J. Shanghai Jiao Tong Univ. (Sci.), 2023
https://doi.org/10.1007/s12204-023-2666-z

Gait Learning Reproduction for Quadruped Robots Based on Experience Evolution Proximal Policy Optimization

LI Chunyang (李春阳), *ZHU Xiaoqing** (朱晓庆), *RUAN Xiaogang* (阮晓钢),
LIU Xinyuan (刘鑫源), *ZHANG Siyuan* (张思远)
(Faculty of Information Technology, Beijing University of Technology; Beijing Key Laboratory of Computational Intelligence and Intelligent System; Engineering Research Center of Digital Community of Ministry of Education, Beijing 100124, China)

Abstract: Bionic gait learning of quadruped robots based on reinforcement learning has become a hot research topic. The proximal policy optimization (PPO) algorithm has a low probability of learning a successful gait from scratch due to problems such as reward sparsity. To solve the problem, we propose a experience evolution proximal policy optimization (EEPPO) algorithm which integrates PPO with priori knowledge highlighting by evolutionary strategy. We use the successful trained samples as priori knowledge to guide the learning direction in order to increase the success probability of the learning algorithm. To verify the effectiveness of the proposed EEPPO algorithm, we have conducted simulation experiments of the quadruped robot gait learning task on Pybullet. Experimental results show that the central pattern generator based radial basis function (CPG-RBF) network and the policy network are simultaneously updated to achieve the quadruped robot's bionic diagonal trot gait learning task using key information such as the robot's speed, posture and joints information. Experimental comparison results with the traditional soft actor-critic (SAC) algorithm validate the superiority of the proposed EEPPO algorithm, which can learn a more stable diagonal trot gait in flat terrain.
Key words: quadruped robot, proximal policy optimization (PPO), priori knowledge, evolutionary strategy, bionic gait learning
CLC number: TP 242 **Document code:** A

0 Introduction

The control of quadruped robots, being a highly nonlinear structure, requires the kinematics and dynamics of the robots, as well as the modeling of their control model. Since the first quadruped robot was born in 1960[1], bionic quadruped robots have been an interesting research topic for scholars. This type of robot is a dynamic system with complex control features that is strongly coupled and has numerous inputs and outputs. Owing to the intricate nature of the situation, conventional control approaches that allow for the examination of the environment and the development of pertinent mathematical models cannot encompass all eventualities[2]. Therefore, deep reinforcement learning (DRL) algorithms[3-5] can be used to tackle the complexity of traditional control techniques, enabling the robot to learn locomotion skills through many interactions with the environment via trial and error, just like animals in nature.

Thor et al.[6] developed a legged robot motion control system utilizing a central pattern generator (CPG). The approach was appraised by its successful application to three different types of legged robots, yet fell short of providing the desired degree of interaction with the environment. Subsequently, Peng et al.[7-8] introduced DeepMimic, which makes use of reinforcement learning methods for the robotic acrobatics learning process, necessitating expert knowledge of both the robotics system and the desired locomotion skill. Lee et al.[5] then implemented the use of proprioceptive information from the robot to achieve a stable walking of ANYmal robots by means of DRL approaches, necessitating large computational resources. Rahme et al.[9] further adjusted the robot's gait by way of Bezier curves as a low-order open loop model combined with the augmented random search method. Tan et al.[10] adopted two sinusoidal functions and proximal policy optimization (PPO) to construct the foot trajectory for the support of the quadruped robot's reinforcement learning. However, the gait learning process can be too rigid due to the priori knowledge provided by the fixed trajectory.

Received: 2023-02-20 **Accepted:** 2023-03-14
Foundation item: the National Natural Science Foundation of China (No. 62103009)
***E-mail:** alex.zhuxq@bjut.edu.cn

To address the aforementioned issues, this paper proposes experience evolution proximal policy optimization (EEPPO) algorithm, a gait learning algorithm for quadruped robots which integrates PPO with priori knowledge comprising evolutionary guidance to enable quadruped robots to achieve an effective bionic diagonal trot gait learned from scratch. Following prior work in Refs. [5, 9-10], the construction of the controller in this research consists of two components: a priori knowledge provided by a foot trajectory generator (FTG) and a residual control signal generated by a neural network (NN)-based policy, both of which are subsequently combined to generate the signal to control the motor. The FTG in this research utilizes a central pattern generator based radial basis function (CPG-RBF) network[6] and leverages the evolution strategy (ES)[11] to seek for the optimal trajectory in trajectory space and optimize the linear network parameters. As opposed to previous work, the provided priori knowledge is not a fixed trajectory, and thus allows the robot's foot-end trajectory to be adapted to the environment and the robot itself during training, which leads to better matching with the quadruped's own motion characteristics, contributes to a more robust update direction of the DRL policy network, and ultimately provides better quality training models. The update of the policy network follows the PPO algorithm, a stable on-policy learning method capable of demonstrating excellent performance in controlling the tasks with continuous episodes[12].

In this paper, the effectiveness of the EEPPO algorithm is evaluated through its application in a quadruped robot learning a forward trot gait from scratch. The algorithm combines prior knowledge optimization with PPO and leverages the reward function of the environment to motivate the robot towards faster and more energy-efficient paths. The results obtained by experiments show that the proposed approach outperforms prior methods in terms of stability, coordination, and flexibility of the robot's movement. Finally, we demonstrate that the robot can learn stable forward bionic gait strategies while interacting with its environment on flat terrain.

