Bio-Inspired Rhythmic Locomotion for Quadruped Robots

Jiapeng Sheng[1], Yanyun Chen[1], Xing Fang[1], Wei Zhang[1], Ran Song[1], Yu Zheng[2], and Yibin Li[1]

***Abstract*— The mechanisms of locomotion in mammals have been extensively studied and inspire the related researches on designing the control architectures for the legged robots. Reinforcement learning (RL) is a promising approach allowing robots to automatically learn locomotion policies. However, careful reward-function adjustments are often required via trial-and-error until achieving a desired behavior, as RL policy behaviors are sensitive to the rewards. In this paper, we draw inspiration from the rhythmic locomotion behaviors of animals and propose a new control architecture by incorporating a rhythm generator to naturally stimulate periodic motor patterns, which actively participates in the timing of phase transitions in the robot step cycle. To speed up training, we use the joint position increments rather than the conventional joint positions as the outputs of the RL policy. During deployment, the rhythm generator can be reused for the state estimation of quadruped robots. We validate our method by realizing the full spectrum of quadruped locomotion in both simulated and real-world scenarios. Supplementary video demonstration is available at https://vsislab.github.io/rhyloc/.**

I. INTRODUCTION

Rhythmic motor patterns widely exist in the locomotion behaviors of humans and animals [1][2], such as walking, running, steering, etc. Flexibility to change motor patterns is crucial for the survival in harsh environments. Therefore, understanding the mechanism of driving diverse rhythmic motor patterns is an essential theme for biology and robotics.

The physiological findings suggested that the central pattern generators (CPGs) [3], the biological neural circuits in spinal cord, play critical roles in rhythm generation [4], which produce appropriate cadence to modulate the motoneuronal outputs [5]. The command signals from the midbrain locomotor region (MLR) [6] and the sensory inputs from the proprioceptive and exteroreceptive receptors[7] can alter rhythm patterns to adapt to different situations. Inspired by such findings, some researchers replicated rhythmic locomotion behaviors by designing simple phase oscillators to provide rhythm information for motor commands [8][9].The more sophisticated control methods like model predictive control (MPC) [10], and whole-body control (WBC) [11] can achieve better performance by introducing sensory feedback and complex control theories. Although a great progress has been made by such methods, rich expertise and laborious design processes are often required [12].

In recent years the model-free reinforcement learning (RL) [13] enables the autonomous design of the locomotion policies for the legged robots [12][14][15][16]. However, reward functions are generally underspecified for the desired rhythmic locomotion behaviors, or reasonable rewards must be carefully shaped to match the need, since small change in rewards may lead to massive variance of policy behaviors [17]. Thus, it is difficult and time-consuming to design good reward functions without bias [18].

In this work, we demonstrate how a rhythm generator can naturally stimulate period motor patterns within current RL frameworks. In a word, we use a rhythm generator (RG) network to adjust the timing of phase transitions between the swing phase and the stance phase, and a pattern formation (PF) network to output motor commands for each leg of the robot. The similar structure also exists in the mammalian nervous system [19][20], where the RG network determines the flexor and extensor phase durations, and the PF network generates cyclical activation of the flexor and extensor motoneurons. From an engineering perspective, we instantiate our method in a similar way by encouraging the robot to lift the corresponding foot high when the leg is in the swing phase or hold the corresponding foot firmly to the ground when the leg is in the stance phase. The cycle of the leg phases ensures the emergence of the animal-like rhythmic locomotion behaviors. With the proposed control architecture, we can focus on the main tasks of legged robots, such as forward movement and steering movement, etc.

It is worth mentioning that the accompanying leg phase estimation provided by the RG network can promote the correct body velocity estimation when deployed on the real robot platform. The current state estimation technique of the quadruped robots [21][22] fuses IMU measurements with leg kinematics to estimate full body states, which requires foot contact information. An alternate way is to use force sensors to detect foot contacts [12]. However, adding more sensors increases robot cost and power consumption, and reduces system robustness [23].

Instead of directly outputting joint positions like the prior works [12][23][24], our RL policy outputs joint position increments, which are added to the target joint position commands of the previous moment as the final motor commands. The new action space can speed up training , since the optional action space of the RL policy are narrowed around the joint positions of the current moment. We argue that the target joint position commands far from the joint positions of the current moment are harmful and meaningless due to the limitation of maximum motor speed.

We summarize our contributions as follows:

- We show that the proposed control architecture consisting of the RG and the PF networks play a similar role

[1]Jiapeng Sheng, Yanyun Chen, Xing Fang, Wei Zhang, Ran Song and Yibin Li are with School of Control Science and Engineering, Shandong University, 250061, Jinan, China.

[2]Yu Zheng is with Tencent Robotics X, Shenzhen, Guangdong Province, China.

