hz.

*Environments*

The task for the robot in this experiment is to move forward without deviating from a straight aisle with a width of 0.4 meters. The robot needs to acquire a policy to navigate a passageway which is as narrow as approximately twice its total width.

To demonstrate that learning is possible in a variety of environments, we prepared five kinds of terrain with different gradients and friction coefficients. The configurations of each environment are listed in Table 1. When changing the friction coefficient, we altered the value of the lateral friction.

*Reward function*

Regardless of the environment, the same reward function:

$$r := k_x \Delta_x - \delta_t (k_y y^2 - k_\varepsilon |\varepsilon|) + p \tag{20}$$

was used. Here, $\Delta_x$ is the distance traveled from the previous control cycle along with the $x$-axis, and corresponds to the average speed of each control step. The faster the robot moves forward, the greater the reward. $y$ is the displacement from the center line along the $y$-axis and the corresponding reward is greater when staying at the center of the aisle. $\varepsilon$ is the sum of the products of torque and rotational speed at each joint. $k_x$, $k_y$ and $k_\varepsilon$ are the constants that determine the contributions of the three factors, and were empirically set to 50.0, 1.0, and 0.0025, respectively. $\delta_t$ (=0.02) is the constant used to take a balance between the instantaneously observed reward and the cumulative rewards per second.

$p$ is a negative reward (penalty, -3) and is applied when the robot tips over or gets off the aisle; when the robot received this, the control episode was terminated. The maximum time length of each episode was set to 20 s (1,000 control steps since the control cycle is 20 ms); even when the robot performed walking for 20 s, the episode was terminated. It was considered as tipping over when the robot's torso tilted more than $\cos^{-1}(0.85)$ rad. Since both sides of the aisle are cliffs, getting off the aisle will result in a fall. The reward function, Eq. (20), encouraged the robot to walk along the aisle faster and stabler without falling for 20 s.

*Initial pose*

Figure 5b (b) depicts the initial pose, which was set empirically; the joint angles were set to 0.904 rad for the thigh parts and 1.48 rad for the knee parts. That is, $s_{r0}^{(i)} \in \{0.904, 1.48\}$ is the initial state of the robot's $i$-th joint angle. The target angles are determined based on the policy's output and those values as $0.904 + a_r^{(i)}$ for the thighs and $1.48 + a_r^{(i)}$ for the knees.

*Initial policy parameters*

$\phi_p$ and $\phi_\omega$, the parameters of neural networks for the phase control policy $\pi_p$ and the base frequency modulation policy $\pi_\omega$, respectively, were initialized with Kaiming initialization[59].

The policy parameters of the waveform policy $\pi_r(a_r|s_p)$ were set as:

$$\begin{aligned}
\phi_r^{(i,0)} &= 0.01 \quad \forall i \\
\phi_r^{(i,1)} &= 0 \quad \forall i \\
\phi_r^{(i,3)} &= -0.1 \quad \text{if the } i\text{-th oscillator controls a thigh} \\
\phi_r^{(i,3)} &= 0.2 \quad \text{if the } i\text{-th oscillator controls a knee} \\
\phi_r^{(i,j)} &= 0 \quad \text{if } j \text{ is not 1 or 3}
\end{aligned}$$

*Training and test performance*

This study aims to assess how quickly the proposed method enables the robot to adapt to individual environments. To ensure a consistent evaluation of this adaptation process, both training and testing were performed in the same environment.

On the other hand, to evaluate whether stable policy can be learned, the behavior is designed to exhibit consistent control during testing, while stochastic behavior is encouraged during training to facilitate exploration. During training, actions are computed through a sampling procedure represented by the joint policy to encourage exploration. The randomness involved in this process is controlled by the parameters $\sigma_p$ of the phase control policy (Eq. 9), $\sigma_\omega$ of the base frequency modulation policy, and $\sigma_r$ of the waveform modulation policy (Eq. 14), which are dependent on the state at each steps. During testing, these values are set to zero, and actions are applied to the environment (and the CPGs) using the same procedure.

## Main results

We used consistent hyperparameters. The base modulation frequency was $\omega_0 = 6\pi$ and $\omega_1 = 2\pi$. The temperature parameters, $\alpha_r$, $\alpha_p$ and $\alpha_\omega$, were determined at 0.00001, 0.1 and 0.1, respectively, such to allow the robot to generate good walks in environment 1 (flat surface). Although the relatively large $\alpha_p$ seems to introduce a significant fluctuation to the PON via the phase modulation policy $\pi_p$, this is not the case; even if we apply large noises to the PON units independently, effective change in the oscillator's state remains small due to compensation of independent perturbations within the whole PON dynamics. Making $\alpha_p$ large thus enables RL on the phase modulation policy to perform effective exploration under such noise compensation. To examine the robustness of the parameters across different environments, we adopted the temperature parameters determined in environment 1 regardless of the target environment.

Figure 6(a) shows the learning curves, i.e., the test performance averaged over 10 training runs. In each test, the robot walks through the same environment where the controller was trained. The vertical axis denotes the accumulated reward for a roll-out episode, and the horizontal axis denotes the learning step. Differences in line color indicate differences in the five environments (Table 1).

A single-episode evaluation roll-out was performed every 5,000 learning steps. In each panel, the solid line and shaded area are the median and 90 and 10 percentiles, respectively. Figure 6(a) indicates that in environment 4, there are wide variation in the performance. Further analysis on the progression of learning, detailed in Appendix C, evaluated the failure rate (the frequency of falls during test episodes before the termination step) and progression distance across successful episodes. In environment 4, the failure rate did not decrease significantly, unlike in other environments, where the failure rate steadily declined and progression distance increased.