1 Deep Reinforcement Learning for Quadruped Robot Gait Learning

1.1 Problem Description

In this section, we formulate the task of controlling quadrupedal robots locomotion as a Markov decision process (MDP) and solve it using DRL. The MDP is defined by a tuple $(S, A, P(s_{t+1}|s_t, a_t), R, \gamma)$[13-14], where S is the finite state space, A is the finite action space, $P(s_{t+1}|s_t, a_t)$ is a probabilistic transition function, s_t is the state of the agent at the current moment, a_t is the action of the agent at the current moment, s_{t+1} is the state of the agent at the next moment, R is the reward function, and $\gamma \in [0, 1]$ is the discount factor that can be used to ensure that the return is finite. Here, r_t indicates the expected reward obtained by the agent at time t after taking the action a_t. The goal of the controller based on reinforcement learning is to select an optimal policy π^*, so that the robot can obtain the maximum expected average reward during the movement process:

$$\pi^* = \arg\max_{\pi} E_{s_{t+1}\sim P}\left(\sum_{t=0}^{\infty}\gamma^t r_t\right). \tag{1}$$

In this paper, we have adopted an actor-critic reinforcement learning framework[15] to identify the optimal policy, wherein the state value function $V_\theta(s_t)$ is used to estimate the expected future reward:

$$V_\theta(s_t) = E_{a_t\sim\pi(s_t)}\left(\sum_{t=0}^{\infty}\gamma^t r_t\right). \tag{2}$$

The state value function can be approximated by minimizing the temporal difference (TD) error, which serves as the loss function of the critic network:

$$\mathcal{L}(V_\theta) = [V_\theta(s_t) - (r_t + V_\theta(s_{t+1}))]^2. \tag{3}$$

We use the neural network θ to represent the policy $\pi_\theta(s_t)$. The optimal policy can be achieved by maximizing the expected reward, with the loss function of the actor network represented as the negative state value function. Consequently, we are able to identify the best policy for our problem.

$$\mathcal{L}(\pi_\theta) = -V_\theta(s_t). \tag{4}$$

1.2 State and Action Space

State space is $s_t \in \mathbb{R}^{49}$, which contains 49-dimensional data. The state s_t includes the roll angle, pitch angle, yaw angle and angular velocity of the robot body obtained from the inertial measurement unit (IMU), the four binary values representing whether the foot joints of the robot are supported on the ground or suspended in the air, the angle and angular velocity of each motor joint of the robot, and the speed of the robot body in the world coordinate system. The distinction between our study and Refs. [5, 9-10] is that we add the FTG signal to the state space, which is a 12-dimensional motor control signal. In this way, various observations may be appropriately utilized to capture the status of the quadruped robot in the environment, thereby learning the most suitable gait strategy for the robot given its real-time state.

Action space is $a_t \in \mathbb{R}^{12}$, which contains 12-dimensional data and consists of the angles of 12 motors of the robot, tracked by the joint-level PD controller in order to reach the desired angle.

1.3 Reward Function

In this study, we have designed a reward function to enable our model to learn a stable and energy-efficient bionic trot gait. The reward function consists of three distinct components:

(1) The speed reward, including two speed rewards, is to encourage the forward movement of the quadruped robot, which can be formulated as

$$r_{\mathrm{v}} = w_1 v_{\mathrm{base}} + w_2 \sum_{f=1}^{4} v_{\mathrm{feet},f}, \tag{5}$$

where $w_1 = 1.0$, $w_2 = 0.2$, v_{base} is the velocity of the robot's center of mass, and $v_{\mathrm{feet},f}$ is the velocity of the four foot-end joints of the robot.

(2) The stability reward, containing three penalties that cause the robot to be unstable, enables the quadruped robot to move smoothly, which can be expressed as follows:

$$\begin{aligned} r_{\mathrm{s}} = &- w_3 \sqrt{\theta_{\mathrm{roll}}^2 + \theta_{\mathrm{pitch}}^2} - w_4 n_{\mathrm{collision}} - \\ &w_5 \max(n_{\mathrm{loss\_contact}} - 2, 0), \end{aligned} \tag{6}$$

where $w_3 = 0.3$, $w_4 = 0.1$, $w_5 = 0.1$; θ_{roll} and θ_{pitch} represent the robot's roll and pitch angles, respectively; $n_{\mathrm{collision}}$ is the number of torso joints that touch the ground; $n_{\mathrm{loss\_contact}}$ is the number of foot-end joints not on the ground. The first item is to penalize the oscillations of two attitude angles of the robot, and thus prevent excessive displacements of its center of mass. The second item is to penalize the case in which the robot's trunk comes into contact with the ground. The third item is to penalize the robot when $n_{\mathrm{loss\_contact}}$ is greater than two joints which indicate the robot is severely unstable.

(3) The energy reward, which has only one penalty term, is used to penalize the robot for consuming excessive energy during movements:

$$r_{\mathrm{e}} = -w_6 \left|\tau q\right| \mathrm{d}t, \tag{7}$$

where $w_6 = 0.1$, τ is the motor torque, and q is the motor velocity.