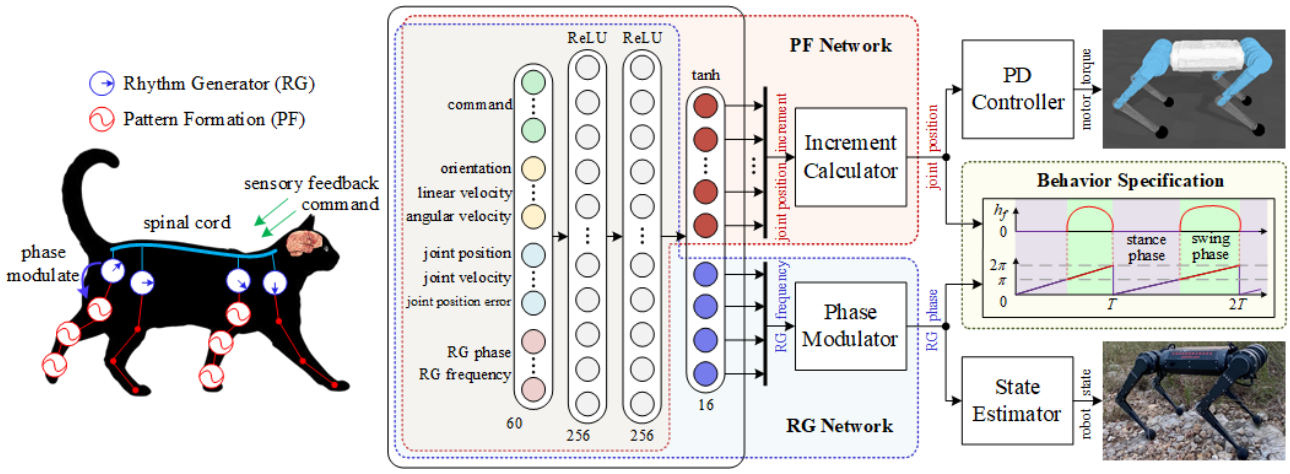

Fig. 1. **Left:** The hierarchical control architecture in the spinal cord strongly contributes the rhythmic locomotion behaviors of animals, which consists of the rhythm generator (RG) and the pattern formation (PF) networks. Both networks can be modulated by the command signals and the sensory feedback to generate different motor patterns. **Right:** The proposed control architecture also consists of the RG and the PF networks. The RG network modulates the leg phases alternating between the swing and the stance phases, and rewards the robots for lifting feet or holding feet firmly to the ground correspondingly, as shown in the "behavior specification" box. The PF network defined in the new action space of joint position increments outputs motor commands to control robots. During deployment, the RG phases are reused in the state estimator for the accurate estimation of the linear velocities of the base. The "increment caculator" and the "phase modulator" boxes refer to Eqs. (2) and (4) respectively.

as the one in mammals, ensuring the emergence of the desired rhythmic locomotion behaviors.

- We demonstrate that the proposed action space of joint position increments speeds up training.
- We demonstrate the agile locomotion behaviors in our small sized quadruped robot that performs coherent steering in forward and lateral movement, with a maximum speed of 3.8 m/s in the real world.

II. Related Work

We categorize the control systems for quadruped robots into two groups: model-based and learning-based controllers.

A. Model-Based Controllers

Model-based controllers for quadruped robots have been developed for years with mature theories and extensive practices. Fukuoka et al. [9] designed phase oscillators to generate foot trajectories. Bloesch et al. [25] precisely modeled the compliant actuation system to ensure good performance on hardware platform. Carlo et al. [10] proposed a new implementation of MPC with a simplified dynamic model to obtain robust controller. Based on [10], Bledt et al. [22] used state machine to generate reference trajectories, and performed MPC to plan ground reaction forces. However, these methods require a large amount of prior knowledge about robot structure and dynamics.

B. Learning-Based Controllers

Data-driven learning has become an effective alternative for automatically learning locomotion behaviors. Haarnoja et al. [26] applied an end-to-end RL framework to train robots to walk. Da et al. [27] proposed a hierarchical controller architecture, of which the high-level controller is trained by RL while the low-level controller provides predefined fixed gaits. This architecture utilizes the traditional control methods to accelerate learning, but limits the robot's athletic ability. Hwangbo et al. [23] successfully trained robots to walk and recover from falling after carefully designing reward functions. Siekmann et al. [17] defined a series of parametric reward functions to specify bipedal gaits, but requires lots of human priors. In contrast, our method only calls for basic rhythmicity without specifying any motor patterns which can flexibly respond to environmental changes.

III. Learning Rhythmic Locomotion

This work aims to naturally stimulate the rhythmic locomotion behaviors of the quadruped robots inspired by the biological locomotion mechanism, and speed up the RL learning procedures. The proposed learning framework can be divided into two components: a bio-inspired control architecture consisting of RG and PF networks, and a new action space of joint position increments. The schematic illustration of the mechanisms of rhythmic locomotion in mammals and our learning framework is shown in Fig. 1. The following sections will describe the whole learning framework in detail.