Finally, we report the number of samples that were needed to achieve suitable gait patterns. On average, in all the environments (except for environment 4), walking was achieved in approximately $3.0 \times 10^4$ learning steps, which corresponded to approximately $1.0 \times 10^4$ physical seconds and $1.5 \times 10^3$ trials (each trial was up

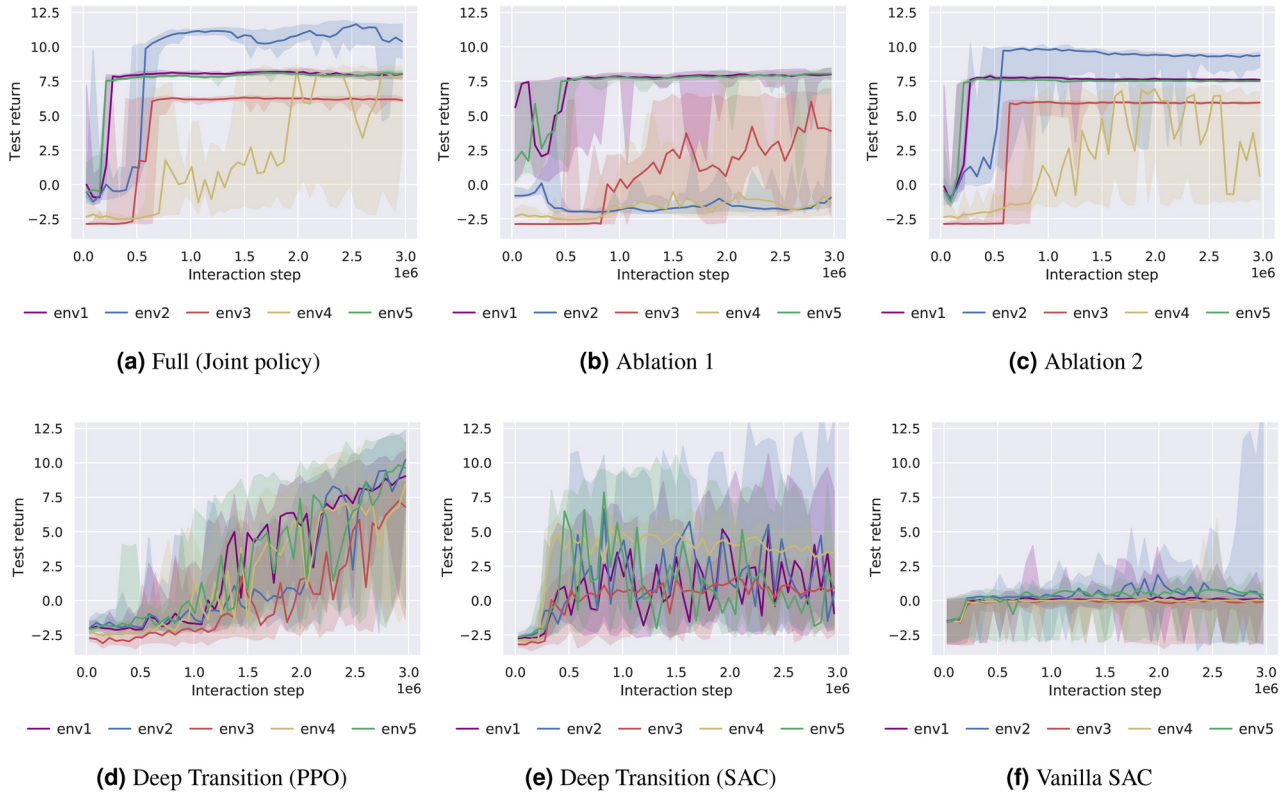



**Fig. 6**. Learning curves; the horizontal and vertical axes denote the number of interaction steps with the environment and the test performance (the total return of one roll-out), respectively. The test was performed every 5,000 interaction steps. "Vanilla SAC" stands for a naive SAC where no CPG was used

| | Full | Ablation 1 | Ablation 2 | DT (PPO) | DT (SAC) | Vanilla SAC |
|---|---|---|---|---|---|---|
| $\pi_p$ | Yes | Yes | Yes | Yes | Yes | N/A |
| $\pi_\omega$ | Yes | No | Yes | No | No | N/A |
| $\pi_r$ | Yes | Yes | No | Yes$^\dagger$ | Yes$^\dagger$ | N/A |

**Table 2**. Characteristics of comparative methods. "DT" stands for Deep Transition and "Vanilla SAC" stands for a naive SAC where no CPG was used

to 20 s), implying that our method is applicable to autonomous adaptation by a realistic quadruped robot with a reasonable number of learning steps and/or physical time.

## Comparisons

We compared our proposed method with two ablation models and three baseline methods. The differences are summarized in Table 2. In Ablation 1, the base frequency modulation policy $\pi_\omega$ is excluded from the joint policy (Full), meaning that $\omega$ in Eq. (10) is always $\omega_0$. In Ablation 2, the waveform modulation policy $\pi_r$ is excluded from the learning targets, meaning that $\pi_r$ is fixed at its initial setting described in "Task description". The hyperparameters were set to the identical values as those in the Full.

As one of the state-of-the-art methods that are mostly related to our method, Deep Transition was adopted[46,60], since this method is designed for RL of a CPG controller. According to this method, the sensory feedback from the robot to the CPG controller and the amplitude of the motion (gait pattern) are adjusted. The former and the latter correspond to $\pi_p$ and $\pi_r$ in our method, respectively. However, the tunable characteristics of the waveform modulation in our method are wider, i.e., not only the amplitude but also the shape can be tuned. The original Deep Transition used the proximal policy optimization (PPO) for RL, which might be slower than SAC, then we also tested its variation with the SAC-based RL to make the comparison fair. Additionally, we also tested a naive SAC where no CPG was used. In this case, a neural network where the input and output were the sensory feedback from the robot and the target joint angles, respectively, was trained by SAC. These ablation models and baseline methods were evaluated following the same training-testing scheme as that for the proposed method. The detail of experiments for the Deep Transition is described in Appendix E.

In the case of Ablation 1 (Fig. 6b), there has been an overall decline in performance as well as an increase in the number of cases that do not adapt well. This suggests $\pi_\omega$ (base frequency modulation) is important for the

adaptation. Actually, the effect of this policy can be realized by adding a constant to the phase modulation policy $\pi_p$, since the both outputs are just added together in Eq. (4). In other words, they are redundant parameters. However, the result would suggest that having an explicit parameter that controls the average period of gait across all four legs is important for adaptation.

In the case of Ablation 2 (Fig. 6c), it can be seen that gait patterns suited to each environment were obtained as well as in the Full case. The returns of successful roll-outs were lower than those by the proposed one, especially in environment 2. This suggests that waveform modulation $\pi_r$ would have committed to obtain better gait patterns thanks to the high degree of freedom.

Deep Transition (PPO) (Fig. 6d) adapts steadily in all environments, where Deep Transition (SAC) (Fig. 6d) exhibited better learning. However, both methods failed to achieve stable gaits. The returns of successful roll-outs, especially by Deep Transition (SAC), are high, suggesting good policies could be obtained if sample efficiency is not considered. The improvement in average return has been stable, especially when PPO is used, and this may be useful as a preliminary training in situations like sim2real.

