Received: 17 July 2022 | Revised: 4 August 2022 | Accepted: 9 August 2022

IET Cyber-Systems and Robotics

DOI: 10.1049/csy2.12062

ZHEJIANG UNIVERSITY PRESS | IET The Institution of Engineering and Technology | WILEY

ORIGINAL RESEARCH

Efficient learning of robust quadruped bounding using pretrained neural networks

Zhicheng Wang[1] | Anqiao Li[1] | Yixiao Zheng[1] | Anhuan Xie[2] | Zhibin Li[3] | Jun Wu[1,4] | Qiuguo Zhu[1,4]

[1]Institute of Cyber-Systems and Control, Zhejiang University, Zhejiang, China

[2]Intelligent Robot Research Center, Zhejiang Lab, Zhejiang, China

[3]Department of Computer Science, University College London, London, UK

[4]State Key Laboratory of Industrial Control Technology, Zhejiang University, Zhejiang, China

Correspondence

Qiuguo Zhu, Institute of Cyber-Systems and Control, Zhejiang University, Zhejiang, 310027, China.
Email: qgzhu@zju.edu.cn

Funding information

Key Research Project of Zhejiang Lab, Grant/Award Number: 2021NB0AL03; Key R&D Program of China, Grant/Award Number: 2020YFB1313300

Abstract

Bounding is one of the important gaits in quadrupedal locomotion for negotiating obstacles. The authors proposed an effective approach that can learn robust bounding gaits more efficiently despite its large variation in dynamic body movements. The authors first pretrained the neural network (NN) based on data from a robot operated by conventional model-based controllers, and then further optimised the pretrained NN via deep reinforcement learning (DRL). In particular, the authors designed a reward function considering contact points and phases to enforce the gait symmetry and periodicity, which improved the bounding performance. The NN-based feedback controller was learned in the simulation and directly deployed on the real quadruped robot Jueying Mini successfully. A variety of environments are presented both indoors and outdoors with the authors' approach. The authors' approach shows efficient computing and good locomotion results by the Jueying Mini quadrupedal robot bounding over uneven terrain.

KEYWORDS

legged locomotion, reinforcement learning, robot learning

1 | INTRODUCTION

Legged robots attracted more attention in recent years for their versatile motion capabilities. The motion planning and control of a legged robot has been well researched. Methods based on reduce-order models are proven to be feasible for generating adaptive gaits for real robots using prior knowledge and fine-tuning. In search of higher generalising performance and agility, learning-based approaches, such as the deep reinforcement learning (DRL), have gained new trends in legged locomotion control to solve these problems because they allow learning multiple input and multiple output feedback control policies that can run in real-time, particularly dealing with very high dimensional sensory inputs.

Constrained by the capability of computing devices and legged robots, DRL has not been applied to motion control for quadruped robots until recent years. Iscen et al. [1] achieved multiple gaits, including running and bounding, with the help of a predefined trajectory generator. The work of Ref. [2, 3] first implemented DRL-trained walking, trotting, and galloping on a real Minitaur robot and verified the feasibility of the end-to-end route. Subsequently, Ha et al. [4] utilised an on-robot DRL method with minimal human interference by constructing a physical reset mechanism quite similar to that of a

computer simulation and achieved trotting and walking on unstructured terrain. The work in Ref. [5–7] presents learning separate skills such as trotting and fall recovery using an end-to-end DRL framework.

When facing discrete terrain, Tsounis et al. [8] introduced DeepGait, a gait planner with a metachronal gait. With the help of a Timed Convolutional Network and privilege imitation, the robot conquered a series of challenging terrains using trotting [9]. To imitate real quadruped animals to the greatest extent, Peng et al. [10–12] collected locomotion data from real dogs and achieved trotting and spinning in robots by domain adaptation. On the basis of learning single motion, to generate multiple locomotion skills within one framework, the multi-expert learning architecture was proposed to fuse multiple neural networks (NNs) into a synthesised one [13].

Based on DRL, researchers have to face the reward hacking problem, an insecure situation in which agents obtain a reward in an unexpected way. One of its reasons is that the optimisation randomly falls into a local optimum other than the expected optimum [14]. The most common way to cope is to add specialised reward and tune manually [5], which partially neutralises the advantage of learning-based methods. Introducing predefined reference motion and imitation [12] can be another solution, which seeks a subtle balance between agility and constraints.

Most papers take trotting as a demonstration task. In addition to trot and gallop, bounding is also an important form of legged locomotion. It can be used to cross different obstacles and as a transition model between trotting and galloping [15]. However, bounding is harder to train by end-to-end DRL than trotting and galloping for several reasons. In bounding gait, the centre of mass (CoM) and pitch angle of the robot change more violently, which usually causes termination. Galloping and trotting outweigh bounding in stability, respectively, under high and low speed condition. As a form of reward hacking, bounding can be easily overrode to prevent falling [1]. Hence, quadruped bounding is rarely learned and shown in details in related studies but serves as a proof of other features, such as behaviour generalising [1].

