Machine Intelligence Research
www.mi-research.net

21(6), December 2024, 1162-1177
DOI: 10.1007/s11633-023-1429-5

Towards Jumping Skill Learning by Target-guided Policy Optimization for Quadruped Robots

Chi Zhang[1,2] Wei Zou[1,2] Ningbo Cheng[2] Shuomo Zhang[1,2]

[1]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100149, China

[2]Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

Abstract: Endowing quadruped robots with the skill to forward jump is conducive to making it overcome barriers and pass through complex terrains. In this paper, a model-free control architecture with target-guided policy optimization and deep reinforcement learning (DRL) for quadruped robot jumping is presented. First, the jumping phase is divided into take-off and flight-landing phases, and optimal strategies with soft actor-critic (SAC) are constructed for the two phases respectively. Second, policy learning including expectations, penalties in the overall jumping process, and extrinsic excitations is designed. Corresponding policies and constraints are all provided for successful take-off, excellent flight attitude and stable standing after landing. In order to avoid low efficiency of random exploration, a curiosity module is introduced as extrinsic rewards to solve this problem. Additionally, the target-guided module encourages the robot explore closer and closer to desired jumping target. Simulation results indicate that the quadruped robot can realize completed forward jumping locomotion with good horizontal and vertical distances, as well as excellent motion attitudes.

Keywords: Jumping locomotion for quadruped robot, policy optimization, deep reinforcement learning (DRL), locomotion control, robot learning.

Citation: C. Zhang, W. Zou, N. Cheng, S. Zhang. Towards jumping skill learning by target-guided policy optimization for quadruped robots. *Machine Intelligence Research*, vol.21, no.6, pp.1162–1177, 2024. http://doi.org/10.1007/s11633-023-1429-5

1 Introduction

Possessing jumping skills can endow legged mobile robots with maneuverability and flexibility for autonomous locomotion or navigation. In nature, many kinds of animals, especially those living in forests and swamps, have proven that jumping locomotion is very effective for overcoming complex terrains or obstacles[1, 2]. Legged mobile robots with jumping ability are possible to have effects on disaster rescue, intensive care, freight transportation, space exploration, etc. Thus, by combining jumping with other common locomotion modes, like walking, trotting, running and so on, legged mobile robots may be able to perform many operations of extraordinary difficulty, e.g., carrying out tasks in certain dangerous environments instead of people, and even transcend the capacity limits of human beings.

At the moment, from a whole trajectory optimization perspective, in order to realize desired jumping locomotion in robotics, one of the major difficulties is to obtain appropriate joint force or torque outputs in each instant for guiding robots to achieve expected height and displacement indices with balanced and stable attitude while jumping. Nevertheless, current jumping locomotion control, also including some other locomotion controls, are depended on manual motion planning or testing to generate trajectories[3–7]. These methods usually need a lot of time and work to give robots the ability to move within a certain limit. Furthermore, jumping, as a typical locomotion mode with high speed and agility, is of difficulty to obtain adaptive jumping trajectories to deal with all kinds of different situations just by manual work. This makes the whole design process of jumping trajectories extremely labor-intensive, and only simple or energy-driven actions have been applied in most cases.

Usually, jumping process can be roughly divided into stance phase, take-off phase, flight phase and landing phase. Different motion characteristics of each phase, and continuity between sequential phases must be considered to support a completed stable jumping locomotion[5, 8, 9]. The primary impacts such as take-off angle, torque/force outputs, position of center of mass (COM) and so on are concluded in [10]. Meanwhile, unlike walking or trotting, highly efficient and violent phase transitions, as well as long flight time with incomplete controllability also bring great challenges to the jumping locomotion control.

In the last decade, deep reinforcement learning (DRL) has been utilized to train locomotion controller for complex robot control tasks in continuous or discrete state

Research Article
Manuscript received on August 23, 2022; accepted on February 23, 2023; published online on February 22, 2024
Recommended by Associate Editor Wei He
Colored figures are available in the online version at https://link.springer.com/journal/11633

and action spaces, which has exhibited tremendous potentials for improving robotic motion performances. For specific scenarios, including robot navigation[11−13], manipulator planning[14], UAV trajectory planning[15, 16], robot rhythmic gaits[17−19], and multi-agent systems[20], several successful examples have been made by introducing DRL and its extensions. DRL usually does not need any prior knowledge about robot dynamics, and can be utilized to robot systems without system identification in an end-to-end mode[21]. It means that the locomotion controller can be learned directly for the robot rather than a simplified model. Even though there exists difficulty for balancing exploration and exploitation, DRL is well adapted for end-to-end model-free control systems. Therefore, taking it into consideration, a control architecture with policy optimization on the basis of DRL is proposed, which can obtain a neural network locomotion controller for helping a quadruped robot to master stable and agile jumping skill. In order to make the robot jump as expected and avoid irregular attitudes, reward function which is composed of expectations and penalties is designed. In addition, aiming at the low exploration efficiency in high dimensional space and poor exploitation in local space of reinforcement learning (RL), target-guided factor and intrinsic curiosity module (ICM) are embedded into soft actor-critic (SAC) method[22] for accelerating learning process of controller optimization. Compared with state of the art works such as [23], the proposed method does not require any geometric or dynamic analysis. Automatic learning process can save manual trial and error. For similar works such as [24, 25] which also use deep reinforcement learning, our method is suitable for more general scenarios, as well as needs less computation and training time consumption to complete similar effects.

The main contributions can be summarized as follows:

1) A model-free control architecture with deep neural networks and reinforcement learning optimization for jumping locomotion of quadruped robots is proposed. The overall robotic motion system is constructed as a partially observable Markov decision process (POMDP) for making it more robust and applicable in predictable unstructured environment. Automatic parameter learning is beneficicial to reduce manual parameter design and tuning in traditional controller. Only a few hyperparameters need to be adjusted manually.

2) Efficient policy optimization which combines target-guided factor and ICM is introduced to increase exploration and accelerate convergence. Special reward design with expectations and penalties improves robustness in the jumping process.

3) Forward jumping is completed with expected jumping targets. It can be regarded as an end-to-end training framework that only needs to specify jumping displacement and height. Thus, it is also convenient for realizing different jumping behaviors, and is reflected in the training process.

2 Related works

2.1 Jumping locomotion control

In recent years, designing a jumping robot and making it jump as expected locomotion is an important issue which enhances maneuverability and flexibility of legged robots. With regard to analytic methods, modular control design with multi-step sequence or cooperation is commonly applied with divided switch models in advance. In [26], the considered robotic system model includes a time-switched model, an event-switched model and a hardware platform model. At the same time, the control process is also divided into three steps corresponding to switch models, respectively. For miniature robot JumpRoACH[27], a single DC motor in jumping module is applied to adjust the quantity of energy storage and releasing, which is aimed to control both take-off speed and jumping height. An active shell module for recovering body attitude and jumping module are integrated to provide continuous jumping ability. A control framework including two control modules, which are active compliance control and angular momentum control for dynamically-balanced jumping is designed in [28]. Ground reaction force error feedback and torque compensation via gyro sensor information evaluation are completed by the two controllers respectively. In general, controller analysis and construction are depended on accurate dynamic models and complicated nonlinear decoupling technique. While the robot model is in uncertain or violent time-varying environment, motion control performances may be poor or even ineffective.

Interpolation method is also usually used to generate jumping trajectory based on simplified polynomial dynamic models[29]. It helps robots obtain real-time system states and rapidly generate control outputs without numerical integration. This kind of method contributes to accelerating online trajectory optimization, but part of the model information may be lost, which is not conducive to increasing control accuracy and adaptation.

Excellent motion performances have been achieved by applying central pattern generators (CPG) to jumping motion control tasks. A two-level CPG control mechanism including interneurons and motoneurons for a biped robot is proposed in [30]. Interneurons are modelled to generate motion rhythm and pattern promptly for motion sequence control, and motoneurons are modelled to control output forces of joint actuators in real time. Based on this framework, unknown perturbations and environment changes from feedback information can be perceived by control systems, and outputs of neurons are adjusted to adapt to unpredictable environment. CPG models for locomotion control in robots allow for flexible modulation of a few control parameters, and provide a good substrate for learning and optimization algorithms[31].

However at present, a major obstacle to the CPG-based method in locomotion control is the lack of a solid theoretical guidance for designing and fine-tuning CPGs. It heavily relies on the development of neurobiological progress on biological CPGs.

2.2 Reinforcement learning

Due to the complexity of dynamic modeling and high dimensional control, automatic controller design and optimization, aiming to reduce manual parameters regulation, represented by RL, has become popular with the development of deep neural networks and intelligent control[32, 33]. In fact, several successful works in manipulator operation[34], UAV navigation[15, 16], athletic imitation[33, 35, 36], network selection[37], locomotion learning[38, 39], optimization[40, 41], etc. have been completed in robotic society.

One of the important advantages of RL is that it can learn polices or strategies by itself through trial and error without prior knowledge. So, it is well adapted for tasks without sufficient data, or tasks where data is difficult to collect. In addition, the property that collects data from real-time trial and error can also help RL overcome dependency of rigid and accurate dynamic models. Automatic optimization process can constrain controller parameters to the convergence domain by training without reconstructing it as an optimal control problem. End-to-end control mode from sensors to low-level controller is conducive to increase control frequency and reduce communication cost[35].