On the contrary to the other methods, the vanilla SAC (without CPG) did not allow the robot to obtain any good gait patterns. This suggests that the use of CPG is crucial for the gait adaptation to the environment.

We present success survival plots in Fig. 7. On the horizontal axis, the reward obtained in each environment is normalized by the best test-trial reward in that environment (excluding outliers), yielding a performance ratio. The vertical axis shows the proportion of all test episodes-collected from 10 random seeds across 5 environments, conducted between 0 and 3,000,000 interaction steps-that exceed the corresponding performance ratio. The area under the curve (AUC), displayed to the right of each method name in the legend, indicates the average proportion of test episodes that achieve at least each performance ratio in the survival curve. The theoretical maximum of the AUC is 1; hence, larger AUC values reflect higher average rewards on average. From Fig. 7, it is evident that the Full achieved the largest AUC, implying that it learns most efficiently on average across all five environments. These results suggest that the proposed method, Full, is capable of maintaining consistently high performance under a variety of environmental conditions.

## Further investigation

In order to investigate the influence of each component in our proposed method in greater detail, we examined how learning performance differs under conditions involving perturbations. This setup serves as a proxy for adaptation to diverse environments. In this study, we focused on perturbations in the slope of the aisle and in the sensor feedback signals. Specifically, at the start of each episode, a slope value was randomly selected from the interval $[-0.5, 0.5]$. For the sensor feedback (observation) noise, independent Gaussian noise $\epsilon_i \sim \mathcal{N}(0, s)$ was added to each observation dimension at every interaction step, where $s$ is a parameter representing the noise intensity and was fixed at 0.005 during training.

Based on Environment 1 (a flat aisle with slope 0) described in "Comparisons", we created three additional variants to examine the combined effects of slope perturbations and observation noise. Specifically, the variants were: (a) slope perturbation present with no observation noise, (b) no slope perturbation with observation noise present, and (c) both slope perturbation and observation noise present. Figure 8 shows the learning curves obtained from the same training procedure applied to these three additional environments. In terms of the median performance, Full exhibited the highest learning curve, whereas Ablation 1 showed performance deterioration or large variance in the middle of training, and Ablation 2 converged to a conservative level.

Next, Fig. 9 presents progression distance for successful episodes (i.e., those that end without falling). Full again showed the highest performance, but Ablation 1 ultimately achieved the second-longest distance. In

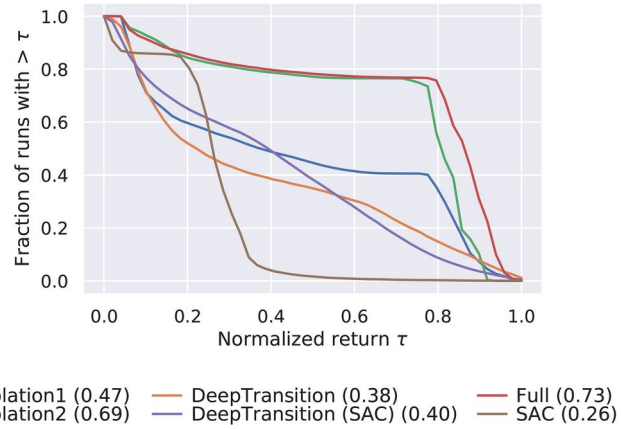
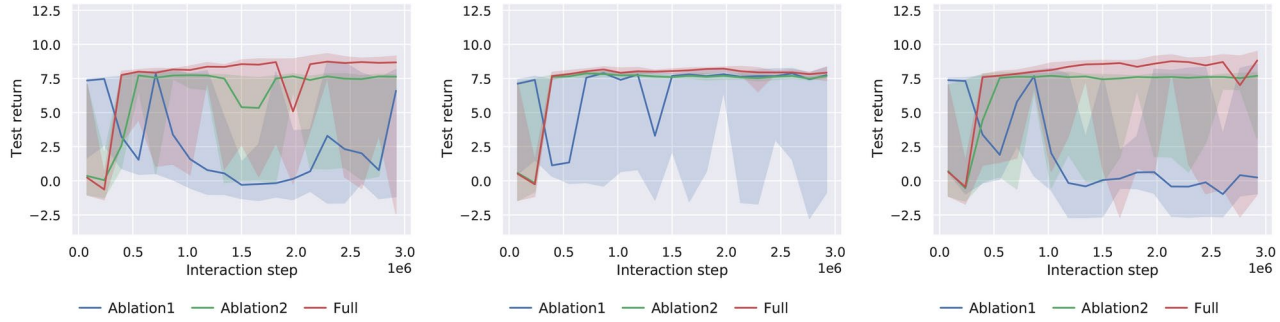



**Fig. 7**. Success survival plots assessed across the five environments. The horizontal axis represents the performance ratio compared to the best test trial, excluding outliers, in each environment. The vertical axis indicates the probability of obtaining a policy through learning that surpasses this performance ratio. The number in the blacket of each method name in the legend is the area under curve (AUC), and indicates the average performance of the test episodes

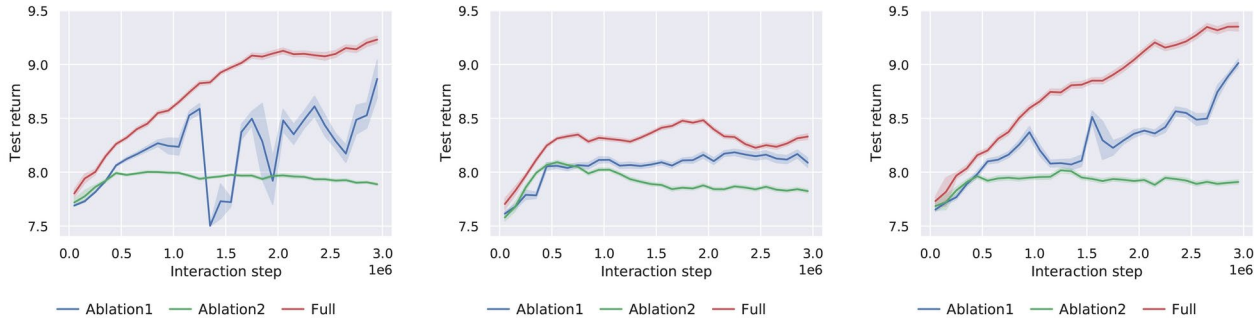



**(a)** Slope perturbation present with no observation noise

**(b)** No slope perturbation with observation noise present

**(c)** Both slope perturbation and observation noise present

**Fig. 8**. Learning curves in noisy environments, following the same training and plotting methodology as Fig. 6, except for the introduction of environmental perturbations