Our work studied an effective solution to train an end-to-end reference-free NN controller by DRL that can perform symmetric bounding gait and can be successfully transferred to a the real robot. The main contributions are as follows:

1. We propose a pre-fitting method to initialise the weights of a warm-start policy trained with data collected from model-based policy, which prevents reward hacking and behaviour overriding.
2. We proposed an effective reward function based on contact phases that well regulate a periodic gait and resolve large variance and subsequent divergence in training because of large fluctuations in the body movements during bounding gaits.

The organisation of this paper is as follows: In Section 2, we introduce the overall structure of our pre-fitting method and DRL workflow. In Section 3, we introduced the platform on that we deployed the algorithms. In Section 4, we validate the feasibility of policy trained with our method on a physical robot and compare the results with those of conventional controllers. Finally, in Section 5, the conclusion, along with inspiration for future work, is stated.

2 | METHOD

2.1 | Main workflow of DRL

The main workflow is shown in Figure 1. The scratch NN is trained on a dataset gathered from a low-cost model-based policy with supervised learning. Then, we initialise the policy in the main DRL loop with the parameters of pre-fitted NN. The conventional policy and pre-fitting algorithms are introduced in Section 2.3.

In the main training loop in simulation, the state vector is retrieved from the simulation environment. It is a 34-dimensional vector comprising 1-dimensional body height, 3-dimensional body orientation, 3-dimensional body linear velocity expressed in body frame, 3-dimensional body angular velocity expressed in body frame, 12-dimensional joint position and 12-dimensional joint velocity. The clock signal is excluded, so the controller is not time-based.

The state vector is then sent to the deep neural network (DNN) as a frame of input, and the DNN instantly generates a corresponding frame of action, consisting of a 12-dimensional output vector, which represents the expected joint positions. A stable proportional-derivative (PD) controller was used to convert the expected joint position into a 12-dimensional expected joint torque vector. In addition to the basic PD torque, calculated with the joint position and velocity, an array of feedforward torques was added to the torque vectors to stabilise the robot system and help the joints move to the expected position more accurately.

The feedforward torques are expected torques calculated with the expected states of the joints with floating-base inverse dynamics compensation, where the acceleration is calculated with the difference in velocity. The torque vector is sent to the robot in the simulation environment as commands. The work loop then repeats itself. When training the NNs, input and output data vectors along with the reward values are stored in tuples for calculating the NNs' weight offsets.

After the simulation training procedure, the NN, which can perform well, is ready to be applied on an actual robot. The same structure, except for the training part, is deployed on the actual robot for offline running. Because of the difference between simulation and real robot systems, the real robot will be bound after the bridging technique mentioned in Section 2.5.

2.2 | Reinforcement learning

The motion state of the robot at a specific time is constrained by the previous state; therefore, the locomotion control is a Markov decision process, which is suitable for reinforcement

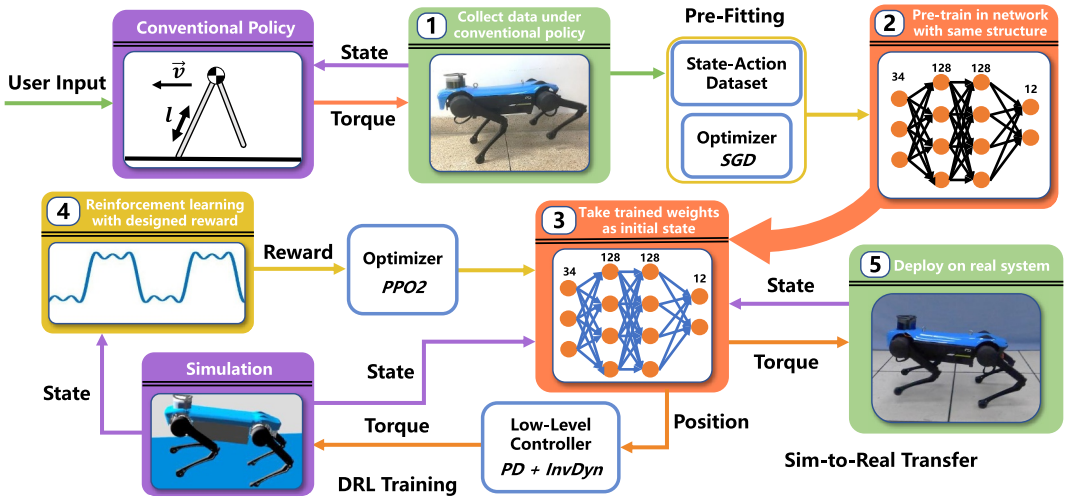

FIGURE 1 Control architecture and workflow of training and deployment, including two parts: pre-fitting and deep reinforcement learning (DRL). After training, the model is deployed on the real robot directly. PD, proportional-derivative

learning (RL). The action performed by the robot using policy influences the probability distribution of the state transition, and the result of the transition leads to a corresponding reward value, which implies how successful the state is. Hence, optimising the performance of the controller is equivalent to maximising the discounted reward function.

$$\pi^*(\theta) = \underset{\theta}{\operatorname{argmax}} \mathbb{E}_{\tau(\pi(\theta))}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \tag{1}$$

where γ is the discount factor, which implies that the importance of the reward value drops as time passes. In addition, θ refers to the parameters in the policy; in our work, θ represents the weights in the DNN. To train the DNN controller more efficiently, we chose Proximal Policy Optimisation algorithm [16, 17], which is based on Actor–Critic structure [18], in our training process.