The total reward can be obtained by

$$r = r_{\mathrm{v}} + r_{\mathrm{s}} + r_{\mathrm{e}}. \tag{8}$$

1.4 Foot Trajectory Generator Including Evolutionary Strategy

We employed a CPG-RBF neural network to generate foot trajectories[6], which provide prior knowledge of the motion to the policy network as a reference trajectory for policy learning. The CPG-RBF network, which combines a CPG with an RBF network, consists of an input layer, a hidden layer and a linear layer[16]. The input layer utilizes two sinusoidal signals to represent the rhythmic signal outputted from the CPG, which are defined as follows:

$$\left.\begin{aligned} o_0(t) &= A\sin(\omega t) \\ o_1(t) &= A\sin(\omega t + B) \end{aligned}\right\}, \tag{9}$$

where A is the amplitude, ω is the frequency, and B is the phase between two signals.

The hidden layer of the CPG-RBF neural network is an artificial neural network that utilizes the RBF as its activation function, thus providing flexibility in the construction of various foot trajectories by altering the parameters in the RBF network. The transfer function for the hidden layer is specified as follows:

$$\begin{aligned} L_m = &\\ &\exp\Big[-\frac{(o_0(t)-\mu_{m,0})^2 + (o_1(t)-\mu_{m,1})^2}{\sigma_{\mathrm{RBF}}^2}\Big], \\ &m = 1, 2, \cdots, H-1, \end{aligned} \tag{10}$$

$$\mu_{m,n} = o_n\Big(\frac{(i-1)T}{H-1}\Big), \quad n = 0, 1, \tag{11}$$

where L_m is the output of the RBF neuron after receiving the CPG signal, $\mu_{m,0}$ and $\mu_{m,1}$ are the two mean values of RBF neuron calculated by Eq. (11), σ_{RBF}^2 is the variance between neuron outputs, and H is the number of neurons in the hidden layer.

The output layer consists of a fully connected layer, which is used to calculate the target angles of each joint of the robot based on the result of

$$K = WL_m + b, \quad m = 1, 2, \cdots, H-1, \tag{12}$$

where W and b are respectively the weights and biases of the linear layer, and K includes 12 target angles.

We employ the CPG-RBF to optimize the shape of the foot-end trajectories, and an RBF network to generate diverse foot trajectories. Because of the low-dimensional structure of the trajectory space and because sampling in the parameter space may lead to trajectories that deviate significantly from the current trajectory[14], an evolutionary strategy is then employed to search for the optimal trajectory in the trajectory space. We use τ_0 to denote the ideal optimal trajectory, while $K(\tau_0)$ denotes the collection of control points as the trajectory space. In each iteration of the optimization process, we add Gaussian noise to $K(\tau_0)$ and sample 50 foot trajectories in trajectory space, the control points of which are solved for joint angles by inverse kinematics. The trajectory with the highest reward value is selected as the optimal foot trajectory and the parameters of the RBF network are updated following Ref. [17]. As a result, FTG constantly optimizes the parameters of the CPG-RBF network to generate optimal prior knowledge of the motion in order to better guide policy network updates.

1.5 Proximal Policy Optimization Algorithm

The PPO[15] algorithm is an innovative DRL algorithm developed by OpenAI in 2017. PPO builds upon the policy gradient (PG)[18] algorithm, addresses issues of inefficient sampling, and slows convergence when the PG is sensitive to hyperparameters. The agent initially interacts with the environment using π_θ, and the policy network learns from the gathered data. Updates to the parameters θ result in a new policy $\pi_{\theta'}$ being acquired. This in turn necessitates the re-sampling of the training data, leading to an exceedingly low utilization efficiency of the sampled data in the traditional PG method[19]. The PG algorithm uses

$$\nabla \bar{R}_{\pi_\theta} = E_{\tau \sim p_{\pi_\theta}(t)}[R(\tau) \nabla \ln p_{\pi_\theta}(\tau)], \tag{13}$$

to update the policy, where τ is the trajectory created by policy π_θ, θ is the parameter of policy π_θ, $R(\tau)$ is the sum of trajectory rewards, $p_{\pi_\theta}(\tau)$ is the probability of a trajectory, and $\bar{R}_\theta$ is the expected rewards.

To reuse training data, the PPO algorithm is to use importance sampling in order to utilize data from another distribution to estimate the expected value of the current distribution[19]. For this purpose, the importance weight $p(x)/q(x)$ is introduced to handle the difference between the two distributions:

$$\begin{aligned} E_{x\sim p}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i) = \\ \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}dx = \\ E_{x\sim q}\Big[f(x)\frac{p(x)}{q(x)}\Big], \end{aligned} \tag{14}$$

where $E_{x\sim p}[f(x)]$ is the expectation of the function $f(x)$, $p(x)$ is a distribution of the x, and $q(x)$ is the another distribution of the x.

Following the use of important sampling, the predicted reward update formula is modified from Eq. (13) to

$$\nabla \bar{R}_{\pi_\theta} = E_{\tau \sim p_{\pi_{\theta'}}(t)}\Big[\frac{p_{\pi_\theta}(t)}{p_{\pi_{\theta'}}(t)} R(\tau) \nabla \ln p_{\pi_\theta}(\tau)\Big], \tag{15}$$

where $p_{\pi_{\theta'}}$ is the new probability after the updates of θ.

The predicted reward update equation is amended from Eq. (13) to Eq. (15), subsequent to the utilization of importance sampling. It should be noted that the difference between the two distributions should not be too large, otherwise it would take a substantial amount of sampling in order to compute an approximate result[19].