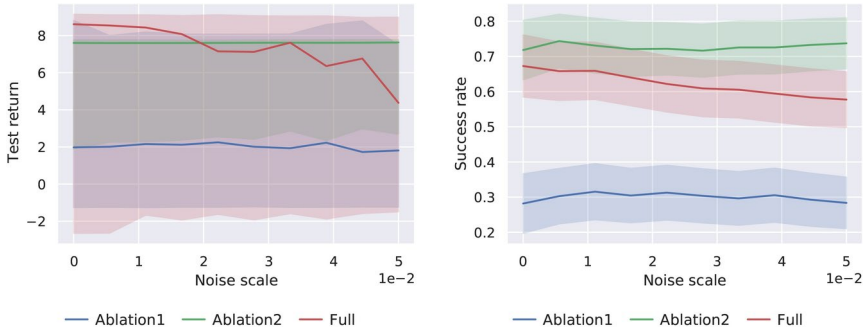



**(a)** Slope perturbation present with no observation noise

**(b)** No slope perturbation with observation noise present

**(c)** Both slope perturbation and observation noise present

**Fig. 9**. Progression distance across successful episodes. The solid and shaded area are the mean and 95% confidence interval, respectively




**(a)** Return.

**(b)** Success rate.

**Fig. 10**. Test performance at different observation noise scales. (**a**) Return, where the solid line and shaded area are the median and is 10% and 90% percentiles, respectively. (**b**) Success rate, where the solid line and shaded area are the mean and 95% confidence interval, respectively

contrast, Ablation 2 plateaued around 8 m after approximately 500,000 interaction steps in all environments, displaying no further improvement.

To analyze the policies learned, we compared control performance under different levels of observation noise. During the above experiments, policy parameters were saved every 50,000 interaction steps after 2,500,000 steps, resulting in different models of policy. Each model was then evaluated in environments where the noise scale $s$ ranges from 0 to 0.05, and we recorded episode return and success/failure outcomes. Figure 10 plots the episode

return and success rates, with the horizontal axis representing the noise scale and the vertical axis indicating the respective metrics. The ranking of algorithms at $s = 0.005$ matched the order observed late in Fig. 8. However, as the noise intensity increases, the episode reward for the Full algorithm decreases, whereas it remains largely unchanged for the two Ablation. This trend in episode return correlated with the success rate.

Summarizing the results from Figs. 9 and 10, Ablation 2-lacking a mechanism to adjust the base waveform-settled into a conservative gait pattern with fewer interaction steps, whereas both Full and Ablation 1 learned faster gaits. Because faster gaits are inherently more unstable, Ablation 1, which does not explicitly modulate the base frequency, failed to stabilize them. By contrast, the Full algorithm appears capable of learning a faster gait with relatively high stability.

Finally, Fig. 11 shows the base waveform adjusted by Full algorithm. Notably, it differs from the hand-designed sinusoidal waveform.

## Concluding remarks

In this study, we developed a data-driven DRL method for optimizing the CPG-involving policies and evaluated its performance in a practical manner with the expectation of its applicability in real robot controls. To this end, we proposed a hierarchical RL framework, called HRL-CPG, which allows a coupled system of the target robot simulator and a CPG network to autonomously adapt to various kinds of terrains to walk through.

In the HRL-CPG, we prepared a hierarchically arranged control policy consisting of a waveform modulation policy, a phase modulation policy, and a base frequency modulation policy, and presented an SAC-based optimization method to tune them. When the HRL-CPG was applied to acquisition of the adaptive gait by a four-legged robot simulator in five kinds of aisles with different gradients and different friction coefficients, we observed that stable gait motions were acquired within a reasonably small number of trials and errors.

In environments incorporating perturbations such as aisle slope and sensory feedback noise, our findings demonstrate that a relatively fast and stable gait can be learned. Specifically, when comparing performance across additional environments involving various combinations of slope and noise, Ablation 1 tended to exhibit performance degradation during training, and Ablation 2 remained confined to a conservative gait. In contrast, Full efficiently acquired a faster gait with relatively high stability.

These number of trials-and-errors to each environments (except for Environment 4) corresponded to 10,000 physical seconds ($\approx$ 2.7 h). Thus, we conclude that our HRL-CPG can be a candidate for a DRL method that allows real or realistic robots to adapt to a variety of environments within a moderate physical time.

In the proposed method, we suggested a learning approach for the CPG controller under the condition of uniform terrains. However, since uniform terrain cannot be necessarily assumed in real-world applications, extensions would be demanded. For example, there can be various types of non-uniform ground surfaces, such as local unevenness (bumps and potholes) or stairs. In such an environment, incorporating sensor information from a vision-equipped robot into the policy must be crucial and remains a challenge for future work. As a preliminary experiment, we applied our HRL-CPG method to train a controller for the robot's walking task on surfaces with zebra-patterned bands of varying friction. We provided the robot with the distance between its