2.3 | PF-DRL (pre-fit DRL)

There are two multi-layer perceptrons (MLPs) deployed in the Proximal Policy Optimization algorithm, named the actor and critic. The structures of the NNs are shown in Figure 1. With the random initialised weights, the most common hacking is in-place hind-leg trembling. Instead of wasting effort tuning reward coefficients, we deployed the pre-fitting method, which is efficient and often used in deep learning to solve this problem [19], and prevents introducing reference to final implementation on actual robots.

As for the pre-fitting data, we turned to model-based control for help. We recorded data corresponding to state space and action space, when the robot constrained under conventional model-based policies. The model-based policy used for data gathering was proposed by Raibert [20]. It adopts a reduce-order model and divides the locomotion task into three sub-tasks, which are velocity tracking, attitude control and orientation control. Velocity target tracking is realised by selecting foothold position according to neutral point theory and switches between phases according to a finite state machine.

After the data were collected, we constructed the NN model with the same structure as the actor net of the RL model. We divided the recorded data into a training set and validation set. The input data of the NN comprise 34 values recorded at the same time, and the label of each input includes the 12 target positions by the conventional controller after 0.01 s, which is the control frequency in the RL simulation. We trained the NN with different optimisers and learning rates in turn to minimise the mean squared error loss (Table 1). By using this hybrid training strategy, we can take advantage of high convergence speed of Adam and good generalisation performance of stochastic gradient descent optimiser. Using single optimiser with a decaying learning rate can lead to similar results, but extra tuning will be needed.

When the trained network converged, we transferred its weights into the actor net of the RL model as the initialisation and started the training process directly.

2.4 | Design of reward function

For RL processes, the reward function is a vital factor in training. The RL algorithms will automatically find the trajectory that maximises the total reward value. A proper reward function can improve the learning efficiency substantially. In our work, the reward function comprises positive terms related to the main purpose of the movement such as the planar velocity of the whole robot, negative terms (also called cost terms) to regulate the locomotion, and negative terms to restrict the energy cost and improve the safety of the robot (Table 2). Where v_x^I refers to the speed of the robot base on axis x under frame I, q_i and τ_i refer to the position and torque of joint i, respectively, ϕ refers to the pitch angle of robot torso, τ_i is the 12-dimensional torque vector at time t, ω is the frequency of desired gait, and δ_i is phase offsets of each leg. In

TABLE 1 Training sequence

Optimiser	Learning rate	Training times
SGD	1×10^{-2}	500
Adam	1×10^{-3}	500
Adam	1×10^{-4}	500

Abbreviation: SGD, stochastic gradient descent optimiser.

TABLE 2 Reward terms

Reward	Formula	Coefficient value
Body velocity	$k(\lvert v_x^I\rvert^2+\lvert v_y^I\rvert^2)$	k=160.0
Joint torque	$k\cdot\tanh(ct)\sum\tau_i$	k=−0.002, c=0.04
Joint velocity	$k\cdot\tanh(ct)\sum\dot{q}_i$	k=−0.0003, c=0.02
Gait	$k\sum_{i=0}^{3}S(t+\delta_i)G(i,t)$	k=−50.0
Position uniformity	$(\lvert q_{LF}-q_{RF}\rvert+\lvert q_{LH}-q_{RH}\rvert)$	k=−0.01
Torque uniformity	$k(\lvert\tau_{LF}-\tau_{RF}\rvert+\lvert\tau_{LH}-\tau_{RH}\rvert)$	k=−0.001
Smoothness	$k\lVert\boldsymbol{\tau}(t)-\boldsymbol{\tau}(t-dt)\rVert$	k=-1×10^{-6}
Pitch limitation	$k(\lvert\phi\rvert)$ when $\lvert\phi\rvert>0.3$	k=20.0

the case of bounding, $\delta_{\text{forelegs}}=0$, δ_{h} *indlegs* $=\frac{\pi}{2\omega}$. $S(t)$ and $G(i,\ t)$ are special functions that can be described as Equations (2) and (3).

$$\mathrm{S(t)}=\sin\omega\mathrm{t}+\frac{1}{3}\sin 3\omega\mathrm{t}+\frac{1}{5}\sin 5\omega\mathrm{t} \tag{2}$$

$$\mathrm{G(i,\ t)}=\begin{cases}+1 & \text{When foot i touches ground}\\ -1 & \text{When foot i does not touch ground}\end{cases} \tag{3}$$

Meanwhile, to train the robot to perform the desired gait, we designed a special gait signal to instruct the robot to take steps at a specific frequency. The gait signal is periodic and zero-symmetric, and it uses a signed term to convert the foot contact state to a reward value. If the foot touches the ground at an undesired time, the negative gait signal will lead to a negative reward and increase the cost (see Figure 2). In our work, we use a third-order superposition of trigonometric functions. This form of signal has a larger root mean square value and is still differentiable, which can avoid the risk of non-convergence when using simple rectangle signals.