In general, robot locomotion control is usually implemented in continuous state and action spaces. But due to the possible curse of dimension, it is a huge challenge to apply RL methods to continuous control problems. In [42], a continuous variant of Q-learning named normalized advantage function (NAF) is presented, which extends Q-learning for continuous control tasks. At the same time, an actor-critic algorithm based on deterministic policy gradient (DPG) for continuous control is proposed in [43]. Thus far, RL continuous control methods mainly include on policy optimization such as trust region policy optimization (TRPO), proximal policy optimization (PPO), and off policy optimization such as deep deterministic policy gradient (DDPG), soft Q learning, SAC[44]. For on policy methods, policy gradient with online stochastic exploration might help robots find global optimal action trajectories for specific locomotion skills. Applying TRPO, a motor dynamic model with actuator network is learned using real-world data, a variety of locomotion skills such as walk or trot are deployed to ANYmal in [35]. In [18], quadruped robot Minitaur which is trained by PPO with sim-to-real strategies is given locomotion abilities of gallop and trot both in simulation and real world. PPO is also used to optimize a feedback control problem which is aimed to find an optimal policy for obtaining walking locomotion of bipedal robot Cassie in [38]. Also, in [36], a proposed imitating learning system that enables legged robot Aliengo to learn agile locomotion skills by imitating actual animals is optimized by PPO. On policy is effective in enhancing exploration, but it also faces the difficulty of long training time or slow convergence. Correspondingly, for off policy methods, experience obtained from interaction between robots and environment usually has significant effects on policy learning process. Stored experience is sampled by RL algorithm for policy network improvement. DDPG with experience replay is used in [45] for reach, push, and pick-and-place skill learning of a manipulator. In [24], SAC is applied to optimize a robust jumping task in a trajectory optimization framework with robot model. Peng et al.[43] try to use DQN with experience replay for planar locomotion skill learning in computer animation. But for the above literatures, models or prior knowledges are all used to some extents. A common design method is to introduce kinematic or dynamic models which are applied to constrain the desired motion trajectory, and then utilize DRL to optimize parameters. It is beneficial to reduce training complexity and improve control accuracy, but brings a problem on modelling.

In this paper, SAC is used to learn a policy for executing jumping locomotion. Differing from other off policy methods, SAC can obtain stochastic policy from experience. Entropy regularization is introduced in objective function to maximize weighted value between entropy and the desired reward for stability and sample efficiency[44]. However, insufficient exploration, slow convergence are still existed on model-free systems while applying SAC to complex and high dimensional continuous control of the robot[46]. For this reason, target-guided policy optimization with ICM is designed. ICM can generate intrinsic rewards by sampling from experience to encourage SAC to explore novel states. Target-guided factor concentrates on task-related exploration by providing extrinsic reward. Task-unrelated exploration that cannot improve efficiency and reduces training stability is not supported.

The rest of this paper is organized as follows. In Section 3, overall system framework including robot platform, actuator model and learning process is introduced. Section 4 illustrates jumping phase analysis and basic reinforcement learning method. In Section 5, policy optimization including expectations, penalties, and exploration mechanisms are proposed. Section 6 shows simulation verification and evaluation of our method. Conclusion and future research are given in Section 7.

3 System overview

In order to make quadruped robots master jumping ability like animals (e.g., dogs or leopards), a system structure shown in Fig. 1 is adopted for jumping skill

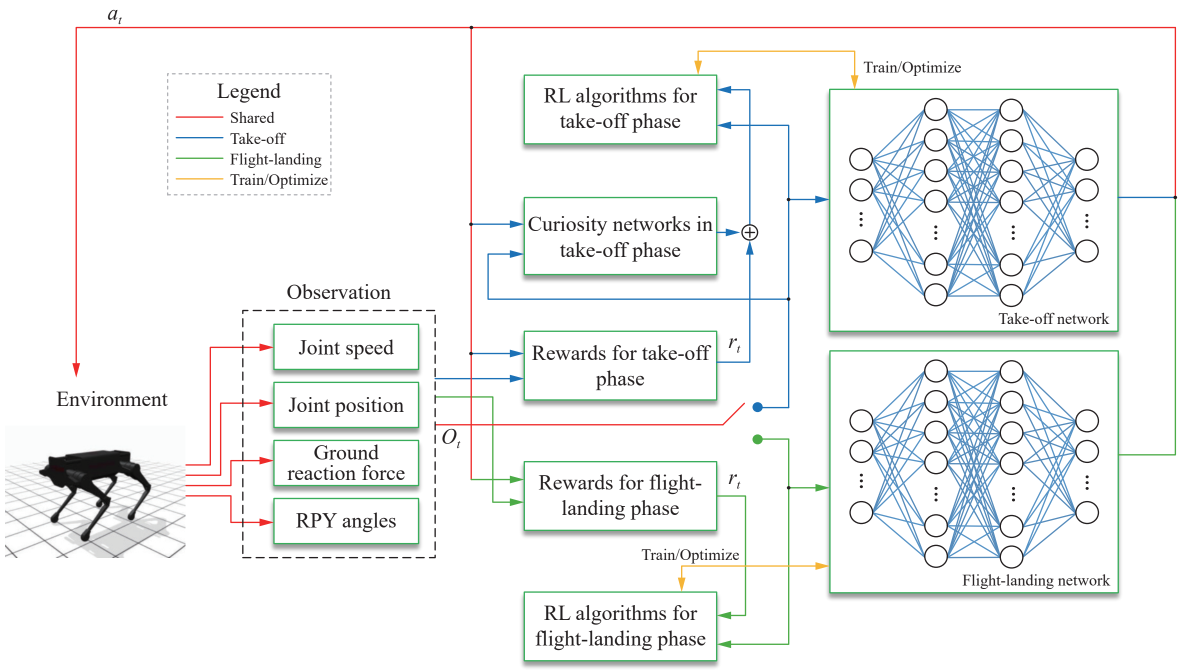


Fig. 1 Overall framework for learning jumping skill. An observation switch in this figure represents sequential unidirectional switching effect from take-off phase to flight-landing phase in one training setup. The switching condition is whether the forces between feet and ground are all zero. If all the forces are zero, observation would be switched from take-off phase to flight-landing phase. If not, phase would not be switched.

learning. The main modules of the overall framework include environment and the robot, RL algorithms, curiosity module, observation and action, as well as policy optimizations. For a completed jumping process, take-off phase and flight-landing phase occur in time sequence with relative judgement conditions, and their SAC algorithms are built by deep neural networks respectively. While jumping, observations are obtained by sensing observable environment and robot information, then are sent to RL algorithm which is constructed by SAC and curiosity module when the robot is in take-off phase, and by SAC when the robot is in flight-landing phase. Next, networks generate actions in the form of reference trajectory and send it to the robot controller for obtaining low-level control signals. While training, reward module receives observation and action information for evaluating control performance and guides RL algorithm to learn optimal policy for maximum rewards.

3.1 Jumping robot platform

The jumping robot used in simulation is Aliengo which is designed by Unitree Ltd. The specific morphology and structure can be seen in Fig. 2. Just like real quadruped animals, the trunk and legs of Aliengo are bilaterally symmetric. There are three joints which are hip, thigh and calf joints respectively on each leg. In order to avoid collisions among joints and thunk, as well as consider motion trajectory of real quadruped animals, range limitations are exerted on all the joints as shown in Table 1. Joint velocity and torque ranges can also be found in Table 1. All the joint range, velocity and torque limitations on all legs are the same.

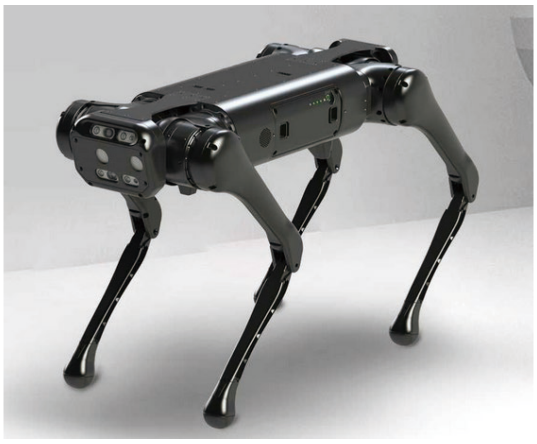


Fig. 2 Simulated jumping robot Aliengo[47]

3.2 Actuator model

A simple actuator model is introduced to control robot joints in simulated environment. The control objective is set as torque control for the low level virtual motors. In order to realize this control mode, joint positions, velocities and torques are all needed for mixed control via a PD control with feedforward torque. Therefore, its specific form is built as

Table 1 Joint range limitations on legs of Aliengo robot

Joint name	Range	Max velocity	Max torque
Thigh joint	$-120^\circ - 240^\circ$	20 rad/s	35.278 N·m
Calf joint	$-159^\circ - 37^\circ$	15.89 rad/s	44.4 N·m

$$\tau = \tau_{ff} + K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) \tag{1}$$

where $\tau_{ff} \in \mathbf{R}^4$ represents feedforward torques, and $q \in \mathbf{R}^4$ and $\dot{q} \in \mathbf{R}^4$ are relatively joint motor positions and velocities. K_p and K_d are 4×4 proportional and derivative coeffient matrices in diagonal forms respectively. Choosing proper proportional derivative (PD) coeffients could improve stabilty and convergence rate during training. It should be adjusted well before executing learning process.

3.3 Jumping skill learning process

In Fig. 1, the process of learning effective and efficient jumping skill is obtained by the interaction between environment and agent. A quadruped robot tries to learn how to jump in the designed simulated world. States of the robot include its joint information, motion information, reaction force between ground and feet, and so on. However, if a real robot is considered, not all these state information can be obtained because it would not be equipped with all kinds of sensors. Thus, in most cases, only partial state information of the robot, which is called observations, is provided to agent for further skill optimization. In order to achieve an excellent jumping locomotion with high accuracy and agility, actions are designed as a mixed control mode with joint positions, velocities and feedforward torques of the robot. Then, given the partial observations, actions for the robot jumping control are sampled from a stochastic policy distribution:

$$a_{t+1} = \pi_\theta(a_t \mid o_t) \tag{2}$$

where a_t represents action or control signal in current time instant t, o_t stands for observation information about the robot in t, and π is learned stochastic policy with parameter θ. At each time instant, the obtained actions including joint positions, velocities and torques will be sent to the corresponding joint controller of the robot as references for controlling these joints approaching to these values until jumping locomotion is done. All the action sequences would guide the robot to achieve the desired height and displacement, as well as land to the ground safely.

Performance indices of the desired jumping skill are obtained by designing appropriate rewards, as well as loss functions with penalties for unwanted behaviors. Thus, optimization process can help the robot learn a jumping locomotion skill from scratch without any prior knowledge or initial training data.