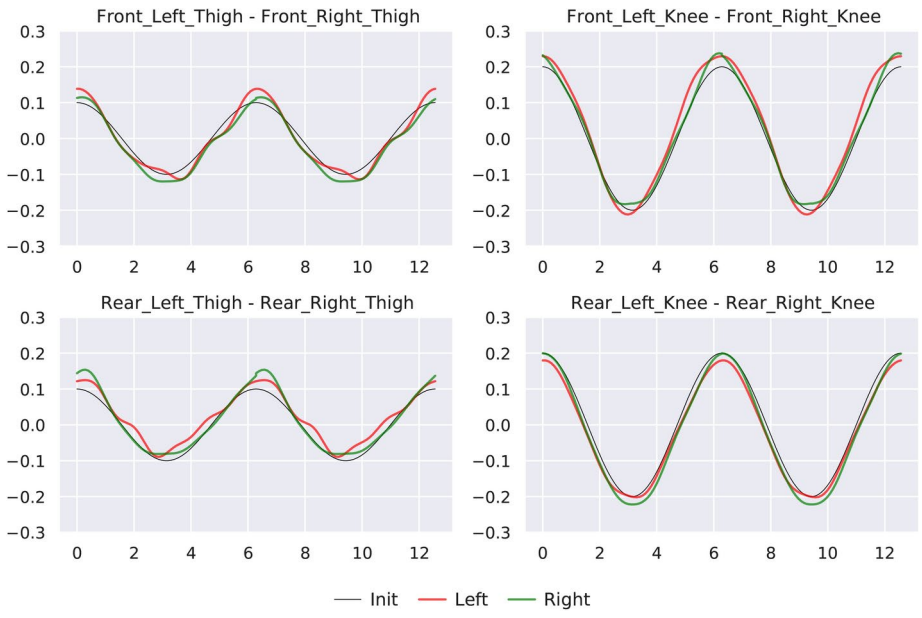



**Fig. 11**. Learned periodic waveforms after 3,000,000 interaction steps. Each of the four panels plots two cycles of the target joint angles in phase coordinates $[rad]$ for the four pairs of leg joints (thigh and knee on both the front and rear legs). The left-side angles (in red) and right-side angles (in green) are compared against the initial (black) waveform. Note that the initial waveforms for the left and right sides were identical at the start, hence the black lines overlap

swung foot and the nearest boundary of a higher-friction zebra band in the forward direction. The robot learned a gait that placed its feet on the boundary of zebra bands, maintaining consistent friction for forward movement (Fig. 13 in Appendix B). This suggests that our method could be applicable to such tasks if appropriate visual features are provided.

The experiments suggest that our method achieves higher sample efficiency in adaptation compared to the Deep Transition, one of the state-of-the-art methods. Unlike our proposed method, which learns variations of a single gait, Deep Transition explores various gait patterns, resulting in different adaptive characters. The need for extensive exploration may impact the learning speed. In our proposed method, expanding adaptation to diverse situations using various gaits is a future challenge. Developing a higher-level mechanism to switch between gaits, such as trot, pace, and bounce, based on the situation, would be essential.

## Data availibility

No datasets were generated during the current study. The source codes and simulation results produced in the current study are available from the corresponding author on reasonable request.

## A Soft actor-critic algorithm

For a particular policy $\pi$, we define the following soft Bellman equation:

$$Q^{\pi}(\mathrm{s},\mathrm{a}) = r(\mathrm{s},\mathrm{a}) + \gamma \mathbb{E}_{\mathrm{s}'\sim p(\cdot|\mathrm{s},\mathrm{a})}\left[V^{\pi}\left(\mathrm{s}'\right)\right], \tag{21}$$

$$V^{\pi}(\mathrm{s}) = \mathbb{E}_{\mathrm{a}\sim\pi(\cdot|\mathrm{s})}\left[Q^{\pi}(\mathrm{s},\mathrm{a}) - \alpha \log \pi(\cdot \mid \mathrm{s})\right], \tag{22}$$

where $Q^{\pi}$ and $V^{\pi}$ are the soft Q-function and soft V-function, respectively, and $\alpha$ is the same as in Eq. (6). By iteratively applying the soft Bellman equation above, we observe that the soft V-function marginalized over the initial state distribution $p_0$ becomes the objective of the entropy-based regularized RL in Eq. (6):

$$J(\pi) := \mathbb{E}_{\mathrm{s}\sim p_0(\cdot),\mathrm{a}\sim\pi(\cdot|\mathrm{s})}\left[Q^{\pi}(\mathrm{s},\mathrm{a}) - \alpha \log \pi(\mathrm{a} \mid \mathrm{s})\right]. \tag{23}$$

To solve the simultaneous equations, Eqs. (21) and (22), corresponds to the policy evaluation. Hereafter, for the sake of simplicity, we will assume that $\alpha = 1$.

A variational optimization of the objective function, Eq. (23), with respect to policy $\pi$ becomes the soft-max of the soft Q-function $Q^{\pi}$. Subsequently, we introduce a policy improvement method such that the current policy approaches the soft-max policy in terms of the Kullback-Leibler (KL) divergence:

$$\pi' := \arg\min_{\bar{\pi}\in\Pi} D_{KL}\left(\bar{\pi}(\cdot|\mathrm{s}) \| \frac{\exp Q^{\pi}(\mathrm{s},\cdot)}{Z(\mathrm{s})}\right), \tag{24}$$

where $\Pi$ is a desired set of policies, e.g., a parameterized family of distributions such as Gaussian distributions, and $Z(\mathrm{s})$ is a normalization term that validates the soft-max policy as a probability distribution. The use of the KL divergence allows this policy improvement to proceed in a moderate manner, and enables a policy implementation with a neural network.

The alternative iterations of the above policy evaluation and policy improvement referred to as soft policy iteration, would lead to an optimal policy with respect to the objective function, Eq. (6).

Because the system dynamics $p(\mathrm{s}'|\mathrm{s},\mathrm{a})$ is unknown in a model-free RL setting, we need to solve the simultaneous equations, Eqs. (21) and (22), without the system dynamics. According to the SAC algorithm[21], we use a least square temporal difference (LSTD) method[61] to obtain an approximated soft Q-function:

$$J_Q(\theta) := \mathop{\mathbb{E}}_{(\mathrm{s},\mathrm{a},\mathrm{s}')\sim\mathcal{D}}\left[\left(Q_\theta(\mathrm{s},\mathrm{a}) - \left(r(\mathrm{s},\mathrm{a}) + \gamma\hat{V}\left(\mathrm{s}'\right)\right)\right)^2\right], \tag{25}$$

where $\mathcal{D}$ is the dataset of previously sampled tuples of the state, action and next state, and $Q_\theta$ is a parameterized function of *s* and *a* with parameter $\theta$ to approximate the soft Q-function. Similar to the one presented in[21], we define an approximated soft V-function $\hat{V}$ as

$$\hat{V}(\mathrm{s}) := \mathbb{E}_{\mathrm{a}\sim\pi_\phi(\cdot|\mathrm{s})}\left[Q_{\bar{\theta}}(\mathrm{s},\mathrm{a}) - \log \pi_\phi(\mathrm{a} \mid \mathrm{s})\right], \tag{26}$$

where $\bar{\theta}$ is the parameter of another function approximator, a target soft Q-function, which is smoothed with an exponentially forgetting weight. That is, we do not define a function approximator for the V-function but approximate it with Monte-Carlo samples using $\pi_\phi$ and $Q_{\bar{\theta}}$. This *critic* learning is performed using one-step backups, i.e., the current state s, the current action a, the associated reward *r*, and the next state $\mathrm{s}'$; the actual algorithm can be done online or offline with minibatches.