A. Problem Statement

We treat the locomotion problem of quadruped robots as a Markov decision process (MDP) $\langle S, A, R, P, \gamma \rangle$ [13], where S and A denote the state and the action spaces, respectively. $R(s_t, s_{t+1}) \to \mathbb{R}$ is the reward function, $P(s_{t+1} \mid s_t, a_t)$ is the transition probability and $\gamma \in (0,1)$ is the reward discount factor. An agent takes an action a at the current state s, receives a scalar reward r, and then transits to the next state s_{t+1} determined by the transition distribution $P(s_{t+1} \mid s_t, a_t)$. The agent intends to find an optimal policy π^*_Φ that maximizes the future discounted reward:

$$\Phi^* = \arg\max_{\Phi} \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t R(s_t, s_{t+1})\right]. \tag{1}$$

B. State-Action Space

The input state $s_t \in \mathbb{R}^{60}$ contains 3-dim commands $\hat{c}$ (including forward velocity $\hat{v}_x$, lateral velocity $\hat{v}_y$, and turning rate $\hat{\omega}_z$), 3-dim linear velocities of the base v, 3-dim angular velocities of the base ω, 3-dim orientation θ^g (represented as the direction of the gravity vector in the IMU frame), 12-dim joint positions q, 12-dim joint velocities $\dot{q}$, 12-dim joint position errors q_e (the joint positions q subtracted from the target joint positions $\hat{q}$), 8-dim RG phases ϕ and 4-dim RG frequencies f both provided by the RG network (the RG phases are represented by the sine and cosine functions).

The output action $a_t \in \mathbb{R}^{16}$ contains 4-dim RG frequencies f and 12-dim joint position increments Δq. Subsequently the target joint position commands $\hat{q}$ are computed according to the pipeline detailed in Sec. III-C. Finally, the PD controllers are used to compute joint torques $\tau = K_p(\hat{\mathbf{q}} - \mathbf{q}) + K_d(\hat{\dot{\mathbf{q}}} - \dot{\mathbf{q}})$, where K_p and K_d are fixed in both simulation and deployment, and the target joint velocities $\hat{\dot{\mathbf{q}}}$ are set to 0.

C. Action Space of Joint Position Increments

Our RL policy outputs the joint position increments Δq_t, which are then added to the target joint positions of the previous moment $\hat{q}_{t-1}$. Thus, the target joint positions of the current moment $\hat{q}_t$ are defined as

$$\hat{q}_t = \hat{q}_{t-1} + \Delta q_t. \tag{2}$$

The intuition behind is that the given target joint position commands will not exceed the operating performance of the motor. Therefore, in practice the maximum joint position increment Δq_{max} is determined by the maximum motor speed $\dot{q}_{max}$ and the duration of one time step T, defined as

$$\Delta q_{max} = \dot{q}_{max} * T. \tag{3}$$

D. Roles of RG and PF Networks

The RG network outputs frequency f to modulate the phase ϕ of each leg, defined as

$$\phi_t = (\phi_{t-1} + 2\pi * f * T) \% 2\pi, \tag{4}$$

where $\phi \in [0, 2\pi)$ represents that a leg is in the stance phase if $\phi \in [0, \pi)$ or the swing phase if $\phi \in [\pi, 2\pi)$. When a leg is in the swing phase, we reward the robot for lifting the corresponding foot high, or when in the stance phase, we reward the robot for holding the corresponding foot firmly to the ground. The detailed reward functions are described in Sec. III-E. As the RG frequencies f are non-negative, the step cycles of quadruped robots must alternate between the swing phase and the stance phase, which ensures the emergence of the rhythmic locomotion behaviors. During the real-world deployment, we use the RG phases to estimate foot contacts which is essential for the state estimator to obtain the accurate linear velocity of the base [21][22].

The PF network is similar to the RL policy used in the prior works [12][24][23], which takes as input the robot state to output motor commands. However, in our work it is defined in the action space of joint position increments and its generated locomotion behaviors are largely modulated by the RG network.

E. Reward Functions

The reward functions encourage the robot to follow commands, maintain balance and exhibit rhythmic movements. Here we keep the same meanings of the notations introduced in the Sec. III-B and Sec. III-C. For other notations, we denote the target linear velocities of the base as $\hat{v}$, the target angular velocities of the base as $\hat{\omega}$, the base orientation (representing the roll-pitch-yaw angles of the base) as θ, the joint positions of the standing posture as q_{ref}, the foot velocities as v_f, the distance between foot and ground as h_f, the binary foot contact indicator provided by the RG network as I_f, the real binary foot contact indicator provided by the physics simulator as $\hat{I}_f$, the raw RL policy outputs as o, the foot quantity in the swing phase as $\cdot_{swing}$, the foot quantity in the stance phase as $\cdot_{stance}$, l_1 norm as $|\cdot|$ and l_2 norm as $\|\cdot\|$. For simplicity, we also denote the command coefficients shared among the reward terms as $c_x = 1/|\hat{v}_x|$, $c_y = 1/|\hat{v}_y|$ and $c_w = 1/|\hat{\omega}_z|$.

