CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion

Guillaume Bellegarda and Auke Ijspeert

***Abstract*— In this letter, we present a method for integrating central pattern generators (CPGs), i.e. systems of coupled oscillators, into the deep reinforcement learning (DRL) framework to produce robust and omnidirectional quadruped locomotion. The agent learns to directly modulate the intrinsic oscillator setpoints (amplitude and frequency) and coordinate rhythmic behavior among different oscillators. This approach also allows the use of DRL to explore questions related to neuroscience, namely the role of descending pathways, interoscillator couplings, and sensory feedback in gait generation. We train our policies in simulation and perform a sim-to-real transfer to the Unitree A1 quadruped, where we observe robust behavior to disturbances unseen during training, most notably to a dynamically added 13.75 kg load representing 115% of the nominal quadruped mass. We test several different observation spaces based on proprioceptive sensing and show that our framework is deployable with no domain randomization and very little feedback, where along with the oscillator states, it is possible to provide only contact booleans in the observation space.**

I. Introduction

Much progress has been made in both understanding and replicating animal locomotion through robotics. Successful implementations include biologically-inspired controllers such as Central Pattern Generators (CPGs) [1]–[4], model-based control [5]–[8], and learning-based approaches [9]–[12]. However, despite these successes and progress, the exact processes animals use to learn and produce motion remain unclear, especially related to how higher parts of the brain interact with CPGs in the spinal cord. The agility of robots also does not yet match that of animals. In this work, we draw inspiration from several of these areas to achieve legged locomotion by combining biology-inspired oscillators with the strength of deep learning.

A. Related Work

1) Biology-Inspired Control: Central Pattern Generators are neural circuits located in the spinal cords of vertebrate animals that are capable of producing coordinated patterns of high-dimensional rhythmic output signals, with evidence from nature and experimentally validated on various robot hardware [1]. Among quadrupeds, the combination of CPGs, sensory input, reflexes, and mechanical design has led to adaptive dynamic walking on irregular terrain [2]. Other works have focused on mechanical design inspired from biology for dynamic trot gaits [13]. Owaki et al. show evidence that gait generation and transitions can occur through simple force feedback, without any explicit coupling between oscillators [3], [4]. Righetti and Ijspeert added feedback from touch sensors to stop or accelerate transitions between swing/stance phases [14]. The oscillators can also be formed in task space and mapped to joint commands with inverse kinematics, with additional control for push recovery and attitude control [15]. Ajallooeian et al. augmented CPGs with both reflexes and Virtual Model Control [16], [17], and Hyun et al. presented a hierarchical controller leveraging proprioceptive impedance control for highly dynamic trot running [18]. Sensory feedback has not just been limited to proprioceptive information, for example Gay et al. used both a camera and gyroscope as feedback to a CPG to learn to walk over varying terrains [19].

This work was supported by the Swiss National Science Foundation (SNSF) as part of project No.197237. Guillaume Bellegarda and Auke Ijspeert are with the BioRobotics Laboratory, Ecole Polytechnique Federale de Lausanne (EPFL). firstname.lastname@epfl.ch

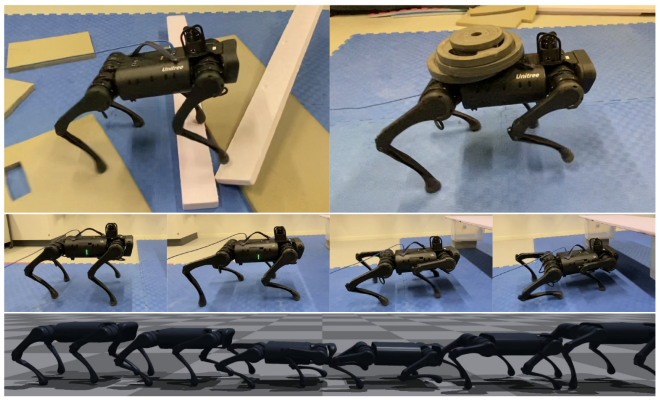

Fig. 1: Quadruped locomotion with CPG-RL. **Top:** crossing uneven terrain (left), and walking at 0.8 m/s with a 13.75 kg load representing 115% of the nominal quadruped mass (right). **Bottom:** user-specified body height modulation to crawl underneath a ledge, both experiment and simulation.

2) Model-Based Control: Conventional control approaches have shown impressive performance for real world robot locomotion [5]–[8]. Such methods typically rely on solving online optimization problems (MPC) using simplified dynamics models, which require significant domain knowledge, and may not generalize to new environments not explicitly considered during development (i.e. uneven slippery terrain). There is also considerable bias in the resulting motions due to the (for example) pre-determined foot swing trajectories, and use of Raibert heuristics [20] for foot placement planning.

3) Learning-Based Control: Deep reinforcement learning (DRL) has emerged as an attractive solution to train control policies that are robust to external disturbances and unseen environments in sim-to-real. Similarly to model-based control, recent works have shown successful results in training "blind" control policies, with only proprioceptive sensing being mapped to joint commands. For quadrupedal robots, such approaches have resulted in successful locomotion controllers on both flat [9], [10] and rough terrain [11], [12]. Other works have shown the possibility of

directly imitating animal motions [21], and the emergence of different gaits through minimizing energy consumption with DRL [22]. Recent works also integrate vision into the DRL framework for obstacle avoidance [23], dynamic crossing of rough terrains [24], [25], and gap crossing [26], [27].

4) Action Space and Phase in DRL: Directly mapping from proprioceptive sensing to joint space commands remains a challenging problem, even for deep networks and millions of training samples. In addition to the complexity of specifying desired motions like foot swing height, there are difficulties with the sim-to-real transfer arising from unmodeled dynamics such as the actuators. To better structure the learning problem and avoid undesired local optima, recent works are exploring different action spaces (for example directly learning torques [28], or desired task space positions [29]–[31]), and in particular encoding a time component (phase) into the framework, such as Policies Modulating Trajectory Generators (PMTG) for Minitaur [32], or for ANYmal [12], [24]. Additionally, incorporating phase encodings facilitates learning different gaits [33], and can also be used together with MPC [34].