4 Reinforcement learning deployment

The jumping locomotion control problem is modelled by a POMDP with a tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, r, γ, Ω, $\mathcal{O}$), where $\mathcal{S}$ represents state space, and state $s \in \mathcal{S}$, $\mathcal{A}$ is action space and $a \in \mathcal{A}$, $\mathcal{P}$ is the transition function, which is usually expressed as a distribution of the next state s' by the current state s and action a. It is usually written as $\mathcal{P}(s' \mid (s,a))$. $r(s,a)$ stands for a scalar reward function that is used to give relative reward or penalty to the robot after action a has been taken from state s. γ is discount factor with value range [0,1). Ω represents the observation space which is a subspace of state space, and observation $o \in \Omega \subset \mathcal{S}$. $\mathcal{O}$ is transition function in the form of observation probability distribution, which is written as $\mathcal{O}(o' \mid (o,a))$. Our problem formulation shows that the robot can only obtain observations which are sampled from states. Therefore, ideal transition function $\mathcal{P}(s' \mid (s,a))$ representing dynamic modelling of the robot cannot be accessed. $\mathcal{O}(o' \mid (o,a))$ is applied in our formulation for implementing control task by reinforcement learning. Consider that SAC[48], one of the best ways to optimize RL tasks, is used in this paper as a basic learning framework to iteratively improve the policy of the robot jumping locomotion controller. The training goal is to find a set of parameters θ to achieve the optimal policy $\pi_\theta(a \mid o)$ with expected accumulated reward and with expected policy entropy $\mathcal{H}$:

$$J_\pi(\theta) = \mathrm{E}_{\sigma \sim \mathrm{p}_{\pi_\theta}(\sigma)}\left[\sum_{\mathrm{t}=0}^{\mathrm{N}} \gamma^{\mathrm{t}} \mathrm{r}^{\mathrm{t}} + \alpha\mathcal{H}(\pi_\theta)(\cdot \mid \mathrm{o_t})\right] \tag{3}$$

where α is a temperature parameter that determines the relative importance of the entropy term against the reward, and thus controls the stochasticity of the optimal policy[41]. $p_{\pi_\theta}(\sigma)$ is probability distribution of observation and action trajectories $\sigma = (o_0, a_0, \cdots, o_{N-1}, a_{N-1}, o_N, a_N)$, which is expressed as

$$p_{\pi_\theta}(\sigma) = p(o_0) \prod_{i=0}^{N-1} \mathcal{O}\big(o_{i+1} \mid (o_i, a_i)\big)\pi(a_i \mid o_i) \tag{4}$$

where o_0 is the initial observation, and a_0 represents the initial action.

In order to obtain a set of optimal parameters θ for maximizing J, policy gradient and its derivative methods are effective and effcient[49]. In SAC, learning a policy $\pi_\theta(a \mid o)$ and a critic $Q_\phi(o,a)$ (with parameter ϕ) is aimed to maximize weighted rewards and policy entropy in J_π. Then, parameter ϕ of critic in experience replay $\mathcal{D} = (o_i, a_i, r_i, o_{i+1})$ is optimized by minimizing Bellman error:

$$J_Q(\phi) = \mathrm{E}_{\sigma \sim \mathcal{D}}\Big[\big(\mathrm{Q}_\phi(\mathrm{o_i}, \mathrm{a_i}) - (\mathrm{r_i} + \gamma \mathrm{V}(\mathrm{o_{i+1}}))\big)^2\Big] \tag{5}$$

where V is the target value in the form of

$$V(o_i) = \mathrm{E}_{a_i \sim \pi}\left[Q_\phi(o_i, a_i) - \alpha \log \pi(a_i \mid o_i)\right]. \tag{6}$$

At last, the policy in (3) is learned by minimizing difference from the exponential of soft-Q function, and written as

$$J_\pi(\theta) = \mathrm{E}_{s_i \sim \mathcal{D}, a_i \sim \pi_\theta}\left[Q_\phi(o_i, a_i) - \alpha \log \pi_\theta(a_i \mid o_i)\right]. \tag{7}$$

Also, in order to enhance the ability of exploration, and accelerate network convergence, target-guided factor and curiosity mechanism are introduced in our proposed method, which will be discussed in Section 5.

4.1 Jumping phase analysis

In Fig. 3, jumping phase of the robot is divided into take-off phase and flight-landing phase, where there are two subphases for each of them. At take-off phase, stance phase is the initial state when the robot prepares to jump. Then, jumping subphase is defined as the stage from the moment the robot starts moving, to its rear legs are both off the ground. Differing from other jumping phase analysis e.g., [23], the subphase division for flight-landing phase is based on the moment when robot arrives at the climax. During the time from the end of take-off subphase to the moment when the robot arrives to the jumping climax, the robot is defined as lying in flight subphase. Then, the landing subphase is stepped into when the robot jumps across the jumping climax, until all the feet touch the ground.

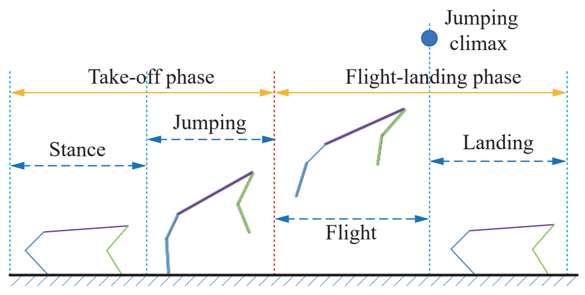


Fig. 3 Jumping phase division. Take-off phase includes stance and jumping subphases, and flight-landing phase include flight and landing subphases. Blue point in the figure represents the jumping climax in the overall process. It is used to divide desired joint positions while learning.

The reason for this definition is that the underactuated property of the robot in the air determines that attitude regulation is difficult to realize. Body attitude is mainly affected by the power generated by interaction with the ground and the time difference between the forward and the rear limbs leaving the ground. There is no extra power to help the robot change its COM position and posture at flight-landing phase before touching to the ground. Therefore, while the robot leaves the ground, the next aim of control is to adjust postures of the body and limbs for preparation to land. The model learns to generate suitable action outputs with the form of joint trajectory reference to the robot for good joint attitudes in the air. Including sudden change of velocity from the maximum to zero instantaneously while landing, it is of importance to touch the ground with an appropriate attitude and be convenient for subsequent control to initial stance phase. Also, incorrect postures or tumbles should be avoided by joint regulation at the end of flight subphase and then at landing subphase.

4.2 Observation space

The observation space in POMDP can be regarded as an observable part of state space which is not fully observable in control system because of insufficient sensor information acquisition. At the same time, too much computing burden must be considered while designing observation space both in actual application and in simulation for avoiding curse of dimension. A high-dimension observation space is adverse to reduce training cost, even makes learning process difficult to complete. Due to the body symmetry of the robot and characteristics of forward jumping, a planar structure is applied to reduce the dimension of observation space. Therefore, taking these elements into account, observation space is chosen to include two dimensional robot COM positions and velocities in vertical and horizontal directions, pitch angles obtained by inertial measurement unit (IMU) quaternions of the robot, two ground contact forces fixed on vertical and horizontal directions respectively obtained from force sensors, as well as joint positions and velocities in thigh and calf joints of fore and rear legs respectively. These seventeen elements constitute the observation space in the overall jumping phase.

It is noteworthy that COM and joint positions are both applied in observation space. COM positions can reflect jumping states, and guide the robot to approach the desired target directly. Joint positions, namely joint angles, embody the jumping attitude of the robot at each instant. Acquiring joint positions information is necessary for motion attitude control. Ground reaction forces can be used to judge whether the robot is in contact with the ground. Thereby, through the judgment of the ground contact force, reward or penalty can be given to the robot due to the rebound phenomenon while landing to the ground in reward function. Pitch angles obtained from IMU illustrate motion attitudes of the robot along horizontal direction, and can be utilized to keep the robot balance in the overall jumping process.

4.3 Action space

Taking actuator model (1) into consideration, action space which is the output of policy is designed as a combination of reference joint position, velocity and feedforward torque. Fast and accurate control is needed to complete a jumping locomotion because the time is quite short. For example, a jump with height 0.2 m just needs about 0.2 s according to $t=\sqrt{2h/g}$ where $g=9.8\ \mathrm{m/s^2}$. Also, position, velocity and feedforward torque of the actuators are all chosen for rapidity and accuracy of the robot control system. In order to ensure the constraints of actuator model and closed kinematic chains[25], control output limitations of joint angles should by satisfied by Table 1 in Section 3.

5 Policy optimization

The jumping control performance is heavily depended on the design of learning policy. Before developing the jumping strategy for the robot, it is necessary to analyze jumping skill of real quadruped animals. Reasonable take-off posture and mechanism can help a quadruped animal achieve the desired jumping target. While jumping, attitude keeping ability contributes to helping a quadruped animal stabilize its body and land to ground, and a flexible kinematic control mechanism decides that it can achieve the above motions[1]. In addition, a strong muscular system in legs and body is used to deliver high level of mechanical powers for motions. Thus, in order to acquire a jumping skill like a real quadruped animal with excellent performance, reward function is defined for take-off and flight-landing phases respectively, which is composed of different terms for ensuring the robot meeting different geometric constraints in different stages. In take-off phase, the specific form of reward function r_{tf} is

$$r_{tf}(o,a)=\lambda r_{tf0}+\omega_c r_c \tag{8}$$

where r_{tf0} is

$$r_{tf0}=\omega_{pos}r_{pos}+\omega_{vel}r_{vel}+\omega_{traj}r_{traj}+\omega_{\phi_{tf}}r_{\phi_{tf}}. \tag{9}$$

In (8) and (9), r_{pos} is jumping position term. r_{vel} is jumping velocity term obtaining from the moment off the ground, $r_{\phi_{tf}}$ is related to pitch angles ϕ of take-off phase, r_{traj} is penalty term related to joint space, and r_c is curiosity mechanism term. Also, ω_{pos}, ω_{vel}, ω_{τ}, $\omega_{\phi_{tf}}$, ω_{traj} and ω_c, are their corresponding weights. λ is target-guided factor that takes effect on task-related rewards.