The reward r_t at each time step is defined as the sum of the following terms:

1) forward velocity: $2\exp\left(-3c_x * (v_x^t - \hat{v}_x^t)^2\right)$;
2) lateral velocity: $2\exp\left(-3c_y * \left(v_y^t - \hat{v}_y^t\right)^2\right)$;
3) angular velocity: $1.5\exp\left(-1.5c_w * \left(\omega_z^t - \hat{\omega}_z^t\right)^2\right)$;
4) balance: $1.3(\exp\left(-2.5c_x * |v_z^t|^2\right) + \exp\left(-2c_x * \|\omega_{xy}^t\|^2\right))$;
5) twist: $-0.6c_x * \|\theta_{xy}^t\|$;
6) foot slip: $-0.07c_x * \left\|v_{f,stance,xy}^t\right\|$;
7) foot stance: $\sum_{i=1}^4 \left(h_{f,stance,i}^t < 0.01\right)$;
8) foot clear: $0.7c_1 * \sum_{i=1}^4 \left(h_{swing,i}^t > 0.01\right)$;
9) foot z velocity: $-0.03c_x * |v_{f,z}^t|^2$;
10) joint constraints: $-0.8c_x * \left\|\mathbf{q}^t - \mathbf{q}_{ref}^t\right\|^2$;
11) joint torque: $-0.0012c_2 * c_x * |\tau^t|$;
12) joint velocity: $-0.0008c_3 * c_x * \|\dot{\theta}^t\|^2$;
13) policy output smooth: $-0.016c_4 * c_x * \|o^t - o^{t-1}\|$;
14) RG frequency: $-0.03c_x * |f_{stance}^t|$;
15) RG phase: $\sum_{i=1}^4 \left(I_{f,i}^t = \hat{I}_{f,i}^t\right)$;

where c_1-c_4 are determined by the different experimental configurations detailed in Sec IV.

Except terms 14-15 are new for the proposed control architecture consisting of RG and PF networks, all the reward terms have been used in the prior works [12][23][24]. Items 1-5 enable the robot to follow the commands and maintain balance. Items 6-8 are the key to stimulating period motor patterns by rewarding the robot for lifting foot or holding foot firmly to the ground based on the different leg phases. It is worth noting that the leg phases are the ground truth of the foot contacts in the prior works while in our work, the leg phases are provided by the RG network. In other words, the leg phases in our work can be actively modulated by the RL policy to generate the rhythmic locomotion behaviors. Items 9-13 encourage smooth and efficient motions. Item 14 expects for a large proportion of stance

TABLE I

LOWER AND UPPER BOUNDS OF RANDOMIZED PHYSICAL PARAMETERS AND SENSOR NOISES

parameters	lower bound	upper bound
mass	80%	120%
inertia	50%	150%
latency	16 ms	28 ms
motor strength	80%	120%
terrain friction	0.2	1
linear velocity noise	-0.08 m/s	0.08 m/s
angular velocity noise	-0.3 rad/s	0.3 rad/s
orientation noise	-0.08 rad	0.08 rad
joint position noise	-0.01 rad	0.01 rad
joint position noise	-1.2 rad/s	1.2 rad/s

phase. Item 15 is used to narrow the difference between the estimated foot contacts provided by the RG network and the real foot contacts provided by the physics simulator, which is useful in the state estimator during deployment.

F. Supplementary Details

1) Curriculum Learning: The curriculum learning method [28] is introduced to allow the robot to preferentially learn the main task (following the commands and maintaining the balance) and prevent the tendency to stand in place [23]. The training process starts with a small multiplicative curriculum factor $k_c \in [0,1]$ attached to the reward terms 4-15, and then k_c increases gradually to satisfy the other criteria. k_c^t is defined as $k_c^t = (k_c^{t-1})^{k_d}$, where $k_d \in [0,1]$ indicates how fast k_c^t reaches 1.

2) Sim to Real Transfer: We use domain randomization [14][29][30] to overcome the reality gap for the real-world deployment, which enables the agent to learn a robust policy by varying physical parameters and adding sensor noises. The lower and upper bounds of randomized physical parameters and sensor noises are summarized in Table I. All the parameters and noises are uniformly sampled.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we experimentally demonstrate that 1) the control architecture consisting of the RG and the PF networks naturally stimulate the rhythmic motor patterns and 2) the action space of joint position increments significantly speeds up training. To expand the application scenarios of our method, we further demonstrate the full-spectrum movement on multiple types of terrains. Supplementary video demonstration is available at https://vsislab.github.io/rhyloc/.

A. Experimental Setups

1) Training Setup: We use the URDF exported from the software SolidWorks to simulate our robot in PyBullet [31]. The control frequency of the RL policy is 100 Hz and the simulation frequency is 2000 Hz. We simulate each independent training for a maximum of 3000 iterations and terminate early if the robot body collides with the ground. We use PPO [32] to train RL policy, a 3-layer MLP (256, 256 hidden layer sizes, as shown in Fig. 1) and its implementation code is modified based on the open-source reinforcement learning platform Tianshou [33]. We use the following experimental configurations:

TABLE II

PPO HYPER-PARAMETERS

parameters	value
batch size	$36728(1024 \times 32)$
mini-batch size	$8192(1024 \times 8)$
number of epochs	3000
clip range	0.2
entropy efficient	0.0025
learning rate	$3.5e-4$
discount factor	0.992
GAE discount factor	0.95
desired KL-divergence	0.015

1) POS: It is the conventional configuration in the prior works [12][34][23] which uses the pure joint positions as the RL policy outputs;
2) INC: It is similar to the above configuration but defined in the action space of joint position increments;
3) RHY+POS: It uses the proposed control architecture consisting of RG and PF networks but is defined in the conventional action space of joint positions;
4) RHY+INC (Ours): It is the configuration of our method which uses the proposed control architecture and is defined in the action space of joint position increments.