Learning with more biologically inspired approaches with explicit use of CPGs has also been shown in simulation for bipeds [35], [36]. Li et al. use reinforcement learning to directly learn the feedback terms of a CPG for a biped to tackle different slopes [35]. Kasaei et al. use DRL to update the parameters of a CPG-ZMP walk engine and output joint target residuals to adapt to commands and different terrains [36]. For quadrupeds, Shi et al. learn both the parameters of a foot trajectory generator as well as joint target residuals to locomote in a variety of environments including stairs and slopes [37].

B. Contribution

In contrast to previous work, in this paper we propose to use deep reinforcement learning to directly learn the time-varying oscillator intrinsic amplitude and frequency for each oscillator which together forms the Central Pattern Generator. We implement the CPG network with one oscillator per limb, but without explicit couplings between oscillators, similarly to the work of Owaki et al [4]. Couplings between oscillators are known to exist in biological CPGs, but recent work has shown that they might be weaker than previously thought [4], [38], and that sensory feedback and descending modulation might play an important role in inter-oscillator synchronization. This biology-inspired approach to locomotion additionally mitigates several drawbacks usually encountered each with CPGs and deep reinforcement learning.

On the CPG side, parameter tuning remains difficult, often necessitating an expert doing so by hand, or the use of genetic algorithms. The tuning of these parameters usually results in only one gait, whereas animals exhibit a continuum of motions at different speeds and directions. Specified (strong) couplings also result in rigid and not completely natural gaits, and the role of sensory feedback and reflexes must then also be tuned and added on top of the CPG.

On the other hand, when applying deep reinforcement learning algorithms for control tasks, mapping from sensory information to joint commands often results in non-intuitive motions, with great care being needed to properly tune reward functions (i.e., how to specify a desired swing foot height). Additionally, the sim-to-real transfer presents added difficulties from non-modeled dynamics and possible overfitting of simulator physics engine parameters.

In this work we address all of the above difficulties, and train and deploy our CPG-RL controller on the Unitree A1 quadruped, shown in Figure 1. Some highlights of our method include:

- fast training and ease of sim-to-real transfer (i.e. without any domain randomization or added noise in simulation)
- a minimal amount of sensory information needed in the observation space (i.e. feedback only from foot contact booleans)
- on the fly parameter selection to achieve locomotion at user-specified body heights and foot swing heights, without any needed specification in the DRL framework
- robustness to disturbances not seen in training, i.e. uneven terrain, and a dynamically added 13.75 $\mathtt{kg}$ load representing 115% of the nominal quadruped mass

The rest of this paper is organized as follows. Section II provides background details on deep reinforcement learning and Central Pattern Generators. Section III describes our design choices and integration of Central Pattern Generators into the deep reinforcement learning framework (CPG-RL). Section IV shows results and analysis from learning our controller and sim-to-sim and sim-to-real transfers for several scenarios including minimal observation space, online modulating body height and swing foot ground clearance, and significant disturbances in terrain and added load. Finally, a brief conclusion is given in Section V.

II. Background

A. Reinforcement Learning

In the reinforcement learning framework [39], an agent interacts with an environment modeled as a Markov Decision Process (MDP). An MDP is given by a 4-tuple $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R})$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions available to the agent, $\mathcal{P}:\mathcal{S}\times\mathcal{A}\times\mathcal{S}\rightarrow\mathbb{R}$ is the transition function, where $\mathcal{P}(s_{t+1}|s_t,a_t)$ gives the probability of being in state s_t, taking action a_t, and ending up in state s_{t+1}, and $\mathcal{R}:\mathcal{S}\times\mathcal{A}\times\mathcal{S}\rightarrow\mathbb{R}$ is the reward function, where $\mathcal{R}(s_t,a_t,s_{t+1})$ gives the expected reward for being in state s_t, taking action a_t, and ending in state s_{t+1}. The agent goal is to learn a policy π that maximizes its expected return for a given task.

B. Central Pattern Generators

While a number of neural oscillators have been proposed to implement CPG circuits, here we focus on the amplitude-controlled phase oscillators from [40] used to produce both swimming and walking behaviors on a salamander robot:

$$\ddot{r}_i = a\Big(\frac{a}{4}(\mu_i - r_i) - \dot{r}_i\Big) \tag{1}$$

$$\dot{\theta}_i = \omega_i + \sum_j r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij}) \tag{2}$$

where r_i is the current amplitude of the oscillator, θ_i is the phase of the oscillator, μ_i and ω_i are the intrinsic amplitude and frequency, a is a positive constant representing the convergence factor. Couplings between oscillators are defined by the weights w_{ij} and phase biases ϕ_{ij}.

Regardless of the particular oscillator selection, several choices exist for the number of oscillators in the network, as well as how to map the output back to joint commands. For example, one oscillator can be used for each joint to directly produce motion in joint space [13], [40], or the oscillator can be formed in task space and mapped to joint commands with inverse kinematics [14]. Thus, this formulation results in numerous design decisions and parameters that must be tuned, commonly by hand or through evolutionary algorithms such as CMA-ES [41]. Notably, this tuning generally results in specific fixed gaits which may not be robust to external disturbances.

III. Learning Central Pattern Generators

In this section we describe our CPG-integrated reinforcement learning framework and design decisions for learning locomotion controllers for quadruped robots. At a high level, the agent learns to modulate the CPG parameters for each leg with deep reinforcement learning. This allows for adaptation based on sensory feedback along with the knowledge of the current CPG state. This type of learning represents an approximation of motor learning in animals, namely how higher brain centers such as the motor cortex and the cerebellum learn to send modulation signals to CPG circuits in the spinal cord. The control diagram for our method is shown in Figure 2, and we discuss all components below.