After $R(\tau)$ in Eq. (15) is replaced with the advantage function $A^\theta(s_t, a_t)$, Eq. (15) can be transformed as

$$\begin{aligned} \nabla \bar{R}_\theta = E_{\tau \sim p_{\theta'}(t)}\Big[\frac{p_\theta(s_t, a_t)}{p_{\theta'}(s_t, a_t)} A^\theta(s_t, a_t) \nabla \log p_\theta(\tau)\Big] = \\ E_{\tau \sim p_{\theta'}(t)}\Big[\frac{p_\theta(a_t|s_t)p_\theta(s_t)}{p_{\theta'}(a_t|s_t)p_{\theta'}(s_t)} A^\theta(s_t, a_t) \nabla \log p_\theta(a_t|s_t)\Big]. \end{aligned} \tag{16}$$

According to the theorem of importance sampling, the difference between the two distributions is expected to be modest, that is $p_\theta(s_t) \approx p_{\theta'}(s_t)$. Without a constraint, the maximization of expected reward would lead to an overly large policy update; thus it becomes imperative to clip the probability ratio $p_\theta(s_t, a_t)/p_{\theta'}(s_t, a_t)$. The surrogate objective is

$$\begin{aligned} J^{\theta'}(\theta) = E_{(s_t,a_t)\sim \pi_{\theta'}}\Big[\min\Big(\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t, a_t), \\ \mathrm{clip}\Big(\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t, a_t), 1-\sigma, 1+\sigma\Big)\Big)\Big], \end{aligned} \tag{17}$$

where $A^\theta(s_t, a_t)$ uses generalized advantaged estimation (GAE)[20], and the clip function modifies the surrogate objective by clipping the probability ratio.

1.6 Algorithmic Framework

This paper proposes a novel gait learning algorithm for quadruped robots, which integrates PPO and priori knowledge that comprise evolutionary guidance. The algorithm framework is shown in Fig. 1.

First, the FTG provides periodic signals that serve as a priori knowledge for the policy network, which is combined with the observation of the robot in the environment as a state. Then, the policy network output and the signal from the FTG are superimposed to produce the robot's final control signal, and the PD controller is used to drive the motor motions to the corresponding locations defined by the signal. Next, FTG searches the trajectory space using an evolution strategy and optimizes the parameters of the corresponding

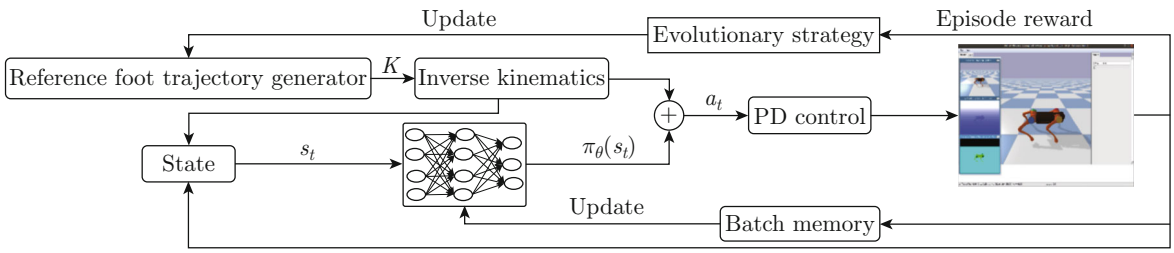


Fig. 1 Algorithm framework

CPG-RBF neural network. Finally, the PPO method is used to optimize the policy network through the data in the experience pool, allowing the robot to acquire motion abilities through its interactions in the simulated environment.

2 Experiment and Result Analysis

2.1 Simulation Experiment Environment

Unitree A1 is a quadruped robot with 12 joints, each driven by a brushless motor. As shown in Fig. 2, each leg has three joints, which include the rolling hip joint, pitching hip joint, and pitching knee joint. Four legs of the robot are labeled FR (front right), FL (front left), BR (back right), and BL (back left). The torso of Unitree A1 has a size of 0.5 m × 0.3 m × 0.4 m (length × width × height), with a total mass of 12.7 kg. An IMU integrated into the torso allows for sensing of the robot's attitude, velocity, and angular velocity; additionally, each joint features an angle sensor for further information retrieval.

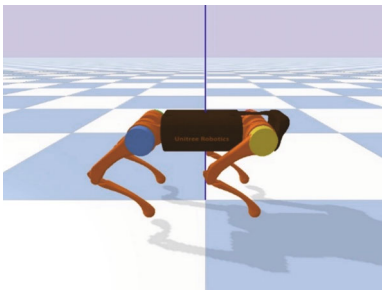


Fig. 2 Unitree A1 robot

All experiments were conducted using a laptop equipped with an NVIDIA GeForce GTX 3060 GPU and running the Ubuntu 18.04 operating system. The Pybullet[21] physics engine was utilized to simulate the training environment. The robot interacted with flat terrain during experiments. Each training episode was composed of two situations: one in which the robot fell and was unable to proceed and the other in which the training steps exceeded the maximum limit of 600 steps. At the end of each training episode, the Pybullet environment would be reset and the robot would be brought back to the origin to start again.