In flight-landing phase, reward r_{fl} is designed as

$$r_{fl}(o,a)=\omega_{traj}r_{traj}+\omega_{com_x}r_{com_x}+\omega_{com_z}r_{com_z}+\omega_{\phi_{fl}}r_{\phi_{fl}} \tag{10}$$

where r_{traj} is defined as two desired joint trajectory evaluation terms that will be described in the following content, r_{com_x} and r_{com_z} involve rewards about COM of the robot while landing to the ground, and $r_{\phi_{fl}}$ is a term related to pitch angle in the moment of landing. ω_{traj}, ω_{com_x}, ω_{com_z} and $\omega_{\phi_{fl}}$ are their corresponding weights.

Then, the detailed definition and analysis for these terms which are divided into expectations and penalties, target-guided factor, as well as curiosity mechanism module are provided.

5.1 Expectations and penalties

Expectations and penalties in reward function include jumping target, jumping attitude and landing terms, which are used to encourage expected jumping behaviors and punish unexpected or unwanted jumping behaviors of the robot on the basis of the desired task.

5.1.1 In take-off phase

The principal jumping targets are defined as jumping displacement and height. In general, suppose the COM of jumping trajectory of the robot is an approximate parabola, then the jumping performance can be evaluated by two indices, which are characterized by climax and landing position respectively along the direction of robot jumping (shown in Fig. 4).

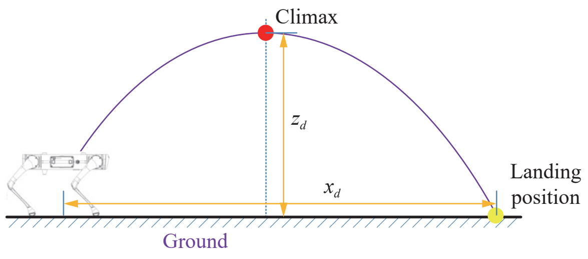


Fig. 4 Desired target setting of robot jumping

Actual jumping position and take-off velocity are both considered to evaluate the basic performance at take-off phase. Position term is directly related to the jumping targets in the horizontal and vertical directions, and velocity term is an important index that determines whether the robot can realize desired jumping targets. First, if the robot performs a forward jump across the desired climax z_d and to the desired landing position x_d, a high reward would be given to this behavior. Thus, the reward should be used to evaluate displacements along both horizontal and vertical directions. The closer to the expected value, the higher reward the robot would get. So, r_{pos} is designed as

$$r_{pos}=-\left(\varsigma_x\mid x-x_d\mid+\varsigma_z\mid z-z_d\mid\right) \tag{11}$$

where $\varsigma_x\geq 0$ and $\varsigma_z\geq 0$ are parameters which decide the priority of displacement and height. It could be adjusted according to training effects.

In addition, linear velocities in horizontal and vertical directions at the instant of take-off that contain torque information of all the joints decide actual jumping displacement and height. The reward about them is defined as

$$r_{vel}=-\left(\varrho_x\| v_x-v_{xd}\|_2+\varrho_z\| v_z-v_{zd}\|_2\right) \tag{12}$$

$\varrho_x \geq 0$ and $\varrho_z \geq 0$ are weight parameters. While the jumping performance is far away from the desired indices, the designed reward would be largely negative. With the increase of training times, the jumping effect tends to be close to the desired indices and approach to zero. The zero point is the desired maximum reward value of the above two rewards.

r_{traj} in reward function about joint motions is a penalty term used to achieve expected jumping locomotion with as few joint position, velocity and torque changes as possible. It also plays a role on increasing control accuracy and reducing energy consumption. It is defined as

$$r_{traj} = -\left(\epsilon_q \sum_{i=1}^{4} \| q_{di} - q_i \|_2 + \epsilon_{\dot{q}} \sum_{i=1}^{4} \| \dot{q}_{di} - \dot{q}_i \|_2 + \epsilon_{\tau} \sum_{i=1}^{4} \| \tau_i \|_2 \right) \tag{13}$$

where ϵ_q, $\epsilon_{\dot{q}}$, ϵ_{τ} are weight parameters of corresponding terms.

Reward about pitch angle is designed to maintain appropriate body posture for avoiding falling or tumbling. Therefore, it is considered at the moment where the robot is leaving the ground. Because the proportion of body mass is much greater than that of legs, COM of the robot can be regarded as unchanged and kept in/near the geometric center of body. At the moment off the ground, pitch angle is directly related to take-off angle. So, $r_{\phi_{tf}}$ is provided as follows:

$$r_{\phi_{tf}} = \begin{cases} 0, & \text{if } 20^\circ \leq \phi \leq 50^\circ \\ \phi - 20, & \text{if } \phi < 20^\circ \\ -\phi + 50, & \text{if } \phi > 50^\circ. \end{cases} \tag{14}$$

5.1.2 In flight-landing phase

In order to increase the control accuracy of joint space of legs in flight-landing phase, reward definition of (13) is also applied. It aims to make the robot optimize jumping attitude in flight time and prepare for landing with small energy consumption. Also, smooth control outputs can be obtained as this term approaches to the desired values.

Referring to zero moment point (ZMP) method[50], while all the feet touch to the ground, COM of the robot should keep in the geometric center of body. Thus, a rectangle plane regarding as support polygon is given and reward is determined by judging whether the actual COM position is in the polygon domain. r_{com_x} is

$$r_{\text{com}_x} = \begin{cases} 0, & \text{if } \bar{x}_r + 0.1 \leq x \leq \bar{x}_f - 0.1 \\ x - (\bar{x}_r + 0.1), & \text{if } x < \bar{x}_r + 0.1 \\ \bar{x}_f - 0.1 - x, & \text{if } x > \bar{x}_f - 0.1 \end{cases} \tag{15}$$

where $\bar{x}_f$ and $\bar{x}_r$ are average values between the thigh and calf joints of forward and rear limbs respectively in horizontal direction. When the robot lands with incorrect attitude like tumbling or overlap of limbs, the COM possibly misses the interval $[\bar{x}_r + 0.1, \bar{x}_f - 0.1]$, and even the interval may not exist. A reward with a large negative value will be given to urge the robot to avoid these situations.

In addition, r_{com_z} is

$$r_{com_z} = - \mid z - z_s \mid \tag{16}$$

where z_s is reference COM of vertical direction in stance phase.

Reward about pitch angle is also used to help the robot obtain proper attitude while touching to the ground:

$$r_{\phi_{fl}} = \begin{cases} 0, & \text{if } -10^\circ \leq \phi \leq 20^\circ \\ \phi + 10, & \text{if } \phi < -10^\circ \\ -\phi + 20, & \text{if } x > 20^\circ. \end{cases} \tag{17}$$

If the above rewards are all satisfied well and second bounding is not considered in the landing phase, desired attitude of the robot is held for landing to the ground and kept in the subsequent time for a successful jump. Landing stability can be guaranteed by the design of ZMP criterion.

5.2 Intrinsic curiosity module with clip

In order to increase exploration efficiency of the RL agent, a curiosity-driven exploration mechanism is introduced in this paper, which aims to help agent search unknown or uncertain states for acquiring more environment information. It plays the roles in accelerating policy learning and avoiding local optimization[49]. Curiosity mechanism originates from instinct of human-beings and animals to spontaneously explore and understand the environment where they live in [51]. When the unknown or deceptive environment information is beyond their current knowledge, instinct of curiosity would drives them to explore and learn in the new environment. For our task, prior knowledge or dataset is not used, and only the form and requirements of the task are specified. Curiosity mechanism can explore new environment information spontaneously when dataset lacks or deceptive information exists, and use these new informations to acquire jumping motion policy by guided learning.

Therefore, based on this idea and referring to [52], an intrinsic curiosity mechanism (ICM) module is introduced to our method to strengthen the exploration efficiency, which aims to help the robot find a set of excellent action sequence to complete the desired jumping motion. First, the original ICM in [52] includes a forward dynamic model and an inverse dynamic model. The forward dynamic model is used to predict environment state $\hat{s}_{i+1}$ in the next step time, and the inverse dynamic model is utilized to predict current action $\hat{a}_i$. Also, due to the property of POMDP in our task, state s is replaced by observation o. The predicted observation $\hat{o}_{i+1}$ is defined as

$$\hat{o}_{i+1} = f_o(o_i, a_i; \theta_o) \tag{18}$$

where f_o is forward dynamic model and θ_o stands for its network parameters. At the same time, the predicted action is written as

$$\hat{a}_i = f_a(o_i, o_{i+1}; \theta_a) \tag{19}$$

where f_a is the inverse dynamic model and the network parameters θ_a are optimized by minimizing its relative loss function. Then, the two loss functions L_o and L_a are both optimized via the form of $\mathcal{L}_2$ norm:

$$L_o(\hat{o}_{i+1}, o_{i+1}) = \frac{1}{2} \parallel \hat{o}_{i+1} - o_{i+1} \parallel_2^2 \tag{20}$$

and

$$L_a(\hat{a}_i, a_i) = \frac{1}{2} \parallel \hat{a}_i - a_i \parallel_2^2 . \tag{21}$$

The intrinsic reward is defined according to the two loss functions. In addition, truncation function clip $(\cdot)$ is utilized to avoid excessively repeated exploration reward and encourage agent to find more novel states and actions. clip $(\cdot)$ can limit the amplitude of reward output in an appropriate interval and avoid too large reward value of curiosity module to affect the overall rewards. The specific form is

$$r_{c,i} = \text{clip}(L_o + L_a, 0, r_{c\max}) \tag{22}$$

where $r_{c\max}$ is the maximum truncation value which is determined according to the overall reward. The overall modified ICM is shown in Fig. 5.

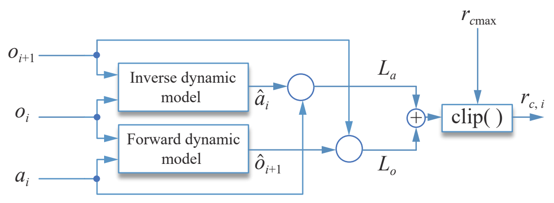


Fig. 5 Modified ICM structure

The introduction of ICM can make the robot pay more attention to the exploration of unknown environment, and measure the novelty discrepancies better between different states and actions in the environment.