A. Action Space

We first consider the coupled oscillators in Equations 1 and 2 (one for each leg, or $i \in \{1,2,3,4\}$), whose output we wish to map to foot trajectories in Cartesian space similar to [14]. Such a strategy poses two issues: 1) the coupling severely biases the rhythm into potentially unnatural motions, and 2) with only two variables the motions are restricted to a single plane. Inspired by animals, which frequently deviate and transition between various gaits and produce omnidirectional motions, we solve both issues by removing any explicit coupling ($w_{ij} = 0$), with the intuition that the agent should learn to coordinate its limbs on its own (as opposed to being fixed by the CPG coupling topology), and we add another state variable, ϕ, to orient motion in any direction (see right of Figure 3). Thus, for each limb we define the following oscillator:

$$\ddot{r}_i = a\left(\frac{a}{4}(\mu_i - r_i) - \dot{r}_i\right) \tag{3}$$

$$\dot{\theta}_i = \omega_i \tag{4}$$

$$\dot{\phi}_i = \psi_i \tag{5}$$

In contrast to recent works making use of phase in reinforcement learning, we propose an action space to directly modulate the intrinsic oscillator amplitude and phases, by learning to modulate μ_i, ω_i, and ψ_i for each leg. This allows the agent to adapt each of these states online in real-time (for instance increasing the amplitude to step farther or accelerating a leg movement during swing), compared with the more traditional CPG approach of optimizing for only a single set of fixed parameters. Additionally, we hypothesize that this framework will lead to more interpretable results, as we can directly observe how the agent modulates the oscillators depending on feedback from both the environment as well as from the current oscillator states (for example in contrast to directly learning joint commands). Thus, for the omnidirectional locomotion task, our action space can be summarized as $\mathbf{a} = [\boldsymbol{\mu}, \boldsymbol{\omega}, \boldsymbol{\psi}] \in \mathbb{R}^{12}$. The agent selects these parameters at 100 Hz, and we use the following limits for each input during training: $\mu \in [1,2]$, $\omega \in [0,4.5]$ Hz, and $\psi \in [-1.5,1.5]$ Hz, and $a = 150$.

To map from the oscillator states to joint commands, we first compute corresponding desired foot positions, and then calculate the desired joint positions with inverse kinematics. The desired foot position coordinates are given as follows:

$$x_{i,\text{foot}} = -d_{step}(r_i - 1)\cos(\theta_i)\cos(\phi_i) \tag{6}$$

$$y_{i,\text{foot}} = -d_{step}(r_i - 1)\cos(\theta_i)\sin(\phi_i) \tag{7}$$

$$z_{i,\text{foot}} = \begin{cases} -h + g_c \sin(\theta_i) & \text{if } \sin(\theta_i) > 0 \\ -h + g_p \sin(\theta_i) & \text{otherwise} \end{cases} \tag{8}$$

where d_{step} is the maximum step length, h is the robot height, g_c is the max ground clearance during swing, and g_p is the max ground penetration during stance. A sample visualization of the foot trajectory for a set of these parameters is shown at left of Figure 3. These parameters make it possible to specify behaviors that are in general difficult to learn when directly outputting joint commands. For example, specifying a foot swing height would usually necessitate keeping track of a history of states and exasperates the temporal credit assignment problem of reinforcement learning. With our framework, we randomly sample h, g_c, and g_p during training (i.e. the agent has no explicit observation of these parameters) to learn continuous behavior, which then allows the user to specify both a robot height and swing foot ground clearance during deployment.

B. Observation Space

We consider three observation spaces in this work with varying amounts of proprioceptive sensing: full (obs_{full}), medium (obs_{med}) and minimal (obs_{min}). Our purpose is to investigate how much the locomotor performance depends on the richness of the observation space, and also to identify which information is sufficient to reach reasonable performance. This investigation is also interesting for neuroscience to explore which type of sensor modality is more important than others for generating stable gaits. The CPG states are always in the observation space, and compared with the proprioceptive sensing which is subject to measurement noise from onboard sensors, the CPG states are always known. This provides a source of stability to the method and we believe eases the sim-to-real transfer.

Full observation: The full observation consists of velocity commands and measurements reasonably available with proprioceptive sensing, and are becoming standard in DRL

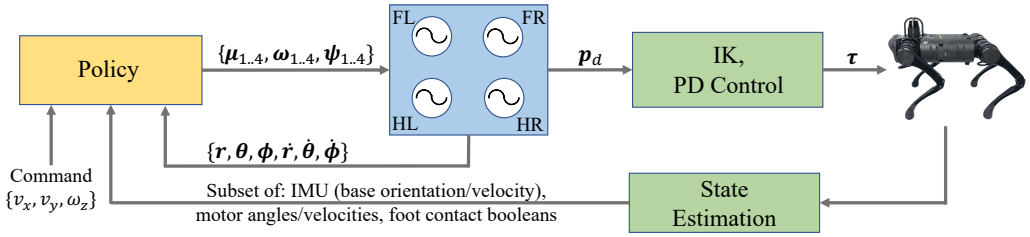

Fig. 2: The CPG-RL control architecture. Given an observation consisting of velocity commands, a subset of proprioceptive measurements, and the current CPG states, the policy network selects CPG parameters μ, ω, and ψ for each leg i (Front Left (FL), Front Right (FR), Hind Left (HL), Hind Right (HR)). The CPG states are mapped to desired foot positions $\boldsymbol{p}_d$, which are then converted to desired joint angles with inverse kinematics, and finally tracked with joint PD control to produce torques $\boldsymbol{\tau}$. The control policy selects actions at 100 Hz, and all other blocks operate at 1 kHz.

