Reward-Adaptive Reinforcement Learning: Dynamic Policy Gradient Optimization for Bipedal Locomotion

Changxin Huang, Guangrun Wang, Zhibo Zhou, Ronghui Zhang, and Liang Lin, *Senior Member, IEEE*

Abstract—Controlling a non-statically bipedal robot is challenging due to the complex dynamics and multi-criterion optimization involved. Recent works have demonstrated the effectiveness of deep reinforcement learning (DRL) for simulation and physical robots. In these methods, the rewards from different criteria are normally summed to learn a scalar function. However, a scalar is less informative and may be insufficient to derive effective information for each reward channel from the complex hybrid rewards. In this work, we propose a novel reward-adaptive reinforcement learning method for biped locomotion, allowing the control policy to be simultaneously optimized by multiple criteria using a dynamic mechanism. The proposed method applies a multi-head critic to learn a separate value function for each reward component, leading to hybrid policy gradients. We further propose dynamic weight, allowing each component to optimize the policy with different priorities. This hybrid and dynamic policy gradient (HDPG) design makes the agent learn more efficiently. We show that the proposed method outperforms summed-up-reward approaches and is able to transfer to physical robots. The MuJoCo results further demonstrate the effectiveness and generalization of HDPG.

Index Terms—Deep reinforcement learning, robotics, bipedal locomotion, hybrid reward architecture

1 Introduction

Bipedal robot locomotion [1] is about controlling a robot to walk on different surfaces with stable walking gaits. This task is challenging because of the complex dynamics involved, especially where undulated surfaces or obstacles are present. Traditional hand-crafted control policies, such as zero moment point (ZMP) [2], hybrid zero dynamics (HZD) [3] and divergent component of motion (DCM) [4], often suffer from limited adaptation to various environments.

Recently, reinforcement learning (RL) has made significant progress in solving complex biped locomotion problems [5], [6], [7], [8]. The core of RL is training robots to take actions that maximize the expected cumulative rewards. A reward function usually consists of multiple components [9], [10], each of which quantitatively describes an aspect of the quality of the walking task, such as body balance maintenance, limb-alteration gait, conservative motor torques, and so on. Most existing approaches simply sum up the component rewards to learn a single value function [5], [10], which may break the correlation between different rewards and therefore limit learning efficiency [11], [12].

In this work, we propose a hybrid and dynamic policy gradient (HDPG) optimization method to address the above-mentioned issues. Specifically, we construct a multi-head critic in the deep deterministic policy gradient (DDPG) [13] framework to capture the gradients separately obtained from multiple rewards. Each head exclusively learns from corresponding reward feedback. Meanwhile, the branch gradients are merged during back-propagation with dynamically updating weights, which aims to guide the policy network to first learn from the "simple" components before the "challenging" ones. The motivation for this dynamic mechanism is that we tentatively guide the robot to give higher priority to learning from components that show fast reward accumulation. For example, the agent is encouraged to learn to maintain body balance before learning to move forward. Intuitively, keeping balance is the prerequisite for moving forward. Thus, the body balance component should have a higher priority than moving forward in the initial learning stage. To achieve this goal, we introduce dynamic weights to allow each component gradient to optimize the policy with a different priority. Therefore, the HDPG agent adaptively learns each reward component during training. The contributions of this work are summaries as follows:

- We introduce a multi-head critic to learn a separate value function for each component reward. The experimental results show that our proposed multi-

- *Changxin Huang and Ronghui Zhang are with the School of Intelligent Systems Engineering, Sun Yat-sen University, Guangzhou 510275, China. E-mail: huangchx53@mail2.sysu.edu.cn, zhangrh25@mail.sysu.edu.cn.*
- *Guangrun Wang is with the Department of Engineering Science, University of Oxford, OX1 2JD Oxford, United Kingdom. E-mail: wanggrun@gmail.com.*
- *Zhibo Zhou and Liang Lin are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China. E-mail: zhouzhib8@mail2.sysu.edu.cn, linliang@ieee.org.*