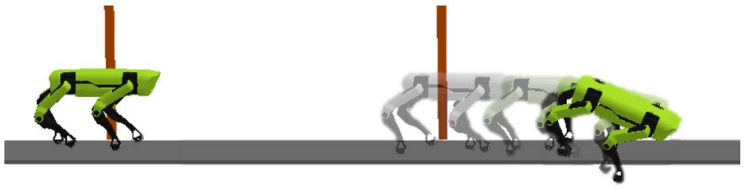
The policy improvement, i.e., the *actor* learning, in the SAC algorithm is performed as

$$\arg\min_{\phi} D_{KL}\left(\pi_\phi(\cdot|\mathrm{s}) \| \frac{\exp Q^{\pi}(\mathrm{s},\cdot)}{Z(\mathrm{s})}\right) = \arg\min_{\phi} \mathbb{E}_{\mathrm{a}\sim\pi_\phi(\cdot|\mathrm{s})}\left[\log \pi_\phi(\mathrm{a}|\mathrm{s}) - Q^{\pi}(\mathrm{s},\mathrm{a})\right], \tag{27}$$

where $\phi$ is the policy parameter. This optimization is conducted in a stochastic gradient decent manner by considering the gradient of the right-hand side of the equation above with respect to the policy parameter.



**(a)** Example of successful case



**(b)** Example of failure case due to fall



**(c)** Front view of a failure case due to fall.

**Fig. 12**. Storoboscopic photographs: two side-view images of successful and failed cases (**a, b**), and a front-view image of a failure case (**c**). The vertical red bars are displayed at 1-meter intervals



**Fig. 13**. A screen shot of a preliminary experiment on surfaces with zebra-patterned bands. The white bands have a lower friction coefficient than the black bands

## B Storoboscopic photography while walking

Figure 12 presents stroboscopic photographs: two side-view images of successful and failed cases, and a front-view image of a failure case.
A "failure" is defined as a situation where the agent loses its posture due to events such as falls, as shown in Fig. 12b, before the termination of an episode.
"Success" refers to cases where no such failure occurs. The detailed definition is provided at the end of the description of the reward function in "Task description".
Additionally, Fig. 13 is a screenshot of a preliminary experiment described in "Concluding remarks" (the end of the 4-th paragraph), where walking occurs on a surface with alternating friction levels at regular intervals, resembling a crosswalk pattern.

## C Evaluation of learning performance: failure rate and progression distance

We investigated the components of reward maximization performance. Figure 14 shows the learning curves for the failure rate and progression distance during successful episodes. The failure rate is plotted on the left axis, while the progression distance is shown on the right axis. The failure rate was calculated based on 500 test episodes conducted every 500,000 steps (evaluations were performed every 5,000 steps with five random seeds). The progression distance was defined as the mean x-coordinate at the 1,000th step of successful test episodes. Since the episode length was fixed, the progression distance is proportional to the average forward speed along the x-axis.
In the early stages of reinforcement learning (RL), the robot rarely completed episodes in any environment. As training progressed, the failure rate decreased, while the progression distance monotonically increased

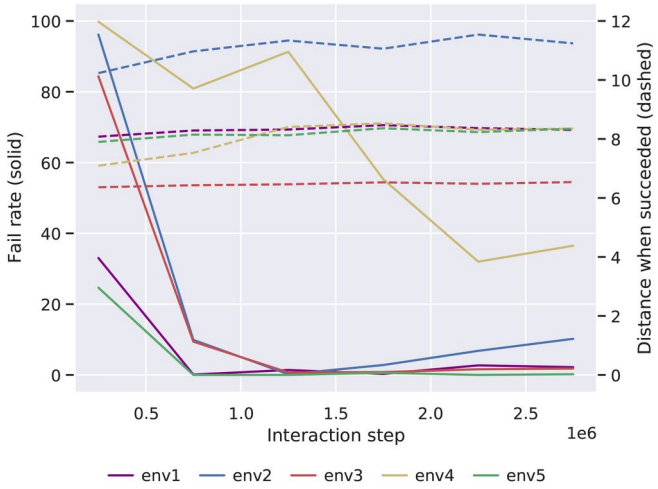



**Fig. 14**. Failure rate and the progression distance when succeeded

| Env | Fail rate | Distance [m] |
|---|---|---|
| 1 | 2.20 % | 8.31 |
| 2 | 10.20 % | 11.24 |
| 3 | 1.80 % | 6.54 |
| 4 | 36.50 % | 8.35 |
| 5 | 0.20 % | 8.35 |

**Table 3**. Fail rate and locomotion distance when succeeded at final interaction steps

| Parameter | Value |
|---|---|
| Discount ($\gamma$) | 0.99 |
| Replay buffer size | $10^6$ |
| Environment step at which HRL-CPG starts ($T_{\text{start}}$) | $10^4$ |
| Stochastic gradient descent optimizer | Adam[62] |
| Number of samples per minibatch ($B$) | 256 |
| Learning rate for critic networks | $3 \times 10^{-4}$ |
| Learning rate for the phase modulation policy and base frequency modulation policy | $3 \times 10^{-4}$ |
| Learning rate for $\mu_r$ (mean of robot control signal) | $1 \times 10^{-8}$ |
| Learning rate for $\sigma_r$ (variance of robot control signal) | $1 \times 10^{-5}$ |
| The number of hidden units (for all neural networks except the waveform policy ) | (256, 256) |
| Degree of Fourier expansion for robot controller ($K$) | 5 |
| Smoothing coefficient for two target networks ($\tau$) | $5 \times 10^{-3}$ |
| Activation function used for all layers | ReLU |

**Table 4**. Hyperparameters commonly used in experiments in "Experiments and results".

from the early stages. Table 3 summarizes the failure rates and progression distances during the final training interval.
The higher forward progression distance observed in Environment 2 (downhill) compared to the lower progression distance in Environment 3 (uphill) reflects differences in the robot's movement speed, which correspond to the maximum return differences shown in Fig. 6.
These findings indicate that the robot learned to avoid failures while simultaneously acquiring gaits that enabled faster forward progression.

## D Supplementary discussion on hyperparameters

Table 4 presents the hyperparameter settings used in the experiments described in "Experiments and results". The implementation generally followed the original SAC algorithm[21], with certain modifications, such as separate learning rates for the mean and variance.