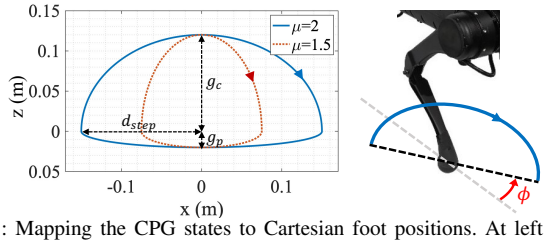

Fig. 3: Mapping the CPG states to Cartesian foot positions. At left in the XZ-plane: ground clearance (g_c), ground penetration (g_p), max step length (d_{step}) are design parameters, whereas CPG states r and θ control amplitude and frequency. Two trajectories are shown representing the mapping for two converged amplitude set points, μ, which the agent can directly modulate to vary the step length. At right, CPG state ϕ controls the orientation of the trajectory with respect to the body, shown for the Front Right (FR) leg.

approaches. These include the body state (orientation, linear and angular velocities), joint state (positions, velocities), and foot contact booleans. The last action chosen by the policy network and CPG states $\{\boldsymbol{r},\dot{\boldsymbol{r}},\boldsymbol{\theta},\dot{\boldsymbol{\theta}},(\boldsymbol{\phi},\dot{\boldsymbol{\phi}})\}$ are concatenated to the proprioceptive measurements.

Medium observation: The medium observation removes the joint state and last action from the full observation. This observation space is chosen to show that joint information is actually not necessary for omnidirectional locomotion through our method. Other states remain the same (i.e. velocity commands, body state, foot contact booleans, and CPG states).

Minimal observation: The minimal observation space consists only of foot contact booleans and the CPG states $\{\boldsymbol{r},\dot{\boldsymbol{r}},\boldsymbol{\theta},\dot{\boldsymbol{\theta}}\}$. This observation space shows that coordination between limbs can be accomplished with very little sensing at all, with the only environmental feedback being from foot contact booleans. The idea for this space is inspired from the force feedback term in traditional CPGs shown to coordinate transitions between gaits [3], [4], also known as *Tegotae*-based control [42]. The importance of contacts and limb loading has also been shown by Ekeberg and Pearson in a simulation of cat locomotion [43]. For this observation space, the task is only to move forward at a particular desired velocity $v^*_{b,x}$.

C. Reward Function

We design our reward function to track desired body velocity commands in the body frame x and y directions as well as a desired yaw rate $\omega^*_{b,z}$. We include terms to minimize other undesired body velocities as well as penalize the work (aiming at keeping the body stable and minimizing the energy consumption). More precisely, the reward function is a summation of the following terms:

- linear velocity tracking, body x direction: $f(v^*_{b,x} - v_{b,x})$
- linear velocity tracking, body y direction: $f(v^*_{b,y} - v_{b,y})$

TABLE I: PPO Hyperparameters and neural network size.

Parameter	Value
Batch size	98304 (4096x24)
Mini-bach size	24576 (4096x6)
Number of epochs	5
Clip range	0.2
Entropy coefficient	0.01
Discount factor	0.99
GAE discount factor	0.95
Desired KL-divergence kl^*	0.01
Learning rate α	adaptive
Number of hidden layers (all networks)	3
Number of hidden units per layer	[512, 256, 128]
Activation	elu

- angular velocity tracking, body yaw rate: $f(\omega^*_{b,z} - \omega_{b,z})$
- linear velocity penalty in body z direction: $-v^2_{b,z}$
- angular velocity penalty (body roll and pitch rates): $-||\boldsymbol{\omega}_{b,xy}||^2$
- work: $-|\boldsymbol{\tau} \cdot (\dot{\boldsymbol{q}}^t - \dot{\boldsymbol{q}}^{t-1})|$

where $(\cdot)^*$ represents a desired command, and $f(x) := \exp\left(-\frac{||x||^2}{0.25}\right)$. These terms are weighted with $0.75dt$, $0.75dt$, $0.5dt$, $2dt$, $0.05dt$, $0.001dt$, where $dt = 0.01$ is the control policy time step. Notably, as discussed in Section III-A, we do not need to put any additional terms on foot swing time. Compared with other learning-based approaches for omnidirectional locomotion, this is a simple set of terms to properly specify desired behavior.

D. Training Details

We use Isaac Gym [25], [44] with PhysX as our physics engine and training environment, and the Unitree A1 quadruped [45]. This framework allows for high throughput, where we simulate 4096 A1s in parallel on a single NVIDIA RTX 3070 GPU. We use PPO [46] to train the policy, and the relevant hyperparameters and neural network architecture details (multilayer perceptron with 3 hidden layers) are listed in Table I. With this framework, similar to in [25], we can learn control policies within minutes.

The maximum episode length is 20 seconds, and the environment resets for an agent if the base or a thigh comes in contact with the ground. The terrain is always flat during training. With each reset, we sample new parameters h and g_c for the mapping from oscillator states to joint commands so the agent can learn to locomote at varying body heights and step heights. New velocity commands $\{v^*_{b,x}, v^*_{b,y}, \omega^*_{b,z}\}$ are sampled every 5 seconds. Although we find that domain randomization is not strictly needed to perform a sim-to-real transfer, unless specified we randomize the following parameters during training (kept constant during an episode):

- **ground coefficient of friction** varied in [0.3, 1]
- **limb mass** varied within 20% of nominal values

- **added base mass** up to 5 kg
- **external push** of up to 0.5 m/s applied in a random direction to the base every 15 seconds

No noise is added to the observation.

The control frequency of the policy is 100 Hz, and the torques computed from the desired joint positions are updated at 1 kHz. The equations for each of the oscillators (Eqns. 3-5) are thus also integrated at 1 kHz. The joint PD controller gains are $K_p = 100$, $K_d = 2$. For the sim-to-real transfer, all proprioceptive information is measured from the Unitree sensors.

IV. Experimental Results and Discussion

In this section we report results from learning locomotion controllers with CPG-RL. We seek to evaluate the necessity of sensory information as defined by the three observation spaces, the difficulty of the sim-to-real transfer, the interpretability of the resulting policy, and the robustness to various disturbances not seen during training. Snapshots of some of our results are shown in Figure 1, and the supplementary video shows clear visualizations of the discussed experiments.