All experimental configurations are setting with the same command range ($\hat{v}_x \in (0.3, 2)$ m/s, $\hat{v}_y \in (-1.0, 1.0)$ m/s, $\hat{\omega}_z \in (-1.0, 1.0)$ rad/s) on the flat terrains and the hill terrains (roughness, frequency, amplitude are 0.3, 0.1, 0.05 respectively, similar to [35]) simultaneously. An external force uniformly sampled from the range of (20, 50) N are applied laterally to the robot for 0.2s at the intervals of 3s. We set the random seed to 0 and use the same forms of the reward terms, but some reward weights are separately adjusted until arriving at a desired behavior. The reward weights c_1-c_4 in Sec. III-E for each experimental configuration are POS: 0.6, 1.2, 1.2, 7.5; INC: 0.7, 1.0, 1.0, 1.0; RHY+POS: 0.8, 1.2, 1.2, 7.5; INC+RHY: 1.0, 1.0, 1.0, 1.0, respectively. And the hyper-parameters of PPO are listed in Table II. We use such fine-tuned parameters as the default setting of each experimental configuration.

2) Real-World Deployment Setup: We validate our method using a quadruped robot built in a way similar to the open-source Mini Cheetah robot [36], with 12 actuated DoFs and 6 underactuated DoFs. The robot wights about 10.5 kg, with the maximum motor speed of 10.6 rad/s and and the maximum torque of 18 Nm. One IMU sensor and 12 motor encoders are mounted on the robot to estimate the full state, including the orientation, the linear velocities, angular velocities of the base as well as the positions and the velocities of the joints. The deployed policy is then converted into ONNX format and the OpenVINO toolkit is used as an inference engine on the onboard computer. The predicted target joint positions are converted into the joint torques using a PD controller ($K_p = 78$, $K_d = 2$) with the frequency of 40 kHz to control motors.

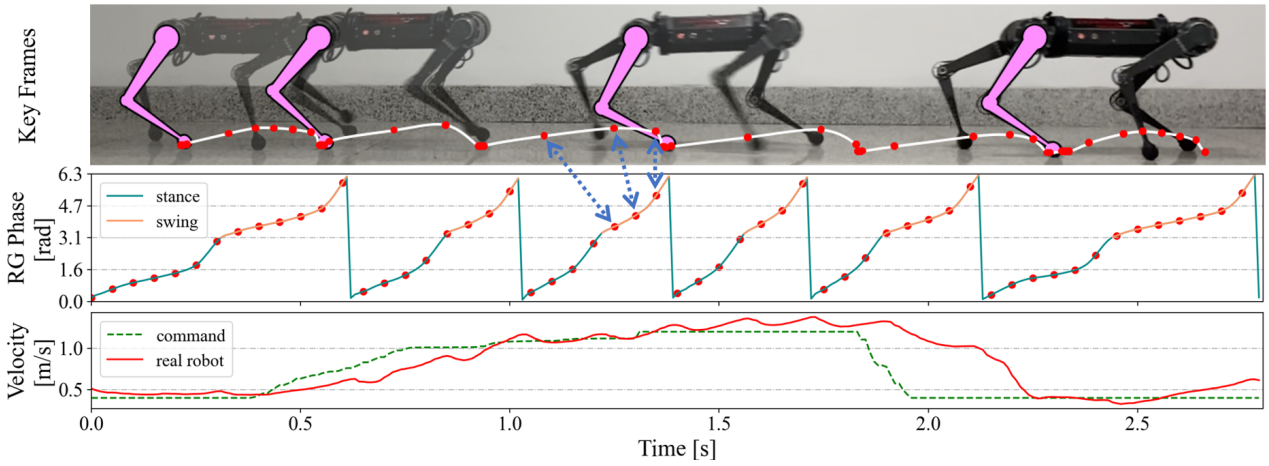

Fig. 2. **Top**: The key frames of forward movement and corresponding foot trajectory of the right rear foot. **Middle**: The timing of phase transitions of the right rear foot estimated by the RG network. **Bottom**: The accuracy of the forward velocity tracking with our method during deployment. The red points in the foot trajectory and the RG phase are one-to-one correspondence (the dotted blue arrows), which are both uniformly sampled at intervals of 50ms with the same start time, indicating that the RG network can actively modulate rhythms to alter the motor patterns.