Manuscript received 24 December 2021; revised 15 September 2022; accepted 7 November 2022. Date of publication 21 November 2022; date of current version 5 May 2023.
This work was supported in part by National Key R&D Program of China under Grant 2021ZD0111600, in part by National Natural Science Foundation of China (NSFC) under Grants U1811463, 61836012, U21A20470, 62006255, 61876224, and 62206314, and in part by GuangDong Basic and Applied Basic Research Foundation under Grants 2017A030312006 and 2022A1515011835.
(Corresponding author: Liang Lin.)
Recommended for acceptance by M. Dudik.
Digital Object Identifier no. 10.1109/TPAMI.2022.3223407

head design outperforms the traditional summed-up-reward methods in biped locomotion tasks.
- We propose dynamic weights for hybrid policy gradients to improve learning efficiency. In this way, the control policy is optimized by hybrid policy gradients in a dynamic manner.
- We build our bipedal robot in the gazebo simulator, forming a non-trivial benchmark to facilitate present and future research on bipedal robots. The benchmark includes three challenges, i.e., *push recovery*, *obstacle*, and *slope terrain*, providing a wide range of walking robustness evaluation environments. The HDPG policy trained with the randomization of dynamics in the simulator is successfully transferred to the physical robot without further tuning.

In addition to the bipedal locomotion tasks, we further conduct experiments on OpenAI gym [14] to verify the generalization of HDPG. Experimental results demonstrate that our HDPG is applicable to more general continuous control tasks to improve learning efficiency. Furthermore, we show that the proposed multi-head critic and dynamic weight can also be applied to another RL algorithms, e.g., proximal policy optimization algorithms (PPO) [15].

2 Related Work

Bipedal locomotion problems were mostly targeted by manually designed policies [16], [17], [18] using methods such as zero moment point (ZMP) [2], [19], hybrid zero dynamics (HZD) [3], [20], [21], divergent component of motion (DCM) [4], [22], and so on. ZMP-based methods [2], [19] use the simplified models, e.g., linear inverted pendulum (LIP) dynamics, to generate walking patterns with the constraints that ZMP should be maintained in the support polygon. DCM-based methods [4], [22] simplify gait generation using the divergent component of the center of mass (CoM), which enlarges the support polygon and allows the robot to recover from external perturbations by step adjustment. HZD approaches [3], [21], [23] impose virtual constraints to put the biped into zero dynamics surface via feedback linearization.

However, the flexibility of these manually designed policies is worrying, especially in adapting to complex and changing environments [24]. Moreover, these manually designed policies rely on having a precise and well-established robot model, and they are inapplicable on the robots where dynamics and kinematics are unknown.

To generalize the walking capability for bipeds to challenging situations and enable the agent to learn the policy without knowing robot dynamics, some model-free methods have been proposed, such as imitation-learning (IL) and model-free RL. IL-based approaches aim to train neural network models supervised by expert policies [5], [6], [25], [26]. But these methods rely on human experts to manually design control policies as references, which can be laborious.

In order to address the aforementioned issues of the traditional controller and imitation learning, researchers attempt to learn bipedal locomotion policies using reinforcement learning (RL) frameworks [27], [28] since RL systems optimize the policies by exploration and interaction with a simulation environment, encouraging favorable movements and punishes improper ones for learning optimal policies. However, [27] and [28] did not demonstrate their effectiveness on physical robots. Recently, Xie et al. [10] formulated a feedback control problem as finding the optimal policy for a Markov Decision Process to learn robust walking controllers that imitated a reference motion with DRL. This proved to be effective in the Cassie simulation. Subsequently, they used an iterative training method and Deterministic Action Stochastic State (DASS) tuples to learn a more robust policy [6]. Siekmann et al. [5] introduced recurrent neural networks (RNNs) to learn memory features that reflected physical properties. Li et al. [24] used a gait library of diverse parameterized motions based on Hybrid Zero Dynamics (HZD) [23], which increased the diversity of behaviors that the robot could learn. The RL agents it adopted could obtain more diverse locomotion behaviors and improved robustness by training to learn a larger variety of potential motions. However, it learned the locomotion policy by imitating reference motions decoded from an HZD-based gait library, which required modeling the robot to design these reference trajectories. Note that all of these approaches use the sum of rewards to learn a single value function, which may be less informative and insufficient to derive effective information from the complex hybrid rewards and limit the learning efficiency of the value functions [11].