A. Sim-to-Real Experimental Results

1) CPG State Modulation: In the video, we examine how the agent trained with CPG-RL coordinates and modulates the CPG states to produce locomotion. We verify how similar the resulting gait is compared with fixed open-loop CPG gaits (trot, walk, pace) as generated by Equations 1 and 2, tuned for locomotion at particular frequencies.

We command the agent to walk forward at 0.8 m/s while adding variable mass up to 13.75 kg. One second of the CPG amplitude and phase plots is shown in Figure 4, where we observe an approximate trot gait with a cycle of approximately 0.5 seconds. The swing duration (when $0 \leq \theta \leq \pi$) can be observed to be lower than the stance duration, as is typical of quadruped animals. It is additionally apparent that there is coordination between phase and amplitude, the latter of which is not quite as periodic or constant, implying the agent uses it more to adapt to sensory feedback. In general however the amplitude appears higher when in stance phase, showing the agent uses this time to push backwards and propel the quadruped forward. This result is also apparent by examining the leg frame foot XZ trajectories, shown in Figure 5. Notably, the trajectories for both front feet are significantly modulated from the nominal task space trajectory used in the open-loop trot gait in the video, where now the foot is primarily underneath and behind the hip in the leg frame. Compared with the fixed open-loop CPG gaits, CPG-RL produces a continuum of more natural gaits that are less rigid, more efficient, more robust, and have lower frequency.

2) Observation Space Effects: As can be seen in Figure 1 and the video, we achieve robust sim-to-real transfer with and without joint state information (i.e. using either the full or medium observation spaces, obs_{full} and obs_{med}). This result holds for omnidirectional motions at varying velocities for scenarios including uneven terrain, push recovery, and

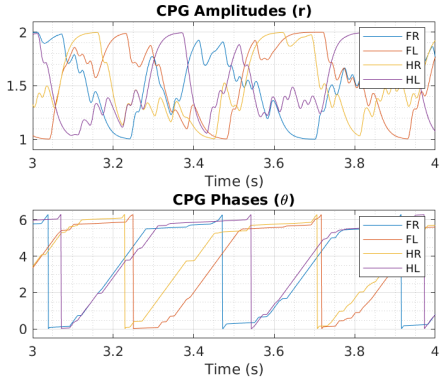

Fig. 4: CPG states for 1 second of a deployed policy locomoting at 0.8 m/s. An approximate trot gait can be observed, with faster swing time ($0 \leq \theta \leq \pi$) than stance time. Coordination between the amplitude and phase variables can be seen, noticeably increasing amplitude in stance phase to push backward and propel the quadruped forward.

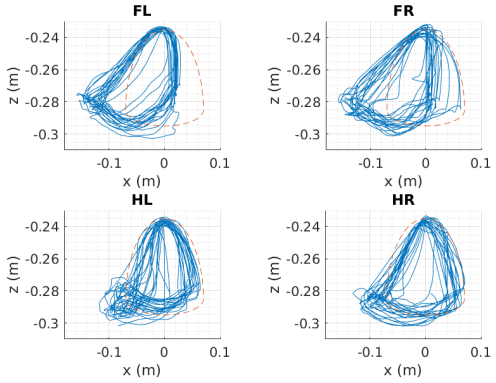

Fig. 5: Leg frame foot XZ trajectories for a deployed policy locomoting at 0.8 m/s with dynamically added mass. Significant modulations can be seen to the sample task space trajectory represented by the dotted line. Notably, the amplitude for the front feet shifts the trajectory to lie mostly behind the hip.

significant added loads encompassing 115% of the nominal robot mass (see Section IV-A.5). This result is attributable to the presence of the oscillator states in the observation space, and its mapping back to the task space trajectories.

We are also able to transfer the CPG-RL obs_{min} policy, for which the only feedback from the environment that the agent observes is through the contact booleans, and is trained without any simulator noise or randomization. While the agent has no direct observation of its body velocity, it can relate the frequencies of the neural oscillator states (which it has direct control over) to the reward it receives at each time step, to track 0.5 m/s in the body x direction while keeping all other velocities 0. This result supports that coordination between limbs is possible through very little sensory feedback [3], [4], [42].

We also tested training policies without the foot contact feedback, i.e. with only CPG states in the observation space, but the agent was unable to learn to locomote at any fixed velocity. This shows that some sort of feedback is necessary to coordinate locomotion.

3) Body Height and Swing Foot Height Online Modulation: As discussed in Section III-A, our framework naturally allows the user to specify the body height and foot ground clearance through setting parameters h and g_c for the mapping from oscillator states to desired foot positions. The agent has no explicit knowledge of these parameters,

but in the full observation space it can find a relationship from observing the direct effects on the joint positions.

Once training is complete a user can change either of these parameters in real time, as the agent continues to locomote at its velocity commands. In Figure 1 and in the video we show an example of lowering the body height h from its nominal height of 0.30 $\mathtt{m}$ to 0.19 $\mathtt{m}$ in order to crawl underneath a ledge, and then increasing it back to 0.30 $\mathtt{m}$ on the other side. We also test varying the foot ground clearance g_c online, changing it from 0.03 $\mathtt{m}$ to 0.12 $\mathtt{m}$ (almost half of the nominal robot height), and then back down to 0.03 $\mathtt{m}$.

4) Uneven Terrain: As discussed in Section III-D, we train our policies only on flat terrain, though the coefficient of friction is varied. Notably, the rigid body dynamics used in simulation are not exactly representative of the hardware, as the A1 feet undergo significant deformations on contact, in addition to other sim-to-real considerations such as the lack of motor modeling. We tested adding debris such as soft foam and hard styrofoam (see Figure 1 and video) and found the policy to be robust to such terrain. These materials are very light and easy to kick and crumple by A1, and even when the material got stuck or rolled up, causing swing feet to catch and impeding progress, it did not immediately fall.