B. Effect of RG and PF Networks

In order to verify the effectiveness of the RG and the PF networks in stimulating rhythmic movements, we conduct two experiments: 1) add the weight noises to the default value of the reward terms 6-8 uniformly sampled from the range of (-0.3, 0.3), and 2) set different random seeds while keeping the other settings as default. We train 10 separate policies for each configuration in each experiment (80 in total), and the resulting number of successful and failure cases are summarised in Fig. 3A and Fig. 3B, where we classify the two obvious unnatural gaits like the cases shown in the top of Fig. 3 as failures.

We observe that the POS and INC baselines are sensitive to the reward weights and the random seeds as a small change leads to massive variance in policy behaviors, while the experimental configurations with RHY stably obtain the period motor patterns without any failure. We argue that in the conventional RL configurations (POS, INC), the shaped rewards are often underspecified and cause bias learning. In our work, the proposed control architecture consisting of RG and PF networks fully specifies the rhythmic locomotion behaviors by ensuring the cycle of leg phases alternating between the swing and the stance phases.

We further transfer the trained policy with our method to the real world to reveal the intrinsic properties of the proposed control architecture. As shown in Fig. 2, we find that the stride length and frequency of the robot increase accordingly as the linear velocity command of the base increases, demonstrating that the RG and the PF networks can actively respond to the command signals to alter motor rhythms and patterns. It is also noteworthy that the red points in the plotted foot trajectory and the RG phase of the right rear foot are both uniformly sampled at intervals of 50ms with the same start time, which visualizes that the true phase of the leg at each moment almost matches the RG phase estimated by the RG network (dotted blue arrows), confirming that the RG network can actively modulate rhythms to alter motor patterns. The perfect match also produces a byproduct

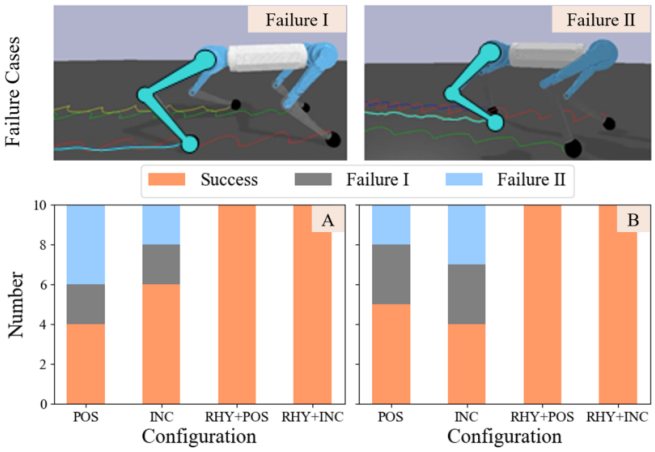

Fig. 3. **Top**: The key frames of two failure cases: 1) One leg dragging on the ground (**Failure I**); 2) One leg hanging in the air (**Failure I**). **Bottom**: The number of successful and failure cases in the two experiments: 1) adding weight noises to the default value (**A**); 2) setting different random seeds (**B**).

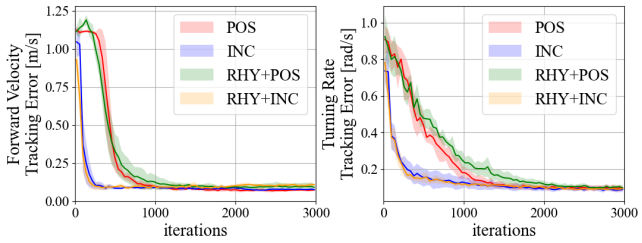

Fig. 4. The resulting average learning speed curves measured with the tracking errors of forward velocity (**left**) and turning rate (**right**) commands.

that the RG phases can be used to detect foot contacts which contributes to the accurate linear velocity estimation during deployment.

C. Effect of Action Space of Joint Position Increments

We compare the learning speed defined in the new action space of joint position increments against the conventional action space of joint positions by training separate policies for each experiment configuration with 5 different random seeds while keeping other settings as default. We keep 5 successful training processes for each experimental con-

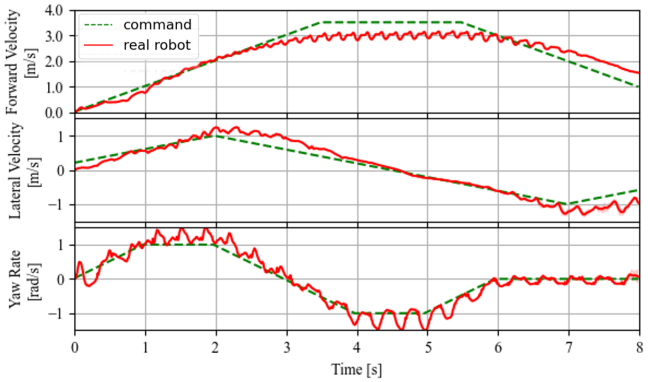

Fig. 5. The accuracy of command tracking with our method, demonstrating a well-performed full-spectrum movement.

figuration and abandon other failure cases shown in the top of Fig. 3. Due to the difference in reward weights of each experimental configuration, we use the tracking errors of forward velocity and turning rate commands during the training process to measure the learning speed. The tracking errors are defined as $e_c = \frac{1}{N}\sum_{t=0}^{N}|\hat{c}^t - c^t|$, where c is v_x or ω_z. We save the trained model and compute the command tracking errors every 50 training iterations. The corresponding learning curves are shown in Fig. 4.