2.5 | Bridging the sim-to-real gap

There are multiple gaps between the simulation environment and physical robots. We identify the main factors as the measurement errors when the systems are modelled.

1. To overcome uncertainty in the robot model, we added stochastic noise to both the environment and robot parameters. The terrain in the simulation was also randomised. The noise coefficients are shown in Table 3.

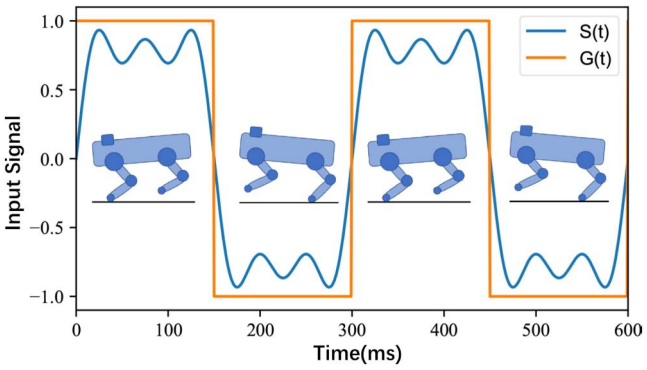

FIGURE 2 Illustration of contact states and reward. $G(i,\ t)$ represents current contact state of the robot, and $S(t)$ indicates whether that foot should touch the ground. The plot shows example signals of forelimbs when bounding with a period of 0.3 s and an illustration of an ideal case of the robot in motion in order to achieve the maximum reward in the *Gait Reward* term

TABLE 3 Noise coefficients

Noise	Standard deviation	Unit
Link mass	±5	%
Link inertia	±10	%
Link CoM	±7.5	cm
Ground friction	±0.1	
Ground restitution	0.15	

Abbreviation: CoM, centre of mass.

2. Randomise the initial head direction every episode. The yaw angle is sampled from a uniform distribution $U(-\pi,\ \pi)$. This is to prevent over-fitting to the head direction and terrain conditions.
3. To achieve high-frequency control in real time on the robot, we transform the weights of NN in CSV format and compute forward network by C++.
4. We lower the joint proportional gain to achieve better sim-to-real transfer, which proves that a lower proportional gain on a real robot can make the joints behave like a torque controller and thus lead to agile performance.

3 | PLATFORM OVERVIEW

3.1 | Robot platform

Our work is deployed on the quadrupedal robot Jueying Mini (see Figure 3). Jueying Mini is a 12-DOF quadruped robot that focuses on agility. With a weight of 22 kg and joints actuated by brushless electric motors, Jueying Mini can perform various movements. Its technical specifications are listed in Table 4.

3.2 | Modelling

To simulate the robot, we simplified it into a joint-link system comprising rigid links and revolute joints. The coordinates of

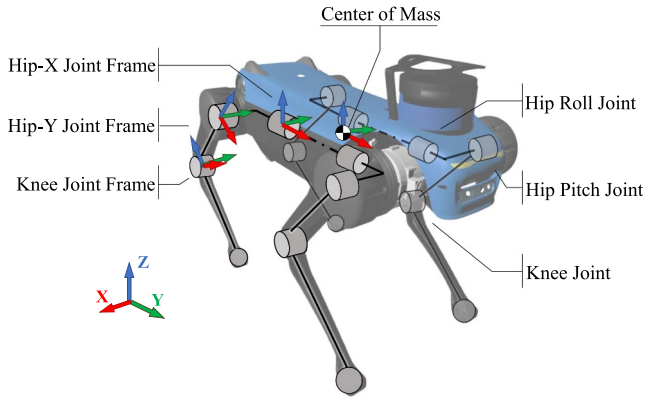

FIGURE 3 Jueying Mini robot and its kinematic configuration

TABLE 4 Technical specification of Jueying Mini

Property name	Value	Unit
Body size	0.7×0.4×0.5	m
Leg length (thigh+shank)	0.22+0.25	
Hip roll joint position	−22.0–22.0	deg
Hip pitch joint position	−158.0–28.0	
Knee joint position	38.0–163.0	
Hip roll joint velocity	−15.0–15.0	rad/s
Hip pitch joint velocity	−18.0–18.0	
Knee joint velocity	−20.0–20.0	
Hip roll joint torque	10.0 (peak 23.0)	Nm
Hip pitch joint torque	10.0 (peak 26.0)	
Knee joint torque	17.0 (peak 41.5)	

each link are located on its parent joint. The directions of all coordinates are the same when the robot is in the initial position. A simplified model of the robot and its coordinates are shown in Figure 3.

3.3 | Simulation environment

We adopted RaiSim [21] as the dynamic simulation environment. Because of its unique method of calculating contact forces, RaiSim is much faster than other dynamic simulation software. The simplified model of Jueying Mini was deployed in the RaiSim environment with the corresponding controllers.