5) Added Mass: We also tested adding varying mass to the robot while it was trotting at various velocities (0.3-0.8 $\mathtt{m/s}$). In all cases there was no noticeable drop in performance, which can be attributed to both the robustness of the method as well as the high joint gains we are able to use. We achieved trotting at 0.8 $\mathtt{m/s}$ with 13.75 $\mathtt{kg}$ (115% of nominal robot mass) dynamically added to the robot, which was all of the available mass we had in the lab. To the best of our knowledge, this represents the highest robustness against loads achieved on A1. Other comparisons include RMA [11] achieving up to a 12 $\mathtt{kg}$ load at lower velocity, a model-based method achieving 11 $\mathtt{kg}$ while standing [7], and the Unitree default model-based controller has a max rating of 5 $\mathtt{kg}$ [45].

Remarkably, we used the domain randomization explained in Section III-D, which only added up to 5 $\mathtt{kg}$ in simulation, again showing significant robustness to out of distribution disturbances. Additionally, as we achieve the same results with both the medium and full observation spaces, we show that joint state information is actually not necessary for dynamic locomotion with our method.

B. Sim-to-Sim Comparison with Learning in Joint Space

To evaluate the benefits of our approach, we compare CPG-RL with a standard joint space training pipeline, which takes as input proprioceptive measurements, and outputs joint space offsets from a nominal resting position using the same method from [25], which we call $Joint\ PD$. The observation space contains all proprioceptive sensing used in the full observation space by CPG-RL, without the oscillator states. All training hyperparameters, environment details, and neural network architectures remain the same. To train the $Joint\ PD$ baseline, we compare using both (a) the same reward function as described in Section III-C, as well as (b) the more complex *special* reward function from [25], which additionally includes terms for orientation, joint acceleration, joint velocity, joint torque, action rate, collisions, feet air time, and base height.

TABLE II: Sim-to-sim tracking of velocity command $v^*_{b,x} = 0.5$ $\mathtt{m/s}$ with different observation spaces and methods trained to locomote for $v^*_{b,x} \in [0.2,1]$ $(\mathtt{m/s})$ (except for CPG-RL with obs_{min}, which is trained to exclusively track $v^*_{b,x} = 0.5$ $\mathtt{m/s}$). We compare performance when training with/without randomization (Sec. III-D).

Mean Quantity	CPG-RL						Joint PD [25]	
Obs. Space	obs_{full}		obs_{med}		obs_{min}		obs_{full}	
Training Randomization?	X	✓	X	✓	X	✓	X	✓
$v_{b,x}$ $(\mathtt{m/s})$	0.494	0.493	0.491	0.488	0.460	0.472	0.486	0.471
Duty Factor	1.74	2.10	1.48	1.75	2.63	2.83	0.87	0.89
Period $(\mathtt{s})$	0.75	0.70	0.70	0.75	0.90	0.75	0.55	0.60
Body Height $(\mathtt{m})$	0.17 - 0.30						0.29	0.30
Foot Ground Clearance $(\mathtt{m})$	0.03 - 0.12						0.025	0.02

TABLE III: Sim-to-sim tracking ability of omnidirectional commands with different observation spaces and methods trained to locomote in the following ranges: $v^*_{b,x} \in [-1, 1]$ $(\mathtt{m/s})$, $v^*_{b,y} \in [-1, 1]$ $(\mathtt{m/s})$, $\omega^*_{b,z} \in [-1, 1]$ $(\mathtt{rad/s})$. We compare performance when training with/without randomization (Sec. III-D).

Command : Actual	CPG-RL				Joint PD [25]	
Obs. Space	obs_{full}		obs_{med}		obs_{full}	
Training Randomization?	X	✓	X	✓	X	✓
$v^*_{b,x} = 0.5$ $(\mathtt{m/s})$	0.447	**0.488**	0.545	0.466	0.467	0.421
$v^*_{b,y} = 0.5$ $(\mathtt{m/s})$	0.483	0.481	0.359	**0.510**	0.260	0.483
$v^*_{b,x} = 0.5$ $(\mathtt{m/s})$	0.470	**0.492**	0.437	0.446	0.435	0.418
$v^*_{b,y} = 0.5$ $(\mathtt{m/s})$	0.476	**0.490**	0.436	0.478	0.366	0.459
$\omega^*_{b,z} = 0.6$ $(\mathtt{rad/s})$	0.463	**0.578**	0.544	0.537	0.418	0.537

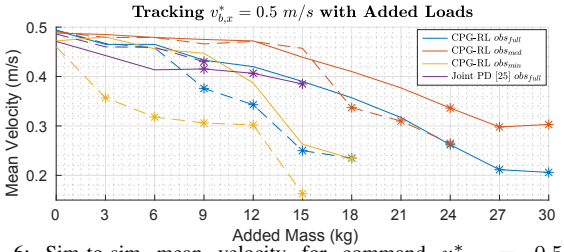

Fig. 6: Sim-to-sim mean velocity for command $v^*_{b,x} = 0.5$ $\mathtt{m/s}$ under increasing added loads. The dashed lines represent the same method/observation trained without any noise in Isaac Gym. The $*$s represent when performance is not guaranteed: the robot sometimes falls down, or cannot make consistent forward progress.

The video shows training curves for learning locomotion policies with CPG-RL as well as with the $Joint\ PD$ baseline trained with both reward functions (a) and (b). While all three methods provide similar returns as training progresses, the policies trained with CPG-RL produce the most natural looking gaits, are easiest to interpret, and allow the user to set swing foot ground clearance. In contrast, $Joint\ PD$ policies trained with the same reward function (a) result in unnatural gaits that overfit the simulator dynamics (shown in the video), and are unlikely to transfer well in sim-to-real. $Joint\ PD$ policies trained with the more complex *special* reward function (b) can achieve more natural gaits, however tracking accuracy is lower (Tables II and III), and it is not possible to directly control the swing foot ground clearance.