5.3 Target-guided factor

Exploring novel states is encouraged while training, and these explored states being task related is also hoped. For training process, stable policy learning must be guaranteed at first, and then ICM carries out exploration for novel states, which is to the advantage of the overall optimization. Ulteriorly, the target-guided factor mechanism proposed in [45] is introduced to the reward r_{tf0} for encouraging RL agent to explore environment where may be closer to completing designed task.

As shown in Fig. 6, target-related exploration behavior p_i^t at the i-th episode can find novel observations along the direction of task completion for accelerating skill learning process. In contrast, random exploration p_i^r behavior would cost more computation and time to search observations and may not converge to an optimization result. Suppose a two-dimension space which is spanned by jumping performance in horizontal and vertical directions. Actual jumping maximum position p_i in the i-th episode is constituted by displacement x_i and height z_i, namely, $p_i = (x_i, z_i)$. The vector angle between position vector $\overrightarrow{p_i p_d}$ of the robot from current maximum position p_i to target indices $p_d = (x_d, z_d)$, and the position vector in last episode $\overrightarrow{p_{i-1} p_d}$ with the last maximum position $p_{i-1} = (x_{i-1}, z_{i-1})$ is used to represent target-guided factor. Also, as a scalar coefficient in desired jumping reward, its purpose is to encourage agent to perform task-related exploration rather than weaken desired reward. So, its range of it must be $(0, 1)$, so the target-guided factor is defined as

$$\lambda = \frac{1}{\pi} \arccos \langle \overrightarrow{p_i p_d}, \overrightarrow{p_{i-1} p_d} \rangle . \tag{23}$$

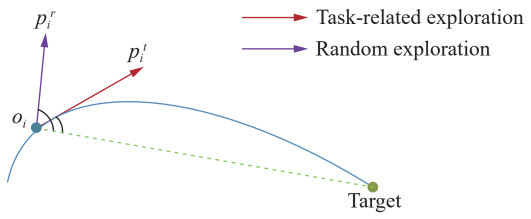


Fig. 6 Effect of target-guided factor

In addition, if the modulus of $\overrightarrow{p_i p_d}$ is in the range of designed tolerance error δ which means current maximum position is very close to the desired target, coefficient $\mu \in (0, 1)$ is used to enhance reward for this jumping process. So, r_{tf0} is replaced by

$$\tilde{r}_{tf0} = \lambda \times \begin{cases} \mu r_{tf0}, & \text{if } \|\overrightarrow{p_i p_d}\| \leq \delta \\ r_{tf0}, & \text{others.} \end{cases} \tag{24}$$

The final reward in take-off phase is written as $r_{tf} = \tilde{r}_{tf0} + \omega_c r_c$.

Algorithm 1 Learning jumping skill by target-guided policy optimization

1) Initialize network parameters θ_{tf}, θ_{fl}, ϕ_{tf}, ϕ_{fl}, temperature coefficients α_{tf}, α_{fl} in the two phases, λ in

```
target-guided module, and $\theta_o$, $\theta_a$ in curiosity networks.
2) for iteration $= 1, 2, \cdots,$ do
3)     Reset robot to initial states
4)     /* Take-off phase */
5)     done = False, step = 0
6)     while not done do
7)         $a_i \leftarrow \pi_{\theta_{tf}}(a_i \mid o_i)$
8)         $o_{i+1} \leftarrow p_{\pi_{\theta_{tf}}}(o_{i+1} \mid (o_i, a_i))$
9)         Collect $r_{traj}$ in take-off phase
10)        if foot force are all 0 then
11)            Collect $r_{pos}$, $r_{vel}$, done = True
12)        end if
13)        Collect experience replay $\mathcal{D}$
14)        step = step + 1
15)        if step $> N$ then
16)            done = True
17)        end if
18)        Update SAC network parameters $\theta_{tf}, \phi_{tf}, \alpha_{tf}$, and
           curiosity network parameters $\theta_o$, $\theta_a$ by ADAM
19)    end while
20)    /* Flight-landing phase */
21)    if step $\leq N$ then
22)        done = False
23)    end if
24)    while not done do
25)        $a_i \leftarrow \pi_{\theta_{fl}}(a_i \mid o_i)$
26)        $o_{i+1} \leftarrow p_{\pi_{\theta_{fl}}}(o_{i+1} \mid (o_i, a_i))$
27)        Collect $r_{traj}$ in flight-landing phase
28)        if foot force are all $> 0$ then
29)            Collect $r_{com_x}$, $r_{com_z}$, done = True
30)        end if
31)        Collect experience replay $\mathcal{D}$
32)        step = step + 1
33)        if step $> N$ then
34)            done = True
35)        end if
36)        Update SAC network parameters $\theta_{fl}$, $\phi_{fl}$, $\alpha_{fl}$
           by ADAM
37)    end while
38)    Compute target-guided coefficients $\lambda$
39) end for
```

6 Training and evaluation

In this section, the proposed controller for the robot model Aliengo is trained in a physical simulator to obtain the desired jumping locomotion. Details of training can be found in Section 6.1, and simulation results, analysis as well as discussion are shown in Section 6.2.

6.1 Training configuration

The simulated environment for verification is ROS melodic and Gazebo physical simulator. The simulated robot Aliengo which is described by unified robot description format (URDF) file would implement control signals in Gazebo obtained from ROS. The network architectures used for actor and critic of SAC in take-off and flight-landing phases are all same, which are parameterized with three-layer neural networks with 512 hidden units on each layer. But for curiosity networks, forward and inverse networks both include a two-layer neural network with 256 hidden units on each layer. Both SAC networks and curiosity networks are activated by ReLU units, and all the networks are optimized by ADAM[53]. Learning rates of actor and critic are 5×10^{-4}, and 1×10^{-3} respectively, as well as forward and inverse networks are both 1E–3. Temperature parameter α in SAC is also trained via the same learning rate as actor. Batch size of the two SAC networks is set as 1 024, and experience replay is designed as 1 000 000 that provides abundant experience for improving exploration sampling and learning performance. Besides, the height of COM z_s is 0.35 m when the robot is in stance. Jumping target indices x_d and z_d are set as 0.4 m and 0.4 m, respectively. Parameters of rewards and policy optimization are shown in Table 2.

Table 2 Parameters of rewards in different terms

Parameter	Value	Parameter	Value	Parameter	Value
ω_{pos}	50	$\epsilon_{q_{tf}}$	1	ω_{com_x}	2
ω_{vel}	1	$\epsilon_{\dot{q}_{tf}}$	1	ω_{com_z}	1
$\omega_{traj_{tf}}$	2	$\epsilon_{\tau_{tf}}$	1	$\omega_{\phi_{fl}}$	1
$\omega_{\phi_{tf}}$	1	r_{cmax}	100	$\epsilon_{q_{fl}}$	2
ς_x	1.5	ω_c	1	$\epsilon_{\dot{q}_{fl}}$	1
ς_z	1	μ	0.2	$\epsilon_{\tau_{fl}}$	0.5
ϱ_x	1	δ	0.15		
ϱ_z	1	$\omega_{traj_{fl}}$	2		

6.2 Simulation results

The designed model takes 22.7 hours as training time in a computer with an i9-7900X CPU @3.30GHz and an Nvidia TITAN RTX GPU. The overall algorithm and training framework can be seen in Algorithm 1.

After training, a policy is obtained to make the robot complete jumping behaviors with the desired jumping height and displacement successfully. The proposed method would be compared to the standard SAC network framework without curiosity module or target-guided factor, and without both of the two modules, which all use the same hyper parameters and network architecture.

6.2.1 Reward analysis

In Fig. 7, learning trends of rewards in take-off phase with different frameworks are given. For our proposed method, it is indicated that the learning curve increases rapidly in 0–1 000 episodes, and then changes gradually to a relatively stable period with oscillations. There are

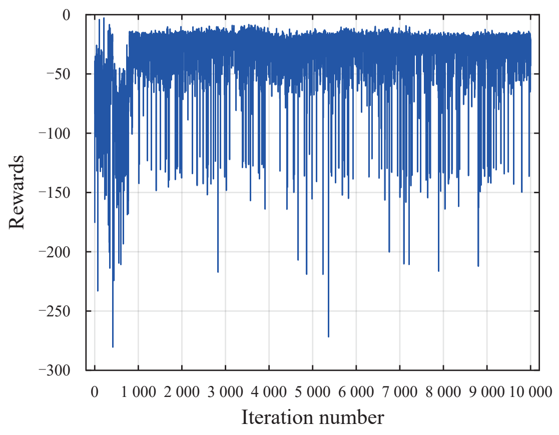


Fig. 7 Learning curve of reward in take-off phase

two possible reasons that lead to oscillations: On the one hand, jumping locomotion includes some moments of sudden motion changes, e.g., the moment while leaving ground. These moments are considered in rewards design for safety and locomotion implementation, but some designs with fixed reward values inevitably bring oscillatory numerical changes in learning curve of rewards. On the other hand, considering that the structural characteristics of a quadruped robot, tiny changes in joint motors may be magnified and make a great impact on balance, stability, or jumping ability. Then, it will be reflected in the numerical changes of reward, i.e, oscillations. In the final episodes, it is noteworthy that there are also a certain amplitude of oscillations. It may be caused by some possible reasons. First, from the perspective of jumping characteristics, fast motion in a short time (about 0.2–0.3 s), indicates its instantaneous and high time-varying properties. Asymptotic convergence in the conventional sense may be difficult to achieve. Second, in the training process, regulating hyperparameters or changing network structure (e.g., increasing network layers and nodes, modifying activation functions, etc.) cannot improve jumping performance on the training curve and the final display effects. Third, jumping performance also has not obvious upgradation by increasing training time and episodes. Thus, it is hard to achieve asymptotic convergence in training process because of time-variation, instantaneity, and nonlinearity at present. Training results with uniform convergence of small-amplitude oscillations can obtain excellent jumping performance to some extent. Similar possibility can be also applicable to the flight-landing phase in the next paragraph.