Our paper is also related to Hybrid Reward Architecture (HRA) [11], Decomposed Reward Q-Learning (drQ) [29], and DDPG [13] in terms of making full use of the dependency between different reward components to improve learning efficiency. Specifically, we first proposed a multi-head critic to learn a separate value function for each component reward function, similar to HRA [11] and drQ [29]. HRA and drQ first decompose the reward function of the environment into n different reward functions. They aim to learn a separate value function; each of them is assigned a reward component. Learning a separate value function is proven to enable more effective learning. However, both HRA and drQ are based on Q-learning and can only be used for discrete action space tasks. Our HDPG is based on DDPG [13], which allows learning continuous actions. Moreover, we further propose the dynamic weight for hybrid policy gradients to optimize the policy with different priorities. This hybrid and dynamic policy gradient (HDPG) design make the agent learn more efficiently.

3 Preliminaries

In this section, we briefly introduce the background and notations of RL. We also introduce the theory of deep deterministic policy gradient (DDPG) [13] as DDPG is the base model of our proposed method.

3.1 Markov Decision Process (MDP)

Biped locomotion can be formulated as a Markov Decision Process problem. A standard RL includes an agent interacting with an environment E and receiving a reward r at every time step t in MDP. MDP can be denoted as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ is a transition probability function, and $\mathcal{R}$ is a reward function, $\gamma \in (0, 1)$ is a discount factor.

At every time step t, the robot observes the current state $s_t \in \mathcal{S}$ and takes an action a_t according to a policy $\pi : \mathcal{S} \mapsto p(\mathcal{A})$ that maps the states to a probability distribution over the actions. The agent then transits to the next state s_{t+1}, determined by transition probability distribution $\mathcal{P}(s_{t+1}|s_t, a_t)$. Then, the agent will receive a reward $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$ as well as a new state $s_{t+1} \in \mathcal{S}$ from the environment E. The return of a state s_t is the cumulative γ-discounted reward, i.e., $\sum_{t=l}^{T} \gamma^{t-l} r_t$, where T is the horizon of an episode. The objective of RL is to optimize the policy π by maximizing the expected returns from the initial state. According to the *Bellman Equation*, a state-action value function $Q_\pi(s_t, a_t)$ can be defined to describe the expected return after taking an action a_t in state s_t following policy π:

$$Q_\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r_t + \gamma Q_\pi(s_{t+1}, \pi(s_{t+1}))] \tag{1}$$

3.2 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) [13] is a representative model-free RL method. Considering its remarkable performance in continuous control problems, DDPG is utilized as our base model. It consists of a Q-function (the critic) and a policy function (the actor) to learn a deterministic continuous policy. The actor π and the critic Q are approximated with deep neural networks, which are parameterized by θ^π and θ^Q, respectively.

An exploration noise can be introduced to the policy to avoid the agents falling into a local optimum and further improve the exploration efficiency. In particular, the Ornstein-Uhlenbeck (OU) process [30] can be used to generate exploration noise (similar to [13]). Then the output action can be expressed as this: $a_t = \pi(s_t) + \mathcal{N}_t$, where $\mathcal{N}_t \in \mathcal{N}_{OU}$. It utilizes the experience replay strategy that collects the experience tuples (s_t, a_t, r_t, s_{t+1}) stored in a replay buffer B during exploration. In the training stage, the training data are sampled from the replay buffer to optimize the policy. DDPG approximates Q-function by a critic network, which is updated by minimizing the *Bellman* loss:

$$L(\theta^Q) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1} \sim B}[(y_t - Q(s_t, a_t|\theta^Q))^2], \tag{2}$$

where $y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1})|\theta^Q)$ is the target value. Then the actor learns a Q-optimal policy by $\pi(s) = argmax_a Q(s, a)$ and optimizes the policy by using the sampled policy gradients:

$$\nabla_{\theta^\pi} J \approx \mathbb{E}_{s_t, a_t \sim B}[\nabla_a Q(s_t, \pi(s_t)|\theta^Q) \nabla_{\theta^\pi} \pi(s_t|\theta^\pi)]. \tag{3}$$

4 Method

The framework of our method is presented in Fig. 1. The robot observes the current state s_t from the environment, which contains the data from the inertial measurement unit (IMU) and angular positions of all joints. Inspired by "memory-based control" [5], [9], we use state sequence (s_{t-2}, s_{t-1}, s_t) as the input of actor to contain temporal information. The predicted action a_t from the actor contains the control signal about the angular position of joints. It is sent to a minimum jerk planning module [31] to generate their smooth transition trajectories. A low-level PD controller is applied to track these joint position trajectories. After that, the agent will receive a reward r_t from the environment. This process is iterative until termination.

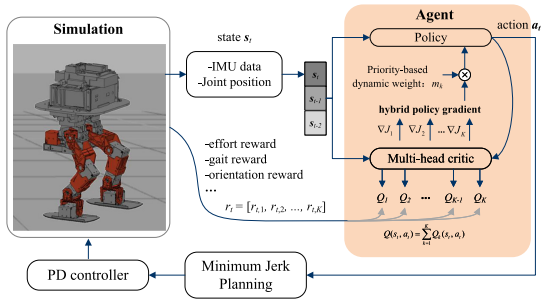


Fig. 1. An overview of our framework. The biped robot interacts with the simulator and obtains experience transitions (s_t, a_t, s_{t+1}, r_t). The multi-head critic of HDPG learns a separate Q-value function for each reward component. The dynamic weight is assigned for the hybrid policy gradient to adjust the learning priority of each policy gradient component.

Learning biped locomotion is a complex task due to multiple constraints involved [32], [33], [34]. When walking forward, the robot needs to maintain the balance of the pelvis and avoid abnormal states, such as *falling down*, *over torque*, and *strange posture*. This usually results in multi-dimensional rewards, each corresponding to a constraint. Existing methods usually use the weighted sum of rewards to learn a single value [5], [6], [9]: $r_t = \sum_{k=1}^{K} w_k r_{t,k}$, where K is the number of rewards and w_k is the weight of each reward. However, simply summing the rewards will lose the dependency between different rewards. Two sets of very different rewards may get the same sum, which results in Q-value not giving insight into factors contributing to policy. For example, when the robot receives a negative reward, it cannot understand whether it is due to the unnatural pose or the over-torque of servomotors.

4.1 Multi-Head Critic for Multiple Values Learning

Inspired by "hybrid reward architecture (HRA)" [11], we conduct a multi-head critic to learn a separate value function for each component reward function (see Fig. 1). Hence, the reward is decomposed and can be expressed as a vector: $r_t = [r_{t,1}, r_{t,2}, ..., r_{t,K}]$. Each head of the critic learns an action-value corresponding to a specific sub-reward. In this way, the overall Q-value is also expressed as a vector:

$$Q(s_t, a_t|\theta^Q) = [Q_{t,1},\ Q_{t,2},\ ...,\ Q_{t,K}] \tag{4}$$

HRA [11] estimate the Q-value based on Deep Q Network (DQN) [35]: $Q_k(s_t, a_t) \leftarrow r_k + \gamma \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1})$. It updates different Q_k with different policies (or $a_{t+1}(k)$), making the learned Q-function inconsistent with the training target. To ensure the learned decomposition is non-trivial, HRA uses the "mean" state-action value as an alternative training target, which is defined as:

$$Q_k(s_t, a_t) \leftarrow r_k + \gamma \sum_{a_{t+1} \in \mathcal{A}} \frac{1}{|\mathcal{A}|} Q_k(s_{t+1}, a_{t+1}) \tag{5}$$