Quantitatively, we evaluate and compare the performance of CPG-RL as well as $Joint\ PD$ policies [25] in a sim-to-sim transfer from Isaac Gym with the PhysX physics engine to Gazebo with the ODE physics engine. While a successful sim-to-sim transfer does not necessarily guarantee a successful sim-to-real transfer, sim-to-sim allows safely testing many

policies without risking damaging the hardware, as used in recent work to verify the agent has not overfit the training simulator's dynamics before sim-to-real transfers [28], [31]. While there is no noise added to the observation space in Isaac Gym which uses ground truth data, in Gazebo we simulate the state estimation (i.e. Kalman Filter) as used on the real hardware. For each method (CPG-RL or $Joint\ PD$ [25]) and observation space (obs_{full}, obs_{med}, obs_{min}), we also compare performance when training with and without randomization as described in Section III-D.

1) Sim-to-sim Forward Locomotion: We first train policies (at least 10) for each method and observation space with the goal of tracking only forward velocities in the body x direction, namely $v^*_{b,x} \in [0.2, 1]$ $(\mathrm{m/s})$. Table II shows sim-to-sim tracking performance of desired command $v^*_{b,x} = 0.5$ $(\mathrm{m/s})$, where we present the mean velocity, duty factor, gait period, as well as body height and foot ground clearance. The best tracking performance is for CPG-RL with both full and medium observation spaces. The mean duty factor and gait period are much higher for policies trained with CPG-RL, suggesting greater stability. Compared with $Joint\ PD$ policies which learn to locomote at a fixed body height and ground clearance, all CPG-RL policies can vary height and foot ground clearance online.

2) Sim-to-sim with Added Loads: We take the same policies and repeat the transfers while adding loads to the robot and still commanding $v^*_{b,x} = 0.5$ $(\mathrm{m/s})$. Figure 6 shows the mean velocity tracking performance for added masses from 0 to 30 kg, in increments of 3 kg. All policies, whether trained with noise (i.e. up to 5 kg in Isaac Gym) or without, are able to make at least some forward progress with loads up to 9 kg. The points labeled with $*$s indicate that performance is not guaranteed: the robot either stops or falls down, with increasing probability for higher loads. The dashed lines show the performance of policies trained without any noise in Isaac Gym, which notably still allows all policies trained with CPG-RL to locomote with 15 kg, even for the minimal observation space obs_{min}. Interestingly, the policies trained with CPG-RL and obs_{med} perform better than policies trained with obs_{full}. Under higher disturbances, the joint states are an additional source of noise and may take value combinations unseen in training, which may shift the expected distribution the agent has learned to map between observations and actions. The results show that CPG-RL allows sim-to-sim transfer with loads representing 250% of the nominal robot mass, while trained with noise of only up to 42% of the nominal robot mass.

3) Sim-to-sim Omnidirectional Commands: We next train policies (at least 10) for each method and observation space to track omnidirectional commands in the following ranges: $v^*_{b,x} \in [-1, 1]$ $(\mathrm{m/s})$, $v^*_{b,y} \in [-1, 1]$ $(\mathrm{m/s})$, $\omega^*_{b,z} \in [-1, 1]$ $(\mathrm{rad/s})$. Table III shows sim-to-sim tracking performance of various commanded omnidirectional velocities within these ranges. The data shows that CPG-RL policies can closely track desired forward, lateral, and angular velocities, as well as combinations of these, even when trained without noise. In contrast, we note that in addition to not tracking the commands as accurately, transferring the omnidirectional $Joint\ PD$ policies comes with several added difficulties compared to CPG-RL. As the training curve converges, there is a very small window (can be fewer than 50 iterations) for which the $Joint\ PD$ policies are able to transfer sim-to-sim, which is also not consistent across different random seeds. If training continues, while the average return does not increase, the resulting policies become more and more unnatural as the agent exploits the simulator dynamics (we observe tiny steps with the front limbs, and both rear limbs in the air), and are unable to transfer sim-to-sim. We also observe that training $Joint\ PD$ policies requires much more tuning and design decisions, including reward function tuning, dynamics randomization parameter tuning, possible motor modeling, etc. which overall results in longer training times compared with CPG-RL.

4) Joint PD Observation Spaces: While possible to learn omnidirectional locomotion with $Joint\ PD$ and obs_{med}, such policies are unable to transfer due to learning high frequency small-step gaits that exploit the simulator dynamics, lacking feedback from the joint states. Training in joint space with obs_{min} is unable to learn any locomotion policy.

V. Conclusion

In this work we have presented CPG-RL, a framework for learning and modulating intrinsic oscillator amplitudes and frequencies to coordinate rhythmic behavior among limbs to achieve quadruped locomotion. Our results have shown that this method results in fast training and ease of sim-to-real transfer, where we show successful transfers with no domain randomization and only minimal sensory feedback. Additionally, the framework allows the user to easily specify (on the fly and/or in training) desired legged robot quantities like body height and swing foot ground clearance, the latter of which can be challenging to specify through reward shaping in end-to-end learning frameworks. The framework also proved robust to disturbances not seen in training, for example A1 was able to walk over uneven terrain or with added mass 2.75x greater than in training (115% of nominal robot mass) in hardware, and 250% of the nominal robot mass in sim-to-sim. To the best of our knowledge, this represents the highest robustness against loads so far achieved on the Unitree A1 quadruped.

In terms of neuroscience, the use of DRL will allow us in the future to explore questions that are still open in animal motor control, namely the exact roles and interactions of descending pathways, interoscillator couplings within CPG networks, and sensory feedback in gait generation. The results presented here suggest (i) that descending pathways are more effective at modulating locomotion by acting on the CPG circuits rather than directly on muscles (CPG-RL performs better than $Joint\ PD$), (ii) that stable locomotion can be obtained with non-existent (or weak) interoscillator couplings, and (iii) that sensing contact (or loading) in the limb is one of the most important sensory information. This last point is in agreement with the conclusion of a neuromechanical simulation of cat locomotion [43].

ACKNOWLEDGEMENTS

We would like to thank Milad Shafiee and Alessandro Crespi for assisting with hardware setup and experiments.

REFERENCES

[1] 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, robotics and Neuroscience.

[2] Y. Fukuoka, H. Kimura, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts," *The International Journal of Robotics Research*, vol. 22, no. 3-4, pp. 187–202, 2003.