We find the experimental configurations with INC can greatly speed up training compared to the configurations with POS. We argue that the proposed action space of joint position increments narrows the target joint position commands of the current moment $\hat{q}_t$ around the joint positions of the current moment q_t, which is more reasonable than the conventional action space of joint positions. Note that our method (RHY+INC) achieves similar learning speed as INC, which shows the extra 4-dim actions introduced by the proposed control architecture do not slow down training.

D. Performance Analysis of Full-Spectrum Movement

We evaluate our method via full-spectrum movement including steering, forward and lateral movement through the real-world deployment. We test each command five times with one trained policy while the other two commands are set to 0. As shown in Fig. 5 we observe that the trained policy achieves accurate command tracking and the mean tracking errors are 0.3 m/s, 0.15 m/s and 0.16 rad/s, respectively.

We use the RG phases to estimate foot contacts which can be used in state estimator to measure the linear velocities of the base. As shown in Fig. 6, when the forward velocity command reaches the maximum value of 4m/s, the velocity estimation using the RG phases (about 3.3 m/s) is more accurate than the one without using the RG phases (about 1.9m/s) while the true average forward velocity is about 3.8 m/s captured by a camera. We observe a flying trot gait when the robot runs at a high speed (see the supplementary video for details), which breaks the basic assumption of state estimator that at least one foot is in contact with the ground causing the gap between the estimated value and the real one. Note that the true average forward velocity of 3.8 m/s is quite fast for our small sized quadruped robot as it is

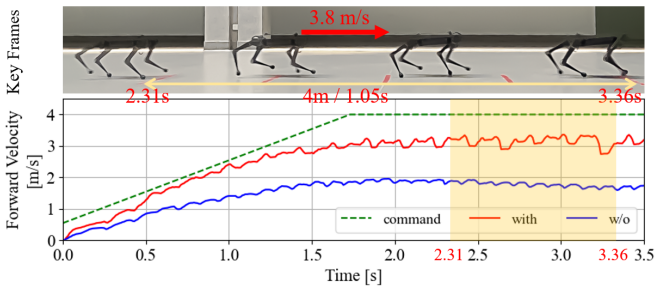

Fig. 6. Ablation studies of using RG phases in state estimator during deployment. We can obtain more accurate linear velocity estimation of the base compared to the one without using the RG phases which only relies on IMU.

slightly larger than the current maximum speed of 3.7 m/s implemented by MPC controller in [11].

E. Robustness in Response to Environmental Disturbances

In order to demonstrate the robustness of our method in complex environments, we drag the robot with a rope until it deviates significantly from the direction of motion as shown in the left of Fig. 7. We observe that when the external force is applied, the forward velocity of the robot fluctuates greatly and the RG network outputs higher frequencies during the swing phase (dotted blue circle) to increase stride frequency and the proportion of swing phase accordingly to maintain balance. And when the environmental disturbances disappear, the robot tracks the command normally again.

We further show the key frames of 4 examples in complex environments to expand the application scenarios of our method. As shown in the right of Fig. 7, the trained policies with our method can adapt to different situations like icy ground and construction site, etc.

V. CONCLUSIONS

In this paper, we introduce a bio-inspired control architecture consisting of RG and PF networks, which naturally stimulates rhythmic locomotion behaviors. This architecture only calls for basic rhythmicity without artificial constraints to the specified motor patterns. We also propose a new action space of joint position increments to speed up RL training. Our method shows fast learning speed and well-performed periodic motor patterns during training. We also deploy the trained RL policy in the real world and demonstrate the full-spectrum movements in complex environments.

REFERENCES

[1] S. Grillner, "Neurobiological bases of rhythmic motor acts in vertebrates," *Science*, vol. 228, no. 4696, pp. 143–149, 1985.
[2] E. Marder and R. L. Calabrese, "Principles of rhythmic motor pattern generation," *Physiological reviews*, vol. 76, no. 3, pp. 687–717, 1996.
[3] S. L. Hooper, "Central pattern generators," *Current Biology*, vol. 10, no. 5, pp. R176–R179, 2000.
[4] N. P. Cramer, Y. Li, and A. Keller, "The whisking rhythm generator: a novel mammalian network for the generation of movement," *Journal of neurophysiology*, vol. 97, no. 3, pp. 2148–2158, 2007.
[5] A. Prochazka and P. Ellaway, "Sensory systems in the control of movement," *Comprehensive Physiology*, vol. 2, no. 4, pp. 2615–2627, 2012.
[6] M. L. Shik, "Control of walking and running by means of electrical stimulation of the midbrain," *Biophysics*, vol. 11, pp. 659–666, 1966.