2.2 Results and Analysis of Trained Model

2.2.1 *Episode Reward During Quadruped Training*

To evaluate the efficacy of the proposed EEPPO algorithm, it was compared with the soft actor-critic (SAC) algorithm, an off-policy model-free DRL algorithm. We trained a robot for 8 million steps, and the results are reflected in Fig. 3, which represent the sum reward value of each episode of the robot in the simulation environment. The red curve in Fig. 3 represents the EEPPO algorithm and the blue curve stands for the SAC algorithm. The EEPPO algorithm was able to reach convergence at 1.6 million steps, retaining a reward value of 3 300 during its steady-state, and manifesting a stable converged policy. In contrast, the SAC algorithm only attained convergence after 0.4 million steps, securing an average reward value of 2 400, which is markedly lower than that turned out by the EEPPO algorithm, as well as presenting an unstable converged policy.

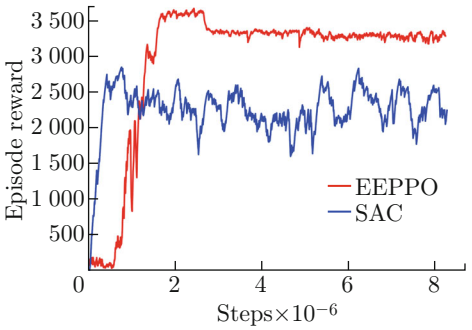


Fig. 3 Reward curves when training

According to Fig. 3, in the beginning of training phase, the proposed EEPPO algorithm needs to explore the environment and has a lot of interactions to gain experience. However, SAC is an off-policy learning method that takes temporal-difference updates, so it is easier to select actions with high scores during policy learning. Therefore, rewards can increase quickly in the initial training phase. As for EEPPO algorithm, due to its stability in training, it has a lot of exploration processes at the beginning. After it gets the message of forward stable movement, the training curve achieved a rapid increase at 700 000 steps. When the increasing speeds of two curves are almost the same, because the ultimate reward of EEPPO algorithm is 30% higher than that of SAC algorithm, thus its convergence point is later than that of SAC algorithm.

2.2.2 *Displacement and Velocity Analysis of Quadruped*

Figure 4 illustrates the horizontal distance traveled by the robot's center of mass in x direction while traversing flat terrain at a consistent velocity of

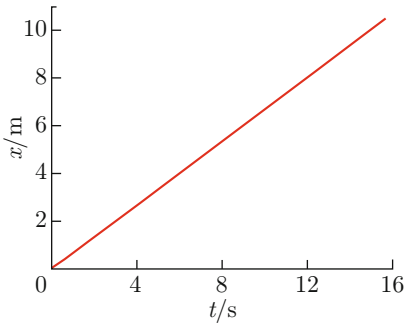


Fig. 4 Distance of the moved center of mass

0.67 m/s in a duration of 15.6 s. This indicates that the robot's trajectory is continuous, progressing in a forward direction.

2.2.3 *Analysis of Quadruped's Swing Angle*

The data analysis of the real roll angle and pitch angle of the robot traveling 600 steps in the simulated environment is shown in Table 1. The standard deviation of the roll angle is 0.030 209 rad, the standard deviation of the pitch angle is 0.015 830 rad, the average swing angle of the robot is less than 0.004 865 rad ($< 0.3°$) and the maximum swing angle is 0.069 181 rad ($< 4°$). The above data shows the feasibility and stability of the gait controller when the robot is walking.

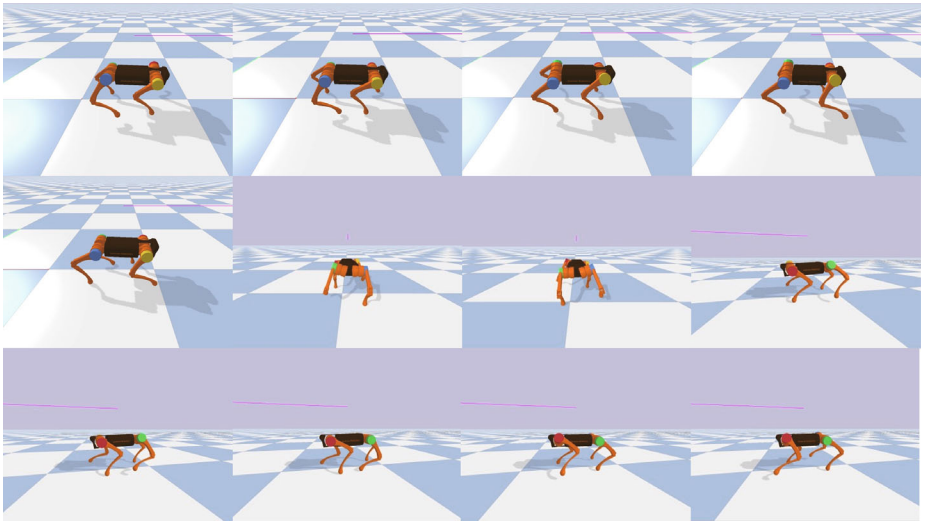
Table 1 Roll and pitch angle data obtained from the robot's IMU rad

Angle	Standard deviation	Average value	Maximum value	Minimum value
Roll	0.030 209	−0.004 557	0.069 181	−0.086 768
Pitch	0.015 830	−0.004 865	0.058 755	−0.030 999