[3] D. Owaki, T. Kano, K. Nagasawa, A. Tero, and A. Ishiguro, "Simple robot suggests physical interlimb communication is essential for quadruped walking," *Journal of The Royal Society Interface*, vol. 10, no. 78, p. 20120669, 2013.

[4] D. Owaki and A. Ishiguro, "A quadruped robot exhibiting spontaneous gait transitions from walking to trotting to galloping," *Scientific reports*, vol. 7, no. 1, pp. 1–10, 2017.

[5] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[6] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[7] M. Sombolestan, Y. Chen, and Q. Nguyen, "Adaptive force-based control for legged robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7440–7447.

[8] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, 2018.

[9] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots." in *Robotics: Science and Systems*, 2018.

[10] 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.

[11] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[12] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020.

[13] A. Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 932–950, 2013.

[14] L. Righetti and A. J. Ijspeert, "Pattern generators with sensory feedback for the control of quadruped locomotion," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 819–824.

[15] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2554–2561.

[16] M. Ajallooeian, S. Pouya, A. Sproewitz, and A. J. Ijspeert, "Central pattern generators augmented with virtual model control for quadruped rough terrain locomotion," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 3321–3328.

[17] M. Ajallooeian, S. Gay, A. Tuleu, A. Spröwitz, and A. J. Ijspeert, "Modular control of limit cycle locomotion over unperceived rough terrain," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3390–3397.

[18] D. J. Hyun, S. Seok, J. Lee, and S. Kim, "High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah," *The International Journal of Robotics Research*, vol. 33, no. 11, pp. 1417–1445, 2014.

[19] S. Gay, J. Santos-Victor, and A. Ijspeert, "Learning robot gait stability using neural networks as sensory feedback function for central pattern generators," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 194–201.

[20] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[21] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," 2020.

[22] Z. Fu, A. Kumar, J. Malik, and D. Pathak, "Minimizing energy consumption leads to the emergence of gaits in legged robots," in *5th Annual Conference on Robot Learning*, 2021.

[23] R. Yang, M. Zhang, N. Hansen, H. Xu, and X. Wang, "Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers," *arXiv preprint arXiv:2107.03996*, 2021.

[24] T. Miki, J. Lee, J. Hwanbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, 2022.

[25] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *5th Annual Conference on Robot Learning*, 2021.

[26] W. Yu, D. Jain, A. Escontrela, A. Iscen, P. Xu, E. Coumans, S. Ha, J. Tan, and T. Zhang, "Visual-locomotion: Learning to walk on complex terrains with vision," in *5th Annual Conference on Robot Learning*, 2021.

[27] G. B. Margolis, T. Chen, K. Paigwar, X. Fu, D. Kim, S. bae Kim, and P. Agrawal, "Learning to jump from pixels," in *5th Annual Conference on Robot Learning*, 2021.

[28] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion," *arXiv preprint arXiv:2203.05194*, 2022.

[29] G. Bellegarda and K. Byl, "Training in task space to speed up and guide reinforcement learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 2693–2699.

[30] G. Bellegarda and Q. Nguyen, "Robust quadruped jumping via deep reinforcement learning," *arXiv preprint arXiv:2011.07089*, 2020.

[31] G. Bellegarda and Q. Nguyen, "Robust high-speed running for quadruped robots via deep reinforcement learning," *arXiv preprint arXiv:2103.06484*, 2021.

[32] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, "Policies modulating trajectory generators," in *Conference on Robot Learning*. PMLR, 2018, pp. 916–926.

[33] Y. Shao, Y. Jin, X. Liu, W. He, H. Wang, and W. Yang, "Learning free gait transition for quadruped robots via phase-guided controller," *arXiv preprint arXiv:2201.00206*, 2022.

[34] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots, "Fast and efficient locomotion via learned gait transitions," in *5th Annual Conference on Robot Learning*, 2021.

[35] C. Li, R. Lowe, and T. Ziemke, "Humanoids learning to walk: A natural cpg-actor-critic architecture," *Frontiers in Neurorobotics*, vol. 7, p. 5, 2013.

[36] M. Kasaei, M. Abreu, N. Lau, A. Pereira, and L. P. Reis, "A cpg-based agile and versatile locomotion framework using proximal symmetry loss," *arXiv preprint arXiv:2103.00928*, 2021.

[37] H. Shi, B. Zhou, H. Zeng, F. Wang, Y. Dong, J. Li, K. Wang, H. Tian, and M. Q.-H. Meng, "Reinforcement learning with evolutionary trajectory generator: A general approach for quadrupedal locomotion," *arXiv preprint arXiv:2109.06409*, 2021.

[38] R. Thandiackal, K. Melo, L. Paez, J. Herault, T. Kano, K. Akiyama, F. Boyer, D. Ryczko, A. Ishiguro, and A. J. Ijspeert, "Emergence of robust self-organized undulatory swimming based on local hydrodynamic force sensing," *Science Robotics*, vol. 6, no. 57, 2021.

[39] R. S. Sutton and A. G. Barto, *Reinforcement learning - an introduction*, ser. Adaptive computation and machine learning. MIT Press, 1998.

[40] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen, "From swimming to walking with a salamander robot driven by a spinal cord model," *Science*, vol. 315, no. 5817, pp. 1416–1420, 2007.

[41] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[42] D. Owaki, M. Goda, S. Miyazawa, and A. Ishiguro, "A minimal model describing hexapedal interlimb coordination: the tegotae-based approach," *Frontiers in neurorobotics*, vol. 11, p. 29, 2017.

[43] O. Ekeberg and K. Pearson, "Computer simulation of stepping in the hind legs of the cat: an examination of mechanisms regulating the stance-to-swing transition," *Journal of neurophysiology*, vol. 94, no. 6, pp. 4256–4268, 2005.

[44] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[45] Unitree Robotics. A1. https://www.unitree.com/products/a1/.

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.