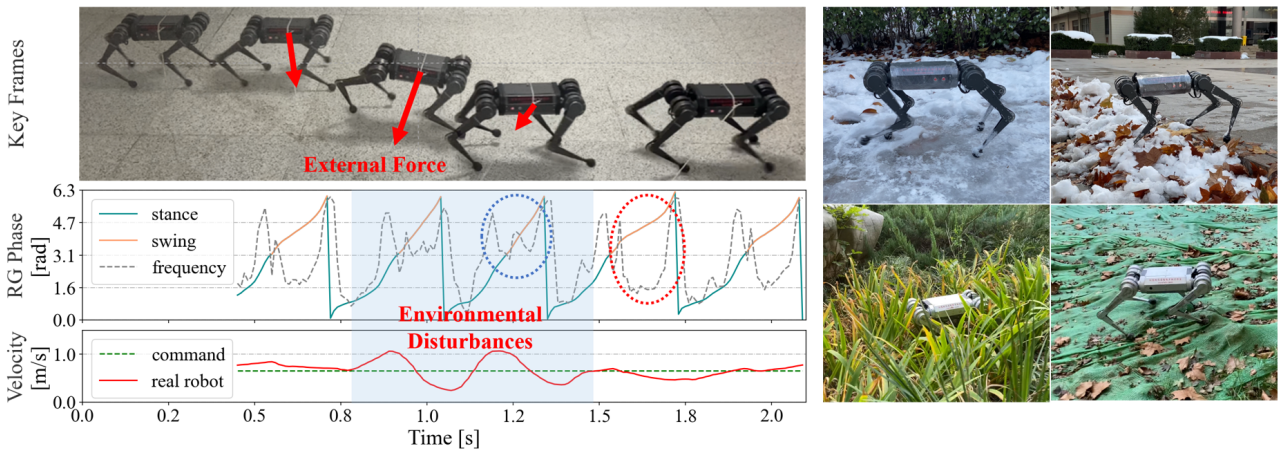

Fig. 7. **Left**: Response to external force. The robot managed to actively modulate its rhythm by increasing the stride frequency and the proportion of swing phase to maintain the balance when dragged by a rope. **Right**: Deployment of the trained policy with our approach in 4 outdoor environments: icy ground, snowdrift, underbrush and construction site.

[7] J. S. Gottschall and T. R. Nichols, "Head pitch affects muscle activity in the decerebrate cat hindlimb during walking," *Experimental brain research*, vol. 182, no. 1, pp. 131–135, 2007.

[8] S. Venkataraman, "A simple legged locomotion gait model," *Robotics and Autonomous Systems*, vol. 22, no. 1, pp. 75–85, 1997.

[9] Y. Fukuoka, H. Kimura, Y. Hada, and K. Takase, "Adaptive dynamic walking of a quadruped robot'tekken'on irregular terrain using a neural system model," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 2037–2042.

[10] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[11] 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.

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

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[14] 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," *arXiv preprint arXiv:1804.10332*, 2018.

[15] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," *arXiv preprint arXiv:1901.07517*, 2019.

[16] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Conference on Robot Learning*. PMLR, 2020, pp. 317–329.

[17] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-real learning of all common bipedal gaits via periodic reward composition," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7309–7315.

[18] A. Irpan, "Deep reinforcement learning doesn't work yet," https://www.alexirpan.com/2018/02/14/rl-hard.html, 2018.

[19] R. Burke, A. Degtyarenko, and E. Simon, "Patterns of locomotor drive to motoneurons and last-order interneurons: clues to the structure of the cpg," *Journal of neurophysiology*, vol. 86, no. 1, pp. 447–462, 2001.

[20] M. Lafreniere-Roula and D. A. McCrea, "Deletions of rhythmic motoneuron activity during fictive locomotion and scratch provide clues to the organization of the mammalian central pattern generator," *Journal of neurophysiology*, vol. 94, no. 2, pp. 1120–1132, 2005.

[21] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots-consistent fusion of leg kinematics and imu," *Robotics*, vol. 17, pp. 17–24, 2013.

[22] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2245–2252.

[23] 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, p. eaau5872, 2019.

[24] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?" in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2017, pp. 1–13.

[25] C. Gehring, S. Coros, M. Hutter, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger *et al.*, "Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot," *IEEE Robotics & Automation Magazine*, vol. 23, no. 1, pp. 34–43, 2016.

[26] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," *arXiv preprint arXiv:1812.11103*, 2018.

[27] X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich, and A. Garg, "Learning a contact-adaptive controller for robust, efficient legged locomotion," *arXiv preprint arXiv:2009.10019*, 2020.

[28] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[29] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "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)*. IEEE, 2017, pp. 23–30.

[30] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[31] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[33] J. Weng, H. Chen, D. Yan, K. You, A. Duburcq, M. Zhang, H. Su, and J. Zhu, "Tianshou: A highly modularized deep reinforcement learning library," *arXiv preprint arXiv:2107.14171*, 2021.

[34] Z. Fu, A. Kumar, J. Malik, and D. Pathak, "Minimizing energy consumption leads to the emergence of gaits in legged robots," *arXiv preprint arXiv:2111.01674*, 2021.

[35] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[36] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6295–6301.