Fig. 8 shows that the reward output of curiosity module is related to the overall reward change. While the take-off reward is in the increasing stage, the curiosity reward keeps a relatively high value to enhance exploration for helping the robot search more possible observation spaces which contribute to completing jumping task. Then, the take-off reward output inclines to stable, and the curiosity reward reduces to a relatively low value that decreases exploration behaviors. At the same time, oscillations in the overall reward also reflect in the change of curiosity module. While the take-off reward changes violently, curiosity output would increase rapidly to the maximum value for preventing deterioration in the learning process. The remedial behavior always exists while training, since in the overall skill learning process jumping failure or incompletion happens on occasion.

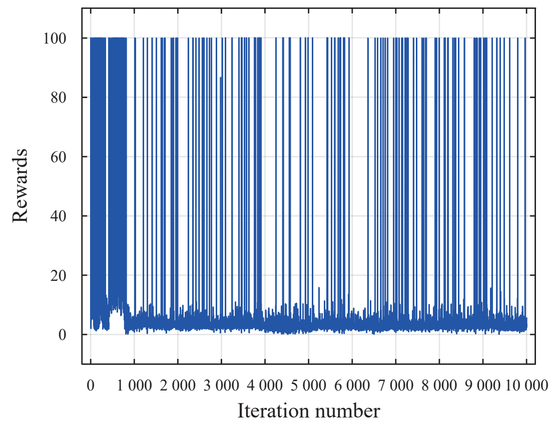


Fig. 8 Reward change of curiosity networks

Learning curve of rewards in flight-landing phase is shown in Fig. 9. It also maintains an increasing trend while learning, and then tends to stable with oscillations in a fixed range. Compared with reward of take-off phase, this reward converges slower obviously because the motion attitude in the flight is depended on excellent take-off attitude and performance. Only with good take-off behavior can the flight attitude be better regulated to the desired jumping performance. Similar to the curve of take-off phase, oscillations also occur in the learning process due to some reward terms and sensitive magnified effects.

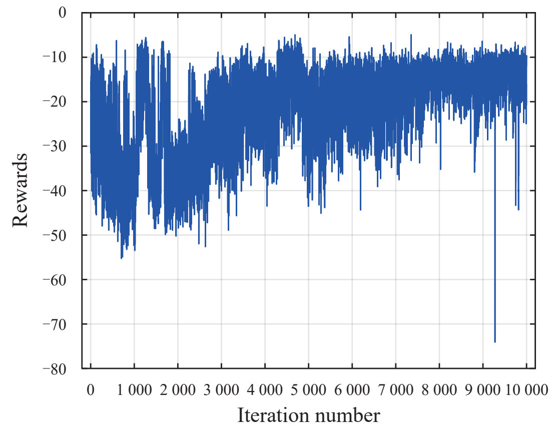


Fig. 9 Learning curve of reward in flight-landing phase

6.2.2 Training process

With the increase of training episodes, jumping performance also tends to the desired targets. The changing trend of jumping performances on maximum horizontal and vertical distances is shown in Fig. 10. The overall training process is divided into four stages equally. In the upper left subfigure, the distribution of maximum horizontal-vertical points is scattered, and different locomotions including jumps in forward or backward directions, falls, slides, flips or others all may occur in this stage. Next, the maximum horizontal-vertical points become concentrated in a relatively small region in the upper right subfigure with range of episodes 2 501–5 000. Fault locomotions like backward jump, and falls are reduced gradually by increasing training episodes. Then, the points region in the lower left subfigure narrows further compared with the previous two subfigures. Concentration degree of points is better and scattered points are less. In the lower right subfigure with episodes 7 501–10 000, the maximum horizontal-vertical points are focused in a small region. Correspondingly, the mean values of these points are the closest to the desired targets. At this stage, success rate of jumping would be high while training.

6.2.3 Ablation comparison

As mentioned above, ablation comparison analysis is provided to verify the effectiveness of our method. Based on the proposed method, target-guided factor or ICM respectively, and both are removed for discussing the effects on different modules in take-off phase. The time consumption of proposed method, without target-guided factor, without ICM, and without both target-guided factor and ICM are 22.7, 22.5, 21.9, and 21.9 hours, respectively. The simulation results are shown in Fig. 11. It is indicated that the pink curve without target-guided factor has higher oscillations than green curve without curiosity module, which may be caused by pursuing novel states and actions so that failing to take advantage of what has been explored. Ulteriorly, the learning performance has no dramatic improvement compared with cerulean curve without both target-guided factor and curiosity module. In contrast, though faster convergence rate is obtained by green curve without curiosity module, higher reward value has been acquired by pink curve. Therefore, combining the two modules is conducive to improving jumping skill learning performance from the perspective of balancing exploration and exploitation.

6.2.4 Jumping tests

After training, the jumping effects are tested by the learned policy fifty times. For ease of showing effects, five results are chosen from the test randomly. In Table 3, the maximum jumping COM displacement and height (eliminating original COM height) which of these tests are given, and with the same policy, repeated jumping tests obtain the similar simulation results. One test example is shown in Fig. 12. Six screenshots of different phases about forward jumping are obtained. The robot realizes a completed jumping locomotion from initial state to the landing state, and maintains a secure foothold on the ground after landing.

Robot COM and pitch angle trajectories while jumping are also given in Figs. 13 and 14. For COM of the robot, it can be found that the trajectories change over time along approximate parabolic curves. All the trajectories are similar in the jumping climax, displacement and implicit time except jump 1. Jumping effects like jump 1 occur occasionally in the test. Performance index jumping height is significantly less than other jumps. According to the property of action selection in DRL, it may be affected by the random terms in networks or small initial parameters change of simulated environment. For pitch angle of the robot, jump 1 still shows a slightly different

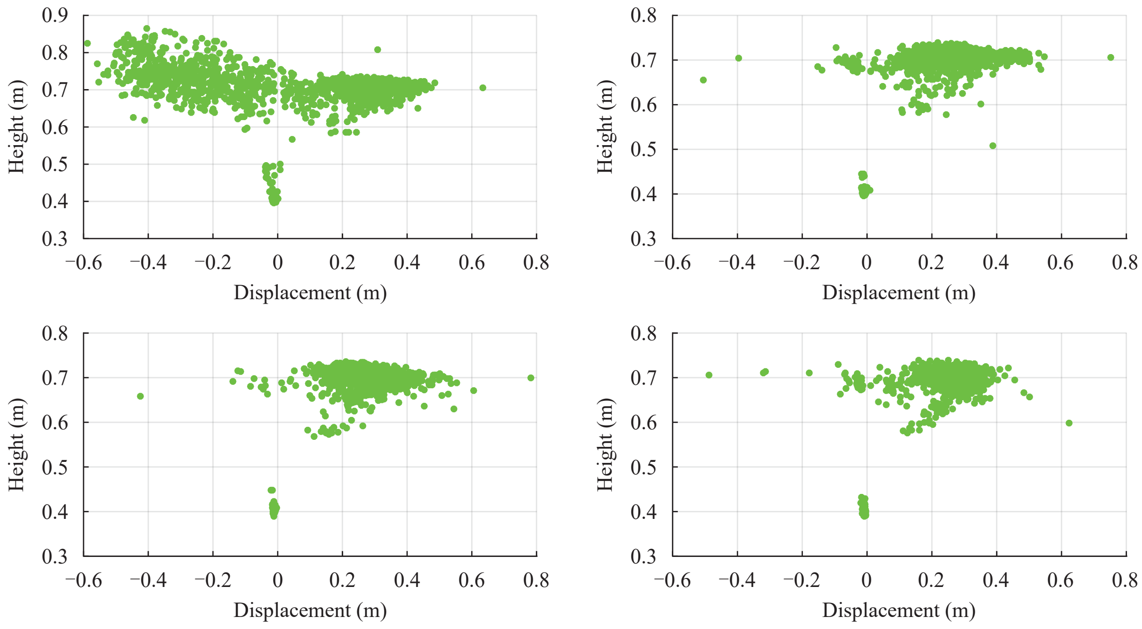


Fig. 10 Actual jumping performance trends in the training process. Upper left: 1–2 500 episodes; Upper right: 2 501–5 000 episodes; Lower left: 5 001–7 500 episodes; Lower right: 7 501–10 000 episodes.

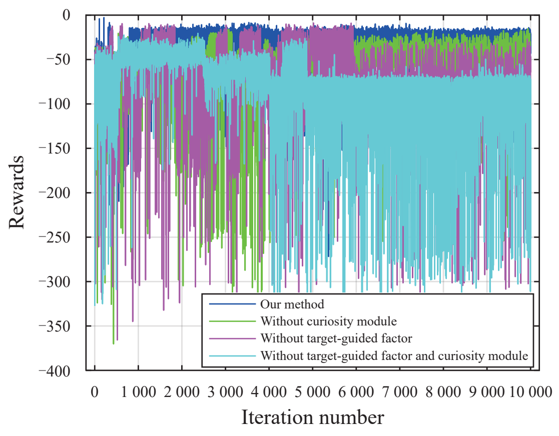


Fig. 11 Ablation comparisons of different methods

Table 3 Jumping performances in tests

Jump order	1	2	3	4	5
Displacement (m)	0.28	0.29	0.29	0.26	0.30
Height (m)	0.33	0.36	0.36	0.35	0.36

change compared with other curves. When the robot is in flight phase, body inclination of the robot is larger than that of other jumps. If using this action sequence, the flight attitude may be slightly different compared with other jumps, but its amplitude indicates that it does not increase the risk of overturning much more. Jump 4 changes relatively smoothly in flight phase, which indicates that pitch angle of the robot body maintains balance to a certain extent. Comparatively, pitch angles of jumps 2, 3, and 5 change with large amplitudes. This phenomenon indicates that due to the instantaneity and rapidity of jumping characteristics, a measure of overshooting and time delay may exist and the final control effects are affected. Random terms in learned model and initialization may also have not negligible effects on the amplitude fluctuation of these jumps. However, though trajectory discrepancies exist, all of these five tests complete forward jumps and stand stably after landing.