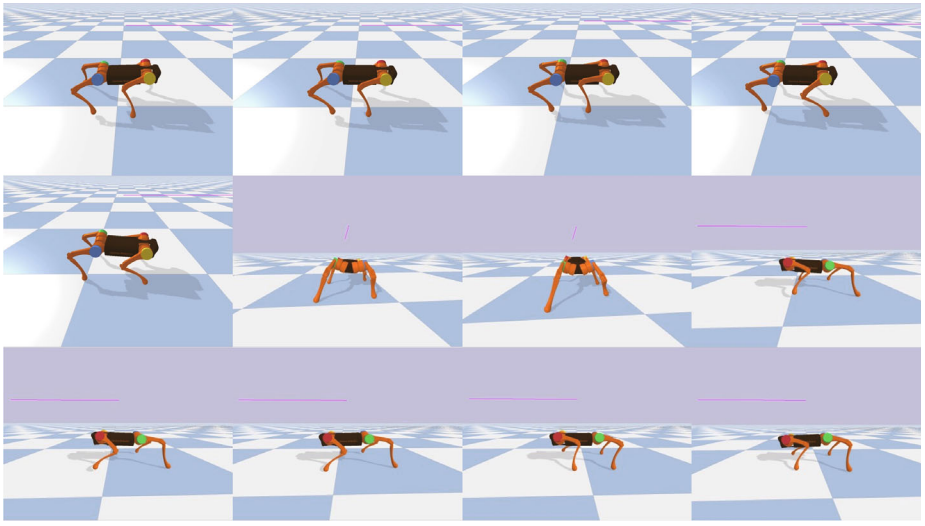
2.3 Performance Comparison Analysis Between EEPPO and SAC

2.3.1 *Screenshot Analysis of Quadruped Movement*

Figure 5 shows the motion screenshots of the quadruped robot within a simulated environment following training with the EEPPO and SAC algorithms, photographed from the right side, rear side, and left side respectively. The EEPPO algorithm enables the robot to move with greater stability and coordination on the flat terrain, indicated by the same alternating movement of all four legs of the robot. In contrast, the SAC algorithm model results in the left hind leg of the robot continuously dragging on the ground. The quadruped robot employs the EEPPO algorithm model



(a) EEPPO



(b) SAC

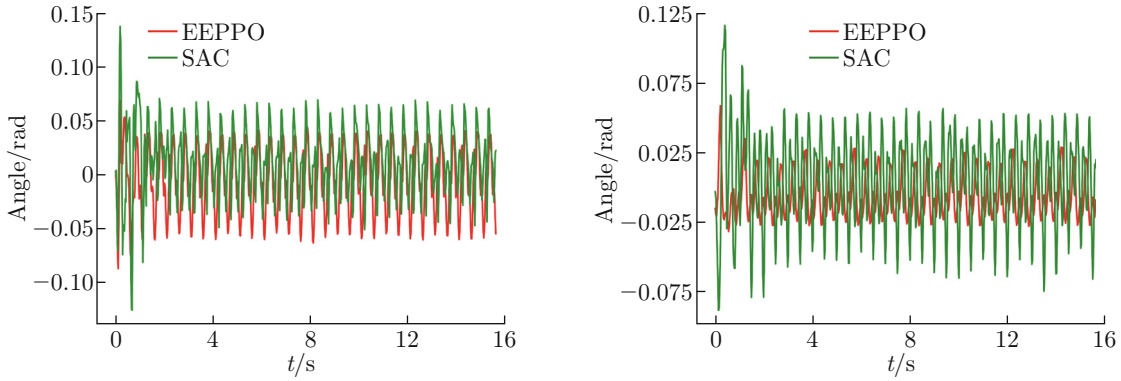
Fig. 5 Motion state of the robot using EEPPO algorithm and SAC algorithm

(shown in motion video 1: https://v.youku.com/v_show/id_XNTkzMjM3MDUyNA==.html?scm=20140719.manual.114461.video_XNTkzMjM3MDUyNA==) and the SAC algorithm model (shown in motion video 2: https://v.youku.com/v_show/id_XNTkzMjM2OTIyNA==.html?spm=a2hbt.13141534.1_2.d_0&scm=20140719.manual.114461.video_ XNTkzMjM2OTIyNA==). Through these visualizations, it can be determined that the EEPPO algorithm is superior to the SAC algorithm, as the former allows for a smoother and more stable movement of the robot.

2.3.2 *Comparative Analysis of Attitude Angles*

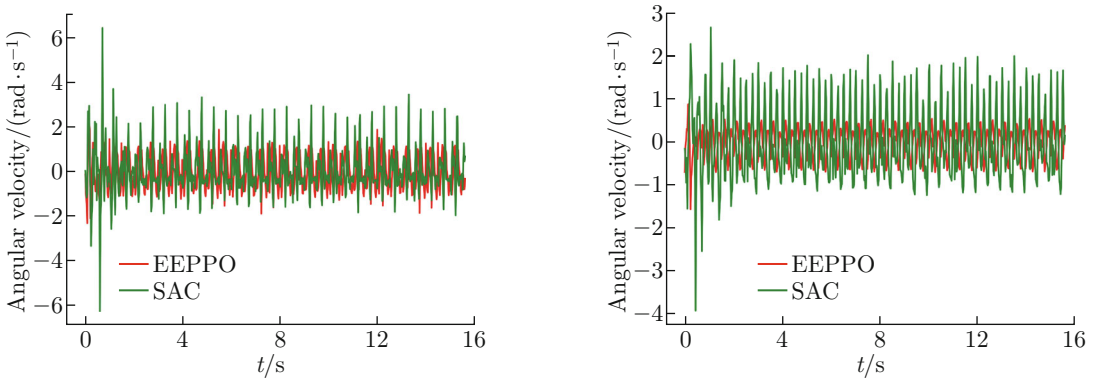
Figures 6 and 7 respectively show the change in the roll and pitch angles, and the roll angular velocity and pitch angular velocity of the center of mass for the simulation experiment of a robot walking on the flat terrain. The red line represents the data from the EEPPO algorithm, whereas the green line represents the data from the SAC algorithm.