4 | RESULTS

4.1 | Training and testing in simulation

We applied the deep pre-fitting methods on the actor network under PyTorch [22], and the training loss is shown

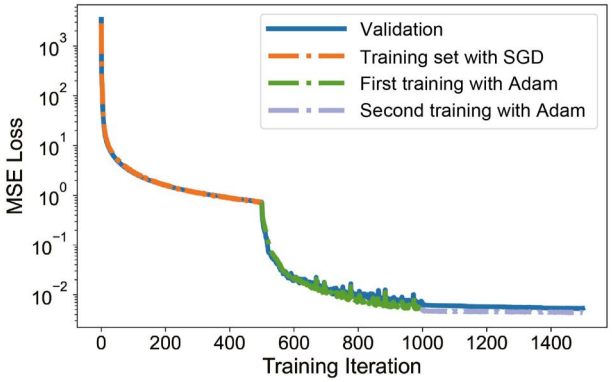

FIGURE 4 Training losses in the pre-fitting stage

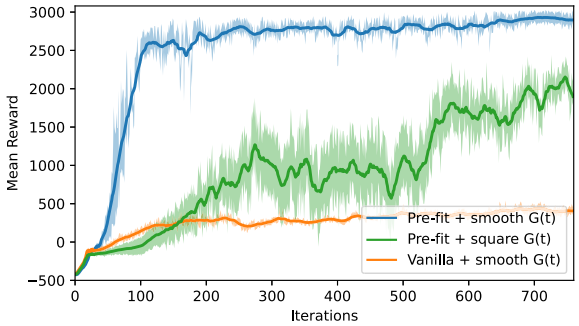

FIGURE 5 Rewards over episodes with and without pre-fitting

in Figure 4. With the introduced pre-fitting method the mean squared error loss was reduced from 10^4 to 4×10^{-3} on the training set after 1500 iterations, and the network began to perform a preliminary bounding motion but with frequent falling as a warm-start policy. On this basis, our next step is to continue training in a DRL fashion to refine the policy, which can then achieve continuous and cyclic bounding.

The pre-fitted actor network, or so-called policy network, was then put into simulation and optimised by DRL methods. We created 160 parallel RaiSim simulation environments that share the same policy network and train it to synchronously speed up the data collection. The reward curves using different methods are shown in Figure 5. Instead of bounding, the policy without pre-fitting is stuck in place and thus receives low and stable rewards compared to the other methods. The pre-fitted policy using a square gait signal performs bounding after training, but training takes longer and its reward is less stable while training. Combining both pre-fitting and smooth gait signals, the policy learns to bound in fewer iterations.

4.2 | Real robot implementation

After the two training steps above, the robot performed the desired gait on a flat terrain in simulation. The joint position,

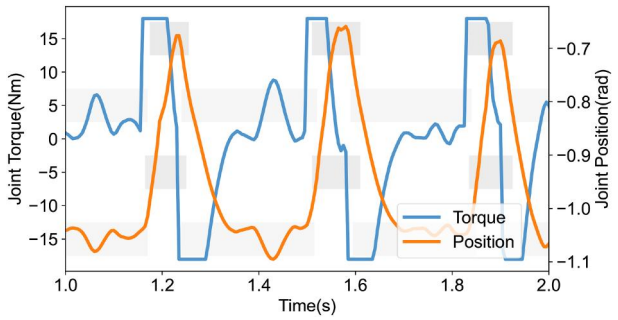

FIGURE 6 Smooth joint position and torque during bounding and grey shades are the estimated contact states

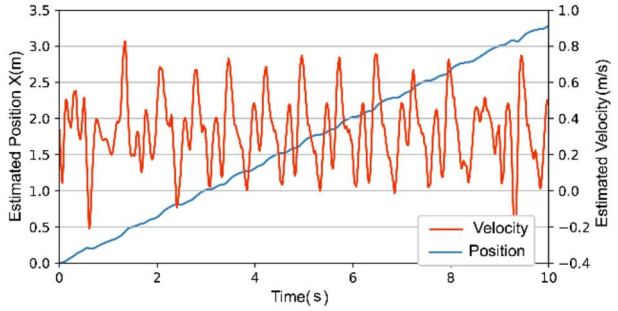

FIGURE 7 Position and forward velocity of the robot torso

velocity, and torque are shown in Figure 6. Here, we take the left hind leg of Jueying Mini as an example. In the chart, it is apparent that every data plot has a frequency of 3 Hz. The step frequency is consistent with the pre-fitting dataset.

When the policy network was optimised and able to perform a continuous and agile bounding gait in simulation, it was ready to be transferred to real robots. The policy MLP was deployed on a physical Jueying Mini robot and reached a peak speed of 0.75 m/s on flat indoor ground, with an average speed of 0.32 m/s, as shown in Figure 7. The gait performed in the real world was consistent with the simulated gait. The corresponding data are shown in Figure 8. Meanwhile, the controller showed sufficient robustness and ability to recover balance. The robot with the trained policy network was able to bound through unstructured terrain, such as grass fields, and step over small obstacles about 4 cm in height (Figures 9b and 10d). See more details in the accompanying Video S1.