Also, in Fig. 14, the final pitch angles of jumps 1–4 do not return to the original states because they are affected by the flight states and landing angles. Subtle differences in the previous states may lead to a certain degree of discrepancies in angle values. Rigorous constraint of pitch angle is not set in reward design since the main aim is to make the robot land safely. Excessively complicated rule may result in reduction of feasible solution space.

It is noteworthy that compared with the desired jumping target, actual jumpings seem to be close rather than completely achieved. From the perspective of trends of maximum horizontal-vertical points in Fig. 10, it is not clear that points will move to a region with a mean value of 0.75 m (COM height of the robot included) in vertical direction. Although the changing points region in horizontal direction covers 0.4 m, it is not sure that if 0.4 m would become the mean value with further learning.

7 Conclusions

In this paper, a model-free control architecture with deep neural networks and reinforcement learning optimization for forward jumping locomotion of quadruped robots is proposed. Soft actor-critic embedded in POMDP is applied as the main reinforcement learning algorithm, and policy optimization is introduced to realize desired jumping targets. Jumping phase analysis is given for designing suitable rewards in different phases. The divided take-off and flight-landing phases are trained by their own learning framework for reducing parameters and raising efficiency. Then, curiosity module is utilized as extrinsic rewards to make the robot explore more observation and action spaces, which contributes to finding more and better possible ways for desired targets. Target-guided factor is collected from observations for giving more rewards while current jumping behavior is closer to target than last time. Moreover, effectiveness of our method is verified by simulation. Completed jumping process is provided with good jumping displacement and height, as well as stable stands after landing.

Some problems also exist in our method and design. Jumping performance is always slightly inferior to the desired targets. It is unlikely helpful to increase training episodes due to the current learning episodes. In addition, training stability also needs to be further improved and

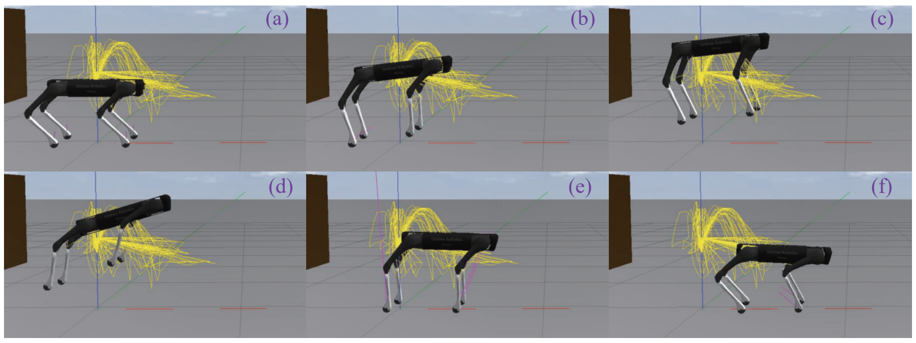


Fig. 12 Jumping process in simulated environment. (a) Stance; (b) Jumping; (c) Off the ground; (d) Flight; (e) Preparing to land; (f) Landing to the ground. Yellow line is trajectory curves that reflects convergence trends by training.

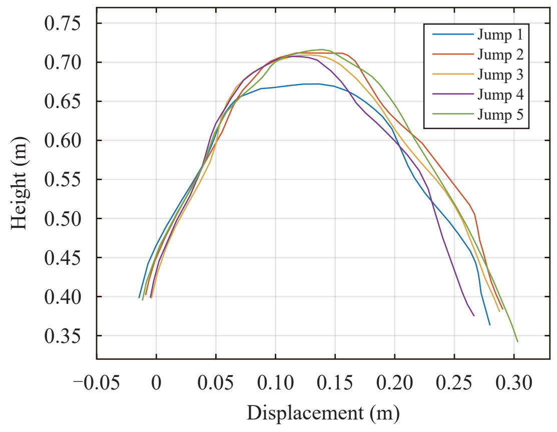


Fig. 13 Robot COM trajectories in jumping process

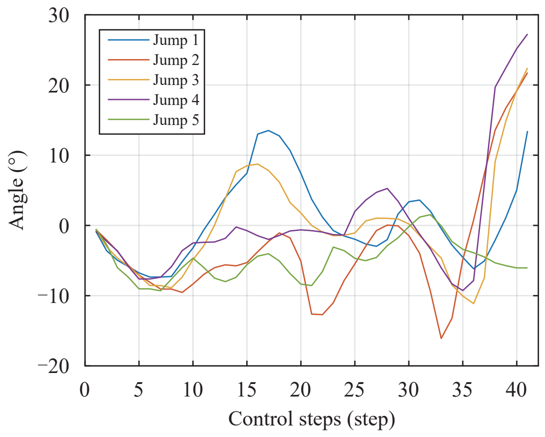


Fig. 14 Robot pitch angle trajectories in jumping process. Control steps mean the sending control signals to the robot. One step is a control period.

reducing oscillations. In future research, these problems will be devoted to solve, and versatility of jumping forms will be enhanced.

Acknowledgements

This work was supported by National Natural Science Foundation of China (No. 61773374) and National Key Research and Development Program of China (No. 2017 YFB1300104).

Declarations of conflict of interest

The authors declared that they have no conflicts of interest to this work.

References

[1] C. T. Richards, L. B. Porro, A. J. Collings. Kinematic control of extreme jump angles in the red-legged running frog. *Kassina maculata. Journal of Experimental Biology*, vol. 220, no. 10, pp. 1894–1904, 2017. DOI: 10.1242/jeb.144279.

[2] J. Z. Yu, Z. S. Su, Z. X. Wu, M. Tan. Development of a fast-swimming dolphin robot capable of leaping. *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 5, pp. 2307–2316, 2016. DOI: 10.1109/TMECH.2016.2572720.

[3] M. Focchi, A. Del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, C. Semini. High-slope terrain locomotion for torque-controlled quadruped robots. *Autonomous Robots*, vol. 41, no. 1, pp. 259–272, 2017. DOI: 10.1007/s10514-016-9573-1.

[4] M. Rutschmann, B. Satzinger, M. Byl, K. Byl. Nonlinear model predictive control for rough-terrain robot hopping. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Vilamoura-Algarve, Portugal, pp. 1859–1864, 2012. DOI: 10.1109/IROS.2012.6385865.

[5] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, S. Kim. Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Madrid, Spain, pp. 7440–7447, 2018. DOI: 10.1109/IROS.2018.8594448.

[6] M. M. G. Ardakani, B. Olofsson, A. Robertsson, R. Johansson. Model predictive control for real-time point-to-point trajectory generation. *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 972–983, 2019. DOI: 10.1109/TASE.2018.2882764.

[7] F. Kikuchi, Y. Ota, S. Hirose. Basic performance experiments for jumping quadruped. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Las Vegas, USA, pp. 3378–3383, 2003. DOI: 10.1109/IROS.2003.1249678.

[8] A. Yamada, H. Mameda, H. Mochiyama, H. Fujimoto. A compact jumping robot utilizing snap-through buckling with bend and twist. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Taipei, China, pp. 389–394, 2010. DOI: 10.1109/IROS.2010.5652928.

[9] C. Gehring, S. Coros, M. Hutter, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, R. Siegwart. 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. DOI: 10.1109/MRA.2015.2505910.

[10] J. Zhong, J. Z. Fan, J. Zhao, W. Zhang. Kinematic analysis of jumping leg driven by artificial muscles. In *Proceedings of IEEE International Conference on Mechatronics and Automation*, Chengdu, China, pp. 1004–1008, 2012. DOI: 10.1109/ICMA.2012.6283387.

[11] Y. K. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Proceedings of IEEE International Conference on Robotics and Automation*, Singapore, pp. 3357–3364, 2017. DOI: 10.1109/ICRA.2017.7989381.

[12] H. B. Shi, L. Shi, M. Xu, K. S. Hwang. End-to-end navigation strategy with deep reinforcement learning for mobile robots. *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2393–2402, 2020. DOI: 10.1109/TII.2019.2936167.

[13] Z. Y. Yang, K. Merrick, L. W. Jin, H. A. Abbass. Hierarchical deep reinforcement learning for continuous action control. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5174–5184, 2018. DOI: 10.1109/TNNLS.2018.2805379.

[14] M. Breyer, F. Furrer, T. Novkovic, R. Siegwart, J. Nieto. Comparing task simplifications to learn closed-loop object picking using deep reinforcement learning. *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1549–1556, 2019. DOI: 10.1109/LRA.2019.2896467.

[15] H. J. Huang, Y. C. Yang, H. Wang, Z. G. Ding, H. Sari, F. Adachi. Deep reinforcement learning for UAV navigation through massive MIMO technique. *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 1117–1121, 2020. DOI: 10.1109/TVT.2019.2952549.

[16] J. Xu, T. Du, M. Foshey, B. C. Li, B. Zhu, A. Schulz, W. Matusik. Learning to fly: Computational controller design for hybrid UAVs with reinforcement learning. *ACM Transactions on Graphics*, vol. 38, no. 4, Article number 42, 2019. DOI: 10.1145/3306346.3322940.

[17] A. Cully, J. Clune, D. Tarapore, J. B. Mouret. Robots that can adapt like animals. *Nature*, vol. 521, no. 7553, pp. 503–531, 2015. DOI: 10.1038/nature14422.

[18] J. Tan, T. N. Zhang, E. Coumans, A. Iscen, Y. F. Bai, D. Hafner, S. Bohez, V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of the 14th Robotics: Science and Systems,* Pittsburgh, USA, 2018. DOI: 10.15607/RSS.2018.XIV.010.

[19] A. Singla, S. Bhattacharya, D. Dholakiya, S. Bhatnagar, A. Ghosal, B. Amrutur, S. Kolathaya. Realizing learned quadruped locomotion behaviors through kinematic motion primitives. In *Proceedings of International Conference on Robotics and Automation*, IEEE, Montreal, Canada, pp. 7434–7440, 2019. DOI: 10.1109/ICRA.2019.8794179.