The standard deviation of EEPPO for roll angle is 0.030 209 rad, whereas the standard deviation of SAC is 0.032 048 rad, which is slightly bigger. Furthermore, the maximum roll angle of EEPPO is 0.069 181 rad, which is much smaller than the 0.137 727 rad of SAC. For the pitch angle, the standard deviation of EEPPO is 0.015 830 rad, much smaller than 0.033 433 rad of SAC, and the maximum pitch angle of EEPPO is 0.058 755 rad, much smaller than 0.116 447 rad of SAC. According to Fig. 7, the peaks and troughs of the angular velocity of the roll and pitch angles of EEPPO are substantially fewer than those of SAC. It can be concluded that the model trained by the EEPPO algorithm in the simulation experiments is more stable and has less jitter during movement than the model trained by the SAC algorithm.



(a) Roll angle (b) Pitch angle

Fig. 6 Swing angle of EEPPO vs. SAC



(a) Roll angular velocity (b) Pitch angular velocity

Fig. 7 Swing angular velocity of EEPPO vs. SAC

2.3.3 *Comparative Analysis of Foot-End Trajectories*

Figure 8 illustrates the four foot-end trajectories in the xOz plane of the centre-of-mass coordinate system of the quadruped robot. The red lines represent the foot end trajectories of EEPPO, whereas the green ones show the SAC trajectories. It is apparent from Fig. 8 that the EEPPO foot-end trajectory exhibits a more consistent pattern during the period of robot

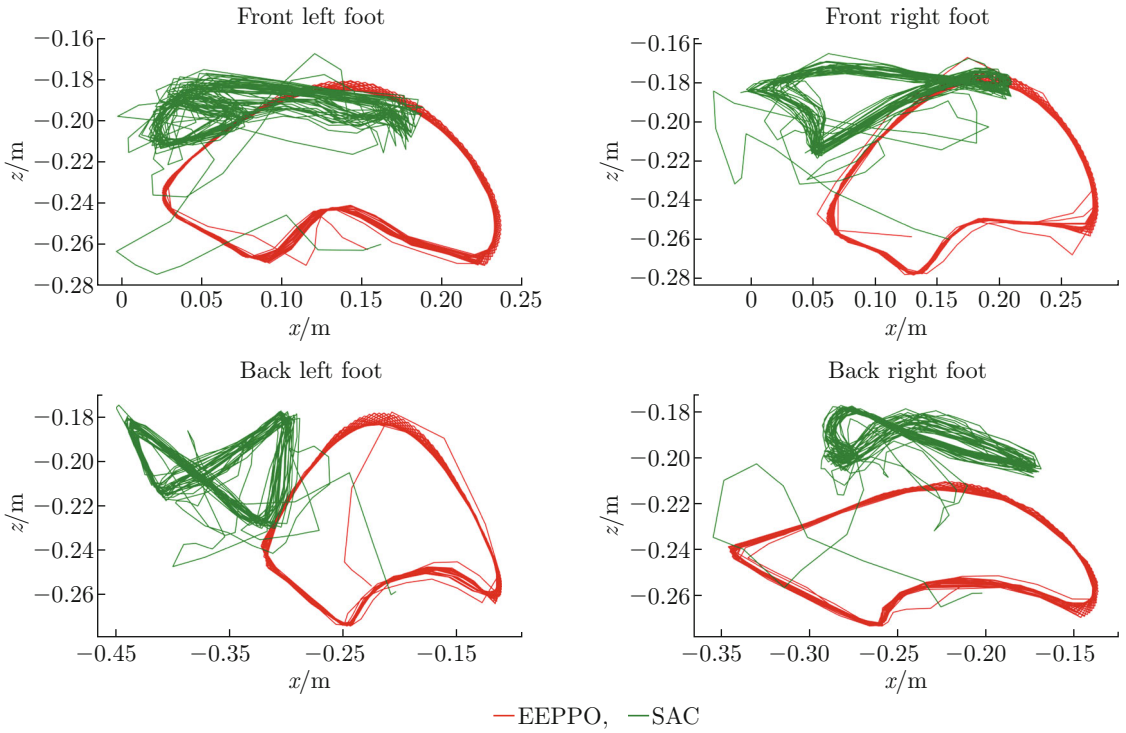


Fig. 8 Foot trajectory

mobility with 600 steps than that of SAC, which exhibits an irregularity in its trajectory. In addition, the four EEPPO foot-end trajectories display a similar shape, whereas the four SAC trajectories differ in shape. As a result, the EEPPO algorithm produces reduced fluctuations in swing angles and angular velocities, resulting in a more stable forward motion as well as a better coordinated movement of the robot.