Eq. (5) indicates HRA updates Q_k with the target value resulting from evaluating uniformly random policies under each componential reward function. However, as mentioned in [36], acting greedily *w.r.t.* this Q-function only

guarantees to be better than a random policy and, in general, may be far from optimal. To avoid this problem, Laroche et al. proposed an "empathic" aggregation method that learned a Q-value function no longer by evaluating a random policy but by evaluating the current aggregator's greedy policy *w.r.t.* their local focus [36]. In this work, we build the hybrid reward architecture on the DDPG framework, whose critic evaluates the value of the learned policy. Therefore, the Q-function can be updated by evaluating the practical interaction policy, free from designing an aggregation method and the aforementioned problems. The Q-value from the kth component can be updated as:

$$Q_k(s_t, a_t) \leftarrow r_k + \gamma Q_k(s_{t+1}, a_{t+1}), \tag{6}$$

where the action is obtained from the actual interaction policy: $a_{t+1} = \pi(s_{t+1}|\theta^\pi)$. Eq. (6) indicates that our multi-head critic converges toward the value of the consistent policy. According to Eq. (2), the loss function associated with the multi-head critic is:

$$L(\theta^Q) = \mathbb{E}_{s_t,a_t,r_t\sim B} \sum_{k=1}^{K} [(y_{t,k} - Q_k(s_t, a_t|\theta_k^Q))^2] \tag{7}$$

where $y_{t,k} = r_{t,k} + \gamma Q_k(s_{t+1}, \pi(s_{t+1})|\theta_k^Q)$. Note that we use θ_k^Q to differentiate the parameters of different Q_k. In practice, different θ_k^Q have shared multiple lower-level layers of a critic network. Only the last layer is independent of each other. Then, the overall Q-value is defined as the sum of each Q_k, i.e., $Q(s_t, a_t) = \sum_{k=1}^{K} Q_k(s_t, a_t)$.

4.2 Dynamic Policy Gradient

Since we learn a separate Q-function, the policy gradient in the Eq. (3) can be re-expressed as:

$$\begin{aligned}\nabla_{\theta^\pi} J &\approx \mathbb{E}_{s_t,a_t\sim B}[\nabla_a \sum_{k=1}^{K} Q_k(s_t, \pi(s_t)|\theta_k^Q) \nabla_{\theta^\pi} \pi(s_t|\theta^\pi)] \\ &= \sum_{k=1}^{K} \nabla J_k \end{aligned} \tag{8}$$

where $\nabla J_k = \mathbb{E}_{s_t,a_t\sim B}[\nabla_a Q_k(s_t, \pi(s_t)|\theta_k^Q) \nabla_{\theta^\pi} \pi(s_t|\theta^\pi)]$.This formula indicates that all Q-values update the policy with the same weight. In other words, the agent learns all skills in parallel with the same priority. However, if there are potential dependencies between reward components, then learning all components with the same priority can be inefficient. Moreover, some components are easier to get returns than others. Thus we propose to learn different components/skills with different priorities. We define their priorities by how easily they can get rewards for the current policy, as it is intuitive to encourage the agent to learn the components with easier returns first and then gradually learn the ones with fewer returns.