With the same policy, the physical robot can also bound on unseen complex discrete terrain. In the experiment, we put aluminium profiles of different shapes on the ground. The robot could bound through successfully (Figures 9c and 10b). In other words, when the contact point of the foot landed on an aluminium profile and negatively influenced the pose of the robot, the robot would adjust its pose to avoid slipping. This

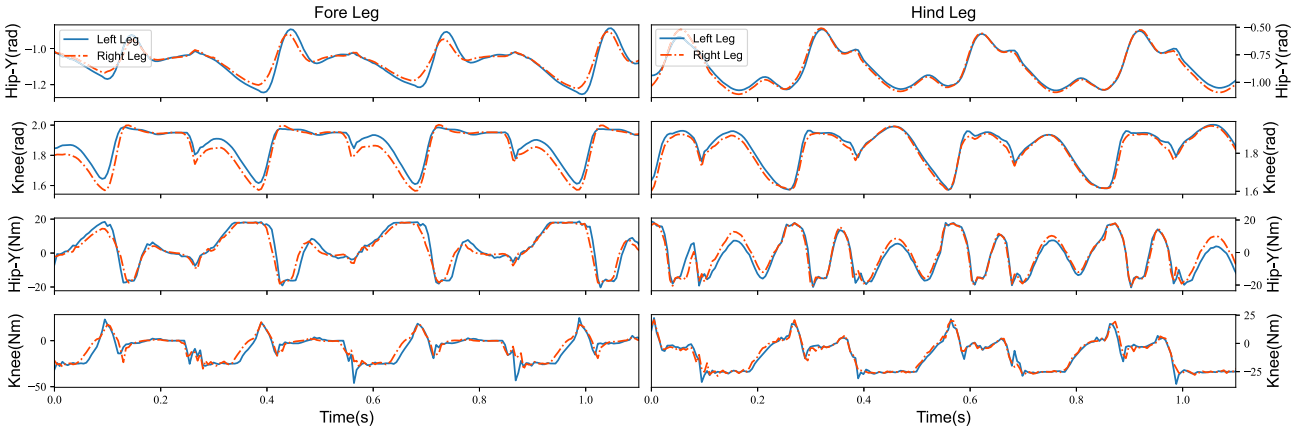

FIGURE 8 Experimental data of joint positions and torques during bounding, showing the symmetry between left and right legs

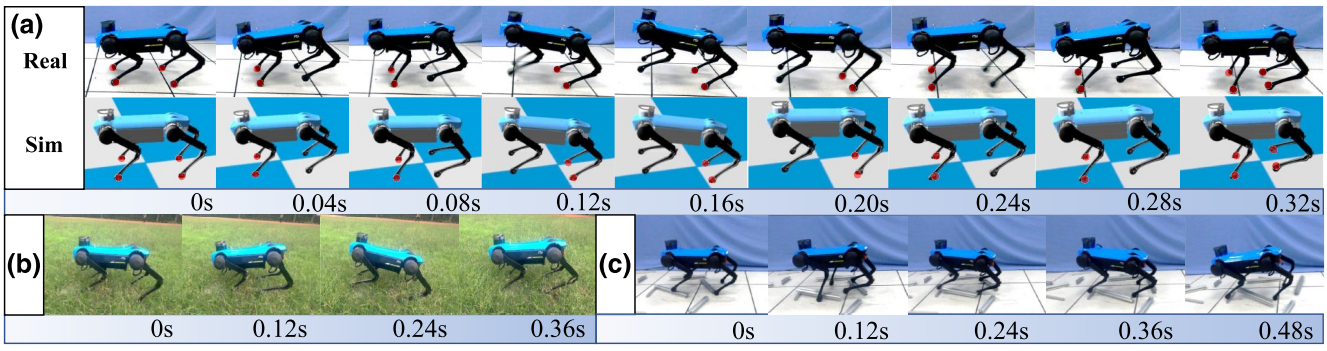

FIGURE 9 Quadruped bounding in simulation and in the indoor and outdoor environments. (a) The comparison between simulation and real world implementation; red dots highlight the contact points with the ground. (b,c) Traversal over a rough grass field and plastic floor with sparse obstacles