3 Conclusion

In this paper, we propose the EEPPO algorithm for quadruped robot bionic gait learning to address the task of walking forward stably on flat terrain. To achieve bionic gait learning, we leverage a CPG-RBF neural network as the FTG, optimize the priori knowledge using an ES, and use the optimized prior knowledge to drive the PPO algorithm to update the policy network. The simulation experiments verified the feasibility and effectiveness of the proposed EEPPO algorithm, which enabled the quadruped robot to acquire a stable forward trot gait while constantly interacting with its environment. Furthermore, the comparison between the SAC algorithm and our proposed EEPPO algorithm reveals that when the model trained by EEPPO is used, the training stability and motion stability of the quadruped robot have been improved. In order to further validate our algorithm, future work entails deploying physical experiments in real-world applications.

Conflict of Interest The authors declare no conflict of interest.

References

[1] YANG J J, SUN H, WANG C H, et al. An overview of quadruped robots [J]. *Navigation Positioning and Timing*, 2019, **6**(5): 61-73 (in Chinese).

[2] ZHANG W, TAN W H, LI Y B. Locmotion control of quadruped robot based on deep reinforcement learning: Review and prospect [J]. *Journal of Shandong University* (*Health Sciences*), 2020, **58**(8): 61-66 (in Chinese).

[3] KOHL N, STONE P. Policy gradient reinforcement learning for fast quadrupedal locomotion [C]//*IEEE International Conference on Robotics and Automation*, 2004. New Orleans: IEEE, 2004: 2619-2624.

[4] YANG C Y, YUAN K, ZHU Q G, et al. Multi-expert learning of adaptive legged locomotion [J]. *Science Robotics*, 2020, **5**(49): eabb2174.

[5] LEE J, HWANGBO J, WELLHAUSEN L, et al. Learning quadrupedal locomotion over challenging terrain [J]. *Science Robotics*, 2020, **5**(47): eabc5986.

[6] THOR M, KULVICIUS T, MANOONPONG P. Generic neural locomotion control framework for legged robots [J]. *IEEE Transactions on Neural Networks and Learning Systems*, 2021, **32**(9): 4013-4025.

[7] PENG X B, ABBEEL P, LEVINE S, et al. DeepMimic: Example-guided deep reinforcement learning of physics-based character skills [J]. *ACM Transactions on Graphics*, 2018, **37**(4): 1-14.

[8] PENG X B, COUMANS E, ZHANG T N, et al. Learning agile robotic locomotion skills by imitating animals [DB/OL]. (2020-04-02). https://arxiv.org/abs/2004.00784

[9] RAHME M, ABRAHAM I, ELWIN M L, et al. Linear policies are sufficient to enable low-cost quadrupedal robots to traverse rough terrain [C]//*2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Prague: IEEE, 2021: 8469-8476.

[10] TAN J, ZHANG T, COUMANS E, et al. Sim-to-real: Learning agile locomotion for quadruped robots [J]. (2018-04-27). https://arxiv.org/abs/1804.10332

[11] WANG Z, CHEN C L, DONG D Y. Instance weighted incremental evolution strategies for reinforcement learning in dynamic environments [J]. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. https://doi.org/10.1109/TNNLS.2022.3160173

[12] BELLEGARDA G, CHEN Y Y, LIU Z C, et al. Robust high-speed running for quadruped robots via deep reinforcement learning [C]//*2022 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Kyoto: IEEE, 2022: 10364-10370.

[13] SHENG J P, CHEN Y Y, FANG X, et al. Bio-inspired rhythmic locomotion for quadruped robots [J]. *IEEE Robotics and Automation Letters*, 2022, **7**(3): 6782-6789.

[14] SHI H J, ZHOU B, ZENG H S, et al. Reinforcement learning with evolutionary trajectory generator: A general approach for quadrupedal locomotion [J]. *IEEE Robotics and Automation Letters*, 2022, **7**(2): 3085-3092.

[15] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal policy optimization algorithms [DB/OL]. (2017-07-20). https://arxiv.org/abs/1707.06347

[16] PITCHAI M, XIONG X F, THOR M, et al. CPG driven RBF network control with reinforcement learning for gait optimization of a dung beetle-like robot [M]//Artificial neural networks and machine learning – ICANN 2019: Theoretical neural computation. Cham: Springer, 2019: 698-710.

[17] SALIMANS T, HO J, CHEN X, et al. Evolution strategies as a scalable alternative to reinforcement learning [DB/OL]. (2017-05-10). https://arxiv.org/abs/1703.03864

[18] SUTTON R S, MCALLESTER D, SINGH S, et al. Policy gradient methods for reinforcement learning with function approximation [C]// *12th International Conference on Neural Information Processing Systems*. Denver: ACM, 1999: 1057-1063.

[19] BIE T, ZHU X Q, FU Y, et al. Safety priority path planning method based on Safe-PPO algorithm [J]. *Journal of Beijing University of Aeronautics and Astronautics*, 2023, **49**(8): 2108-2118 (in Chinese).

[20] SCHULMAN J, MORITZ P, LEVINE S, et al. High-dimensional continuous control using generalized advantage estimation [DB/OL]. (2015-06-08). https://arxiv.org/abs/1506.02438

[21] COUMANS E, BAI Y F. PyBullet quickstart guide [EB/OL]. [2023-02-01]. https://usermanual.wiki/Document/PyBullet20Quickstart20Guide.543993445.pdf