## E Deep transition in the experiments

As a baseline method In this study, we selected the approach presented by[46], utilizing the open-source implementation of Deep Transition[60]. The original Deep Transition paper employed a quadruped robot model of a similar size to the Rex model, adapting a single policy network model across multiple environments with varying physical parameters. For the comparison in the experiments, we adapted the CPG and PPO implementations from the Deep Transition codebase and applied them to the control task of the Rex model under the same problem setting as our proposed method. Specifically, we retained the original architecture of the policy and critic networks, with the neural network parameters numbering 183,692 and 182,273, respectively. This implementation is referred to as Deep Transition (PPO) in our experiments. When conducting experiments using the default hyperparameters, the learning process was found to be slow. To address this issue, a hyperparameter search was performed on the mini-batch size, learning rate, and target KL divergence of the PPO algorithm.
On the other hand, for Deep Transition (SAC), we utilized the SAC algorithm implementation and network architecture described in "Comparisons". The numbers of the actor and critic model parameters were 80,920 and 156,162, respectively.
In both PPO and SAC implementations, the state inputs to the actor and critic included 18 quantities representing the robot's state, consistent with our proposed method, augmented with 16-dimensional quantities summarizing the CPG state.

Received: 28 November 2024; Accepted: 12 March 2025
Published online: 02 April 2025

## References

1. Raibert, M., Blankespoor, K., Nelson, G. & Playter, R. BigDog, the Rough-Terrain quadruped robot. *IFAC Proceedings Volumes* **41**, 10822–10825. https://doi.org/10.3182/20080706-5-KR-1001.01833 (2008).
2. Hutter, M. et al. ANYmal - A highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 38–44. IROS.2016.7758092 (2016).
3. Hwangbo, J. et al. Learning agile and dynamic motor skills for legged robots. *Sci. Robot.* **4**, eaau5872. https://doi.org/10.1126/scirobotics.aau5872 (2019).
4. Hutter, M. et al. ANYmal - toward legged robots for harsh environments. *Adv. Robot.* **31**, 918–931 (2017).
5. Sutton, R. S. & Barto, A. G. *Reinforcement Learning: An Introduction* (MIT Press, 1998).
6. Wieber, P.-B. Model predictive control for biped walking motion generation. *Journal of the Robotics Society of Japan* **32**, 503–507. https://doi.org/10.7210/jrsj.32.503 (2014).
7. Bledt, G. & Kim, S. Extracting legged locomotion heuristics with regularized predictive control. In *2020 IEEE International Conference on Robotics and Automation*. 406–412. https://doi.org/10.1109/ICRA40945.2020.9197488 (2020).
8. Kalashnikov, D. et al. Scalable deep reinforcement learning for Vision-Based robotic manipulation. In *Proceedings of the 2nd Conference on Robot Learning*. Vol. 87. *Proceedings of Machine Learning Research*. 651–673 (2018).
9. Zhou, S. K., Le, H. N., Luu, K., V Nguyen, H. & Ayache, N. Deep reinforcement learning in medical imaging: A literature review. *Med. Image Anal.* **73**, 102193. https://doi.org/10.1016/j.media.2021.102193 (2021).
10. Jin, J. et al. Real-time bidding with multi-agent reinforcement learning in display advertising. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2193–2201 (2018).
11. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533. https://doi.org/10.1038/nature14236 (2015).
12. Moerland, T. M., Broekens, J., Plaat, A. & Jonker, C. M. Model-based reinforcement learning: A survey. arXiv:2006.16712 (2020).
13. Kaiser, L. et al. Model based reinforcement learning for atari. arXiv:1903.00374 (2019).
14. Ha, D. & Schmidhuber, J. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems* (Bengio, S. et al. eds.). Vol. 31 (Curran Associates, Inc., 2018).
15. Hafner, D., Lillicrap, T., Ba, J. & Norouzi, M. Dream to control: Learning behaviors by latent imagination. arXiv:1912.01603 (2019).
16. Brockman, G. et al. OpenAI gym. arXiv:1606.01540 (2016).
17. Engstrom, L. et al. Implementation matters in deep policy gradients: A case study on PPO and TRPO. arXiv: 2005.12729 (2020).
18. Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J. & Quillen, D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Robot. Res.* **37**, 421–436. https://doi.org/10.1177/0278364917710318 (2018).
19. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. arXiv: 1707.06347 (2017).
20. Schulman, J., Levine, S., Moritz, P., Jordan, M. I. & Abbeel, P. Trust region policy optimization. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*. Vol. 37. 1889–1897 (2015).
21. Haarnoja, T. et al. Soft actor-critic algorithms and applications. arXiv: 1812.05905 (2018).
22. Abdolmaleki, A. et al. Maximum a posteriori policy optimisation. arXiv: 1806.06920 (2018).
23. Peng, X. B., Andrychowicz, M., Zaremba, W. & Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. arXiv:1710.06537 (2017).
24. James, S. et al. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. arXiv:1812.07252 (2018).
25. Tan, J. et al. Sim-to-real: Learning agile locomotion for quadruped robots. arXiv:1804.10332 (2018).
26. Zhao, W., Queralta, J. P. & Westerlund, T. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In *2020 IEEE Symposium Series on Computational Intelligence*. 737–744. https://doi.org/10.1109/SSCI47803.2020.9308468 (2020).
27. Janner, M., Fu, J., Zhang, M. & Levine, S. When to trust your model: Model-based policy optimization. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 12519–12530 (2019).
28. Chua, K., Calandra, R., McAllister, R. & Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 4759–4770 (2018).
29. Chow, Y., Cui, B., Ryu, M. & Ghavamzadeh, M. Variational model-based policy optimization. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*. 2292–2299. https://doi.org/10.24963/ijcai.2021/316 (2021).
30. Clavera, I. et al. Model-based reinforcement learning via meta-policy optimization. In *Proceedings of the 2nd Conference on Robot Learning*. Vol. 87. *Proceedings of Machine Learning Research*. 617–629 (2018).
31. Tobin, J. et al. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 23–30 (IEEE, 2017).
32. Chebotar, Y. et al. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*. 8973–8979 (IEEE, 2019).
33. Uchendu, I. et al. Jump-start reinforcement learning. arXiv: 2204.02372 (2022).