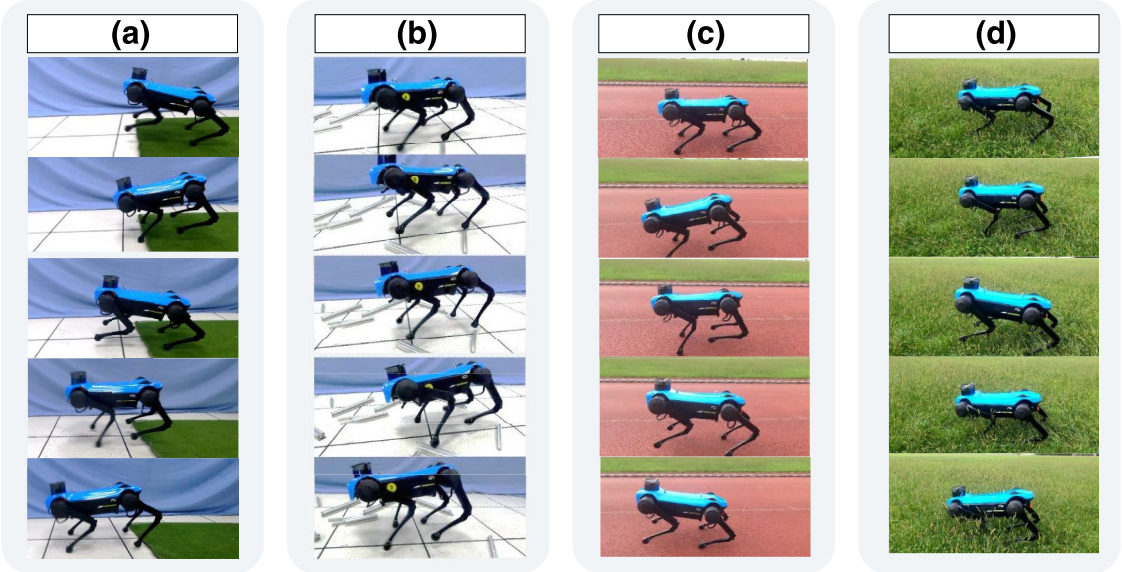

FIGURE 10 Jueying Mini robot bounding in various testing scenarios using the same PF-DRL trained policy. (a) Smooth transitions between regions with different properties (friction, stiffness). (b) Restoring balance after stepping on slippery objects twice. (c) Traversing soft outdoor ground. (d) Robust traversal on uneven grass field with shallow pits PF-DRL, pre-fit DRL

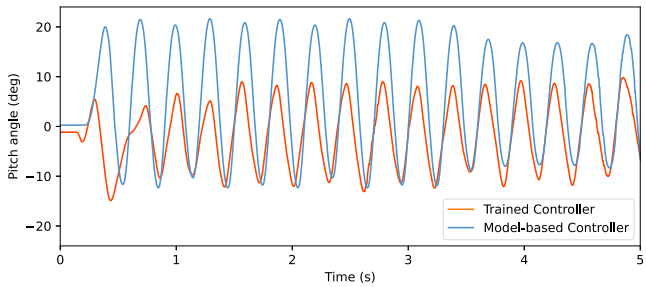

FIGURE 11 Pitch angle of the robot torso

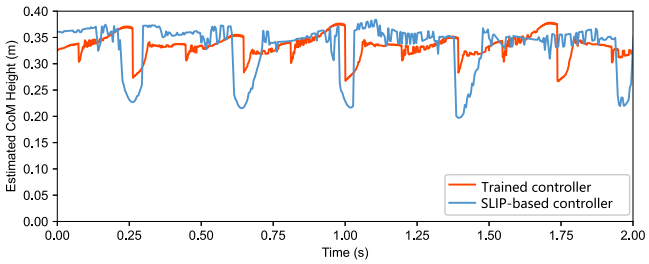

FIGURE 12 Height of centre of mass (CoM) from the robot state estimation

means that the policy we trained with RL is robust and has strong generalisation ability.

In contrast, the model-based controller used for data generation failed to pass terrain with a 4 cm metal profile. When the robot tries to lift its rear legs while fore legs step over an obstacle, the robot crashes its head onto the ground and then loses balance. This makes sense because if the robot makes contact with parts other than feet, the unexpected contact force may severely violate the decision of the model-based controller. We attribute this failure to the pitch angle fluctuation of model-based policy. Figure 11 plots pitch angles of both policies moving forward on solid flat ground. Model-based policy has a higher positive peak pitch value than the trained policy, which means the robot's head would be much lower upon reaching the peak pitch. We come to the conclusion that trained policy improves traversability on complex terrains and reduces risk of unexpected contacts by performing a safer gait with fewer fluctuations.

Despite pitch angles, the trained NN performs more smoothly when bounding in terms of body height. With trained policy, the height of the CoM changes in the range [0.25 m, 0.38 m] with a standard deviation of 0.02, whereas it fluctuates in [0.19 m, 0.41 m] with a standard deviation of 0.04 using the model-based controller used for pre-fit data generation. The comparison plot is shown in Figure 12.

5 | CONCLUSIONS

This paper presented a control method based on DRL for the Jueying Mini robot, which can achieve a bounding gait both in simulation and on the real physical robot by a direct deployment of a learned NN policy. The proposed method can train a robust control policy that can be bound blindly on different indoor and outdoor terrains.

With the proposed pre-fitting method, we can transfer the conventional controller to an NN first, as a means of warm-start, and then further improve the control policy via RL training. By changing the coefficients of each reward, we were able to adjust the gait frequency more easily than using a conventional control method.

Future work will consider extensions of aiding PF-DRL with environmental perception. In this study, we did not use

camera or LiDAR systems, but they could provide more precise localisation of the robot and provide navigation for bounding and traversal in different environments.