[20] P. X. Long, T. X. Fan, X. Y. Liao, W. X. Liu, H. Zhang, J. Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *Proceedings of IEEE International Conference on Robotics and Automation*, Brisbane, Australia, pp. 6252–6259, 2018. DOI: 10.1109/ICRA.2018.8461113.

[21] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, S. Levine. Learning to walk via deep reinforcement learning. In *Proceedings of the 15th Robotics: Science and Systems*, Freiburg im Breisgau, Germany, 2019. DOI: 10.15607/RSS.2019.XV.011.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov. Prox imal policy optimization algorithms, [Online], Available: https://arxiv.org/abs/1707.06347, 2017.

[23] Q. Nguyen, M. J. Powell, B. Katz, J. Di Carlo, S. Kim. Optimized jumping on the MIT cheetah 3 robot. In *Proceedings of International Conference on Robotics and Automation*, Montreal, Canada, pp. 7448–7454, 2019. DOI: 10.1109/ICRA.2019.8794449.

[24] G. Bellegarda, Q. Nguyen. Robust quadruped jumping via deep reinforcement learning, [Online], Available: https://arxiv.org/abs/2011.07089, 2020.

[25] N. Rudin, H. Kolvenbach, V. Tsounis, M. Hutter. Cat-like jumping and landing of legged robots in low gravity using deep reinforcement learning. *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 317–328, 2022. DOI: 10.1109/TRO.2021.3084374.

[26] H. W. Park, P. M. Wensing, S. Kim. High-speed bounding with the MIT Cheetah 2: Control design and experiments. *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, 2017. DOI: 10.1177/0278364917694244.

[27] G. P. Jung, C. S. Casarez, J. Lee, S. M. Baek, S. J. Yim, S. H. Chae, R. S. Fearing, K. J. Cho. JumpRoACH: A trajectory-adjustable integrated jumping-crawling robot. *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 3, pp. 947–958, 2019. DOI: 10.1109/TMECH.2019.2907743.

[28] B. Ugurlu, K. Kotaka, T. Narikiyo. Actively-compliant locomotion control on rough terrain: Cyclic jumping and trotting experiments on a stiff-by-nature quadruped. In *Proceedings of IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, pp. 3313–3320, 2013. DOI: 10.1109/ICRA.2013.6631039.

[29] H. W. Park, P. M. Wensing, S. Kim. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. In *Proceedings of Robotics: Science and Systems*, Roma, Italy, 2015. DOI: 10.15607/RSS.2015.XI.047.

[30] T. T. Wang, W. Guo, M. T. Li, F. S. Zha, L. N. Sun. CPG control for biped hopping robot in unpredictable environment. *Journal of Bionic Engineering*, vol. 9, no. 1, pp. 29–38, 2012. DOI: 10.1016/S1672-6529(11)60094-2.

[31] J. Z. Yu, M. Tan, J. Chen, J. W. Zhang. A survey on CPG-inspired control models and system implementation. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 441–456, 2014. DOI: 10.1109/TNNLS.2013.2280596.

[32] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Y. Wang, S. M. A. Eslami, M. Riedmiller, D. Silver. Emergence of locomotion behaviours in rich environments, [Online], Available: https://arxiv.org/abs/1707.02286, 2017.

[33] X. B. Peng, G. Berseth, M. Van De Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics*, vol. 35, no. 4, Article number 81, 2016. DOI: 10.1145/2897824.2925881.

[34] A. Zeng, S. R. Song, S. Welker, J. Lee, A. Rodriguez, T. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Madrid, Spain, pp. 4238–4245, 2018. DOI: 10.1109/IROS.2018.8593986.

[35] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, vol. 4, no. 26, Article number eaau5872, 2019. DOI: 10.1126/scirobotics.aau5872.

[36] X. B. Peng, E. Coumans, T. N. Zhang, T. W. E. Lee, J. Tan, S. Levine. Learning agile robotic locomotion skills by imitating animals. In *Proceedings of the 14th Robotics: Science and Systems*, Corvalis, USA, 2020.

[37] Y. Li, D. Xu. Skill learning for robotic insertion based on one-shot demonstration and reinforcement learning. *International Journal of Automation and Computing*, vol. 18, no. 3, pp. 457–467, 2021. DOI: 10.1007/s11633-021-1290-3.

[38] Z. M. Xie, G. Berseth, P. Clary, J. Hurst, M. Van De Panne. Feedback control for Cassie with deep reinforcement learning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Madrid, Spain, pp. 1241–1246, 2018. DOI: 10.1109/IROS.2018.8593722.

[39] D. O. Won, K. R. Müller, S. W. Lee. An adaptive deep reinforcement learning framework enables curling robots with human-like performance in real-world conditions. *Science Robotics*, vol. 5, no. 46, Article number eabb9764, 2020. DOI: 10.1126/scirobotics.abb9764.

[40] Q. L. Dang, W. Xu, Y. F. Yuan. A dynamic resource allocation strategy with reinforcement learning for multimodal multi-objective optimization. *Machine Intelligence Research*, vol. 19, no. 2, pp. 138–152, 2022. DOI: 10.1007/

s11633-022-1314-7.

[41] Z. Li, S. R. Xue, X. H. Yu, H. J. Gao. Controller optimization for multirate systems based on reinforcement learning. *International Journal of Automation and Computing*, vol. 17, no. 3, pp. 417–427, 2020. DOI: 10.1007/s11633-020-1229-0.

[42] S. X. Gu, T. Lillicrap, I. Sutskever, S. Levine. Continuous deep Q-learning with model-based acceleration. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, New York, USA, pp. 2829–2838, 2016. DOI: 10.5555/3045390.3045688.

[43] X. B. Peng, M. Van De Panne. Learning locomotion skills using deepRL: Does the choice of action space matter? In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Los Angeles, USA, Article number 12, 2016. DOI: 10.1145/3099564.3099567.

[44] N. P. Farazi, T. Ahamed, L. Barua, B. Zou. Deep reinforcement learning and transportation research: A comprehensive review, [Online], Available: https://arxiv.org/abs/2010.06187, 2020.

[45] B. Y. Li, T. Lu, J. Y. Li, N. Lu, Y. H. Cai, S. Wang. ACDER: Augmented curiosity-driven experience replay. In *Proceedings of IEEE International Conference on Robotics and Automation*, Paris, France, pp. 4218–4224, 2020. DOI: 10.1109/ICRA40945.2020.9197421.

[46] C. Banerjee, Z. Y. Chen, N. Noman. Improved soft actor-critic: Mixing prioritized off-policy samples with on-policy experiences. *IEEE Transactions on Neural Networks and Learning Systems*, to be published. DOI: 10.1109/TNNLS.2022.3174051.

[47] S. Qi, W. Lin, Z. Hong, H. Chen, W. Zhang. Perceptive autonomous stair climbing for quadruped robots. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Prague, Czech Republic, pp. 2313–2320, 2021. DOI: 10.1109/IROS51168.2021.9636302.

[48] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, pp. 1856–1865, 2018.

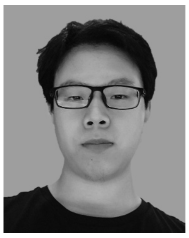
[49] R. S. Sutton, A. G. Barto. *Reinforcement Learning: An Introduction*, Cambridge, UK: MIT Press, 1998.

[50] X. Han, J. Stephant, G. Mourioux, D. Meizel. A ZMP based interval criterion for rollover-risk diagnosis. *IFAC-PapersOnline*, vol. 48, no. 21, pp. 277–282, 2015. DOI: 10.1016/j.ifacol.2015.09.540.

[51] P. Y. Oudeyer. Computational theories of curiosity-driven learning, [Online], Available: https://arxiv.org/abs/1802.10546, 2018.

[52] D. Pathak, P. Agrawal, A. A. Efros, T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Honolulu, USA, pp. 488–489. DOI: 10.1109/CVPRW.2017.70.

[53] D. P. Kingma, J. Ba. Adam: A method for stochastic optimization, [Online], Available: https://arxiv.org/abs/1412.6980v9, 2015.



Chi Zhang received the B.Sc. degree in automation from Shenyang University of Chemical Technology, China in 2015, and the M.Sc. degree jointly in control engineering from Harbin University of Science and Technology, and Institute of Automation, Chinese Academy of Sciences, China in 2018. He is currently a Ph.D. degree candidate in computer applied technology at Institute of Automation, Chinese Academy of Sciences, China.

His research interests include robotics, intelligent control and reinforcement learning.

E-mail: zhangchi2015@ia.ac.cn

ORCID iD: 0000-0001-5527-3362

Wei Zou received the B.Sc. degree from Inner Mongolia University of Science and Technology, China in 1997, the M.Sc. degree from Shandong University, China in 2000, and the Ph.D. degree from Institute of Automation, Chinese Academy of Sciences, China in 2003, all in control science and engineering. He is currently a professor with Institute of Automation, Chinese Academy of Sciences, China.

His research interests include visual control and intelligent robots.

E-mail: wei.zou@ia.ac.cn (Corresponding author)

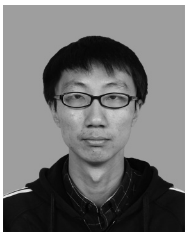
ORCID iD: 0000-0003-4215-5361

Ningbo Cheng received the B.Sc. and M.Sc. degrees in mechanical engineering from Harbin University of Science and Technology, China, in 2004 and 2007, respectively, and the Ph.D. degree in mechanical engineering from Tsinghua University, China in 2012. He is currently an assistant professor with Research Center of Precision Sensing and Control, Chinese Academy of Sciences, China.

His research interests include parallel robot and control.

E-mail: ningbo.cheng@ia.ac.cn

ORCID iD: 0000-0002-7504-4097



Shuomo Zhang received the B.Sc. degree in mechanical engineering from Shandong University, China in 2016, and the M.Sc. degree in control science and engineering from Institute of Automation, Shanghai Jiao Tong University, China in 2020. Currently he is a Ph.D. degree candidate in computer applied technology at Institute of Automation, Chinese Academy of Sciences, China.

His research interests include quadruped locomotion and optimal control.

E-mail: zhangshuomo2020@ia.ac.cn

ORCID iD: 0009-0009-7869-0826