34. Kober, J. & Peters, J. Imitation and reinforcement learning. *IEEE Robot. Autom. Mag.* **17**, 55–62 (2010).
35. Nakamura, Y., Mori, T., Sato, M. & Ishii, S. Reinforcement learning for a biped robot based on a CPG-actor-critic method. *Neural Netw.* **20**, 723–735. https://doi.org/10.1016/j.neunet.2007.01.002 (2007).
36. Schaal, S., Peters, J., Nakanishi, J. & Ijspeert, A. Learning movement primitives. In *Robotics Research. The Eleventh International Symposium*. 561–572. https://doi.org/10.1007/11008941_60 (2005).
37. Harada, T., Nakamura, Y., Matsumoto, Y. & Ishiguro, H. Vibration suppression control using a pattern generator for a robot driven by air actuators. In *Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics*. ROBIO '09. 756–761 (2009).
38. Whelan, P. J. Control of locomotion in the decerebrate cat. *Prog. Neurobiol.* **49**, 481–515. https://doi.org/10.1016/0301-0082(96)00028-7 (1996).
39. Matsuoka, K. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biol. Cybern.* **52**, 367–376. https://doi.org/10.1007/BF00449593 (1985).
40. Fukuoka, Y. & Kimura, H. Realization of dynamic turning motion on irregular terrain of a quadruped robot "TEKKEN4" with neural oscillators. *Trans. Jpn. Soc. Mech. Eng. Ser. C* **72**, 3848–3854. https://doi.org/10.1299/kikaic.72.3848 (2006).
41. Ijspeert, A. J., Crespi, A., Ryczko, D. & Cabelguen, J.-M. From swimming to walking with a salamander robot driven by a spinal cord model. *Science* **315**, 1416–1420. https://doi.org/10.1126/science.1138353 (2007).
42. Taga, G., Yamaguchi, Y. & Shimizu, H. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biol. Cybern.* **65**, 147–159. https://doi.org/10.1007/BF00198086 (1991).
43. Wang, J., Hu, C. & Zhu, Y. CPG-Based hierarchical locomotion control for modular quadrupedal robots using deep reinforcement learning. *IEEE Robot. Autom. Lett.* **6**, 7193–7200 (2021).
44. Deshpande, A. M., Hurd, E., Minai, A. A. & Kumar, M. DeepCPG policies for robot locomotion. *IEEE Trans. Cognit. Dev. Syst.* **15**, 2108–2121 (2023).
45. Campanaro, L., Gangapurwala, S., De Martini, D., Merkt, W. & Havoutis, I. CPG-Actor: Reinforcement learning for central pattern generators. In *Towards Autonomous Robotic Systems*. 25–35 (Springer, 2021).
46. Bellegarda, G. & Ijspeert, A. CPG-RL: Learning central pattern generators for quadruped locomotion. *IEEE Robot. Autom. Lett.* **7**, 12547–12554 (2022).
47. Hoyt, D. F. & Taylor, C. R. Gait and the energetics of locomotion in horses. *Nature* **292**, 239–240. https://doi.org/10.1038/292239a0 (1981).
48. Matsuoka, K. Analysis of a neural oscillator. *Biol. Cybern.* **104**, 297–304. https://doi.org/10.1007/s00422-011-0432-z (2011).
49. Coumans, E. & Bai, Y. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org (2016–2019).
50. Kuramoto, Y. Self-entrainment of a population of coupled non-linear oscillators. In *International Symposium on Mathematical Problems in Theoretical Physics*. 420–422. https://doi.org/10.1007/BFb0013365 (1975).
51. Matsuoka, K. Mechanisms of frequency and pattern control in the neural rhythm generators. *Biol. Cybern.* **56**, 345–353 (1987).
52. Sajjad, M. & Youngjin, C. A unified neural oscillator model for various rhythmic locomotions of snake-like robot. *Neurocomputing* **173**, 1112–1123. https://doi.org/10.1016/j.neucom.2015.08.069 (2016).
53. Owaki, D., Goda, M., Miyazawa, S. & Ishiguro, A. A minimal model describing hexapedal interlimb coordination: The tegotae-based approach. *Fronti. Neurorobot.* **11**. https://doi.org/10.3389/fnbot.2017.00029 (2017).
54. Maeda, I., Maeoka, S., Tokuda, K. & Nakajima, S. Gait emergence by discrete leg motion period control using contact signals and CPGs. *Trans. Japan Soc. Mech. Eng.* **87**, 20–00340. https://doi.org/10.1299/transjsme.20-00340 (2021).
55. Haarnoja, T., Tang, H., Abbeel, P. & Levine, S. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. 1352–1361 (2017).
56. van Hasselt, H., Guez, A. & Silver, D. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. https://doi.org/10.1609/aaai.v30i1.10295 (2016).
57. Dempster, A. P., Laird, N. M. & Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *J. R. Stat. Soc. Ser. B (Methodol.)* **39**, 1–22. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x (1977).
58. Russo, N. Rex: An open-source quadruped robot. https://github.com/nicrusso7/rex-gym (2021).
59. He, K., Zhang, X., Ren, S. & Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*. 1026–1034. https://doi.org/10.1109/ICCV.2015.123 (2015).
60. Shafiee, M., Bellegarda, G. & Ijspeert, A. Viability leads to the emergence of gait transitions in learning agile quadrupedal locomotion on challenging terrains. *Nature Communications* **15**, 3073 (2024).
61. Bradtke, S. J. & Barto, A. G. Linear least-squares algorithms for temporal difference learning. *Machine Learning* **22**, 33–57. https://doi.org/10.1023/A:1018056104778 (1996).
62. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. arXiv: 1412.6980 (2014).


## Acknowledgements

This work was supported by a Grant-in-Aid for Scientific Research S (22H04998) from the Japanese Society for the Promotion of Science (JSPS) KAKENHI, Japan, and by the New Energy and Industrial Technology Development Organization (NEDO), Japan.

## Author contributions

J.I., S.I and Y.N. conceived the study. T.W. and K.T. implemented the algorithm, and T.W., A.K., K.T., and M.T. obtained the main results, under cooperation by S.A., Y.N., H.K., and J.I. S.I. summarized the study. K.A., S.I., and Y.N. prepared the draft, and all the authors finalized the manuscript.

## Additional information

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1038/s41598-025-94163-2.

**Correspondence** and requests for materials should be addressed to S.I.

**Reprints and permissions information** is available at www.nature.com/reprints.

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

© The Author(s) 2025