ACKNOWLEDGEMENT

First, we wish to give our thanks to Chao Li, Xueyin Zhang, Feng Li, Xiaobo Mo, Zhen Chu and Chengxiao Li from DeepRobotics. As the logistics staff for Jueying Mini, they kept the hardware in good condition with their effort even after serious damage, and thus created the possibility for experiments on real robots. Second, we would like to express our gratitude to Yue Wu and Yan Xia, who helped with debugging and tuning learning algorithms. Their experience on software framework and learning systems prevented us from plenty of potential time-consuming design breaches. This work was supported by the National Key R&D Program of China (Grant No. 2020YFB1313300) and Key Research Project of Zhejiang Lab (Grant No. 2021NB0AL03).

CONFLICT OF INTEREST

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Zhicheng Wang https://orcid.org/0000-0002-2657-7591
Qiuguo Zhu https://orcid.org/0000-0002-4965-5126

REFERENCES

1. Iscen, A., et al.: Policies modulating trajectory generators. In: Proceedings of the 2nd Conference on Robot Learning, Proceedings of machine learning research, Vol. 87, pp. 916–926. (2018)
2. Haarnoja, T., et al.: Learning to walk via deep reinforcement learning (2018) arXiv preprint arXiv:181211103
3. Tan, J., et al.: Sim-to-real: learning agile locomotion for quadruped robots. In: Proceedings of robotics: science and systems (RSS), Pittsburgh, Pennsylvania, pp. 1–13 (2018)
4. Ha, S., et al.: Learning to walk in the real world with minimal human effort. In: Proceedings of the 2020 conference on robot learning. Proceedings of machine learning research, Vol. 155, pp. 1110–1120. (2021)
5. Hutter, M., et al.: Anymal-a highly mobile and dynamic quadrupedal robot. In: 2016 IEEE/RSJ International conference on Intelligent robots and systems (IROS), pp. 38–44. IEEE (2016)
6. Hwangbo, J., et al.: Learning agile and dynamic motor skills for legged robots. Science Robotics 4(26), eaau5872 (2019). https://doi.org/10.1126/scirobotics.aau5872
7. Lee, J., Hwangbo, J., Hutter, M.: Robust recovery controller for a quadrupedal robot using deep reinforcement learning (2019). arXiv preprint arXiv:190107517
8. Tsounis, V., et al.: DeepGait: planning and control of quadrupedal gaits using deep reinforcement learning. IEEE Rob. Autom. Lett. 5(2), 3699–3706 (2020). https://doi.org/10.1109/lra.2020.2979660
9. Lee, J., et al.: Learning quadrupedal locomotion over challenging terrain. Science Robotics 5(47), eabc5986 (2020). https://doi.org/10.1126/scirobotics.abc5986
10. Peng, X.B., Berseth, G., van de Panne, M.: Terrain-adaptive locomotion skills using deep reinforcement learning. ACM Trans. Graph. 35(4), 1–12 (2016). https://doi.org/10.1145/2897824.2925881
11. Peng, X.B., van de Panne, M.: Learning locomotion skills using DeepRL: does the choice of action space matter. In: Proceedings of the ACM SIGGRAPH/eurographics symposium on computer animation, SCA '17, pp. 1–13. Association for Computing Machinery, New York (2017)
12. Peng, X.B., et al.: Learning agile robotic locomotion skills by imitating animals. In: Proceedings of robotics: science and systems (RSS), Corvalis, Oregon, USA, pp. 1–8 (2020)
13. Yang, C., et al.: Multi-expert learning of adaptive legged locomotion. Science Robotics 5(49), eabb2174 (2020). https://doi.org/10.1126/scirobotics.abb2174
14. Amodei, D., et al.: Concrete problems in AI safety (2016). arXiv preprint arXiv:160606565
15. Haynes, G.C., Rizzi, A.: Gaits and gait transitions for legged robots. In: 2006 IEEE international conference on robotics and automation (ICRA), pp. 1117–1122 (2006)
16. Schulman, J., et al.: Trust region policy optimization. In: Proceedings of the 32nd international conference on machine learning. Proceedings of machine learning research, Vol. 37, pp. 1889–1897 (2015)
17. Schulman, J., et al.: Proximal policy optimization algorithms (2017). arXiv preprint arXiv:170706347
18. Konda, V., Tsitsiklis, J.: Actor-critic algorithms. In: Advances in neural information processing systems 12 (NIPS 1999), vol. 12, pp. 1008–1014 (1999)
19. Bengio, Y.: Learning deep architectures for AI. Found Trends Mach. Learn. 2(1), 1–127 (2009). https://doi.org/10.1561/2200000006
20. Raibert, M., Tello, E.: Legged robots that balance. IEEE Expert 1(4), 89 (1986). https://doi.org/10.1109/mex.1986.4307016
21. Hwangbo, J., Lee, J., Hutter, M.: Per-contact iteration method for solving contact dynamics. IEEE Rob. Autom. Lett. 3(2), 895–902 (2018). https://doi.org/10.1109/lra.2018.2792536
22. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Advances in neural information processing systems 32, pp. 8024–8035. Curran Associates, Inc., Red Hook (2019)

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Wang, Z., et al.: Efficient learning of robust quadruped bounding using pretrained neural networks. IET Cyber-Syst. Robot. 1–8 (2022). https://doi.org/10.1049/csy2.12062