To achieve this, the agent first utilizes current policy π_T to interact with the environment and obtains N experience samples containing N one-step rewards: $r^n = [r_1^n, \cdots, r_K^n], n \in [1, N]$. All componential rewards are normalized to $(0, 1)$. Then, the mean and the variance of reward samples are calculated: $\mu = [\mu_1, \cdots, \mu_k, \cdots, \mu_K], \sigma = [\sigma_1, \cdots, \sigma_k, \cdots, \sigma_K]$. A larger μ_k means that the kth component is easier to get a larger reward and vice versa. Hence, we give a larger priority weight to the components with a larger μ_k. Besides, we hope to make learning stable, so we increase the priority weight of the unstable components (i.e., those with larger σ_k) to let the robot learn stability as early as possible. Given these, we define the priority weight m_k of each policy gradient component by:

$$m_k = \frac{K(\mu_k + e^{\sigma_k^2})}{\sum_{k=1}^{K} (\mu_k + e^{\sigma_k^2})} \tag{9}$$

Although one could use m_k in Eq. (9) to weight the reward of channel k for priority adjusting (see [37]), this would entangle the update of the Q-value network and leads to unstable training. To cope with this critical problem, we propose using m_k to weight the *policy gradient* rather than the *reward*. This significantly disentangles policy network training from Q-value network training, keeping Q-value network training free from perturbation and ensuring training stability. With m_k weighting policy gradient of channel k to aggregate the hybrid policy gradients, the total policy gradient is dynamically updated by:

$$\begin{aligned}\nabla_{\theta^\pi} J &\approx \mathbb{E}_{s_t,a_t\sim B}[\nabla_a \sum_{k=1}^{K} m_k Q_k(s_t, \pi(s_t)|\theta_k^Q) \nabla_{\theta^\pi} \pi(s_t|\theta^\pi)] \\ &= \sum_{k=1}^{K} m_k \nabla J_k \end{aligned} \tag{10}$$

The priority weights are updated every M episode.

Avoiding Reward Deadlock. Prior works have a scenario where the weight of a reward channel remains low simply because of exploration (i.e., a reward channel has not been seen, and the agent never tries to get into it). Fortunately, our method has two mechanisms to prevent this deadlock. First, similar to DDPG, our method includes exploration noises to prevent our agent from falling into this deadlock. Second, although the visible channels have high weights in the beginning, their rewards gradually converge, and their σ_ks become smaller. This indicates the weights of visible channels become smaller (as a result of Eq. (9)), and the weights of unseen channels become larger, giving the unseen channels a chance to be trained.

4.3 Rewards Design

To make the robots walk with a human-like gait, prior works mainly introduce reference trajectories to guide the learning process or design specific reward functions to encourage robots to walk with a regular gait. Instead of using manually designed reference trajectories, in this work, we exploit some principles of human gait to design the reward functions. Without manually-designed references, careful design of the reward functions is usually required, which tends to result in a complex reward function with many components. In this work, we design a reward function consisting of 6 sub-rewards *w.r.t.* walking forward, walking gait, energy consumption, and robot pose, which are expressed as step reward, gait reward, height reward, torque reward, orientation reward, and fall down reward. The multi-head critic learns the value

function with the reward vector, which is defined as: $r_t = [r_t^g \;\; r_t^s \;\; r_t^f \;\; r_t^h \;\; r_t^o \;\; r_t^d]$

Gait Reward r_t^g. The gait walking is the ultimate goal of the biped robot. The gait occurs when the single and double support periods arise in a time sequence. We formulate this principle as follows:

$$r_t^g = \begin{cases} w_1 d_t^g \cos\theta^g, & if\ a\ gait\ occur \\ 0, & otherwise \end{cases} \tag{11}$$

where d_t^g is the length of the gait, and w_1 is a weight parameter. Assuming T is the time's length of a gait occurs at s time step. θ^g is the angle between the direction of the robot at $(t - T)$ and t. The intention is to encourage the robot to walk forward instead of in other directions.

Step Reward r_t^s. We also introduce a step reward r_t^s to encourage the robot to move forward at each time step. The step reward function is defined as:

$$r_t^s = w_2 d_t^s \cos\theta^s \tag{12}$$

where d_t^s is the move distance at t time step, and w_2 is a weight parameter of step reward. θ^s is the angle between the directions of the robot at $(t-1)$ and t.

Torque Reward r_t^f. The torque reward is inverse proportional to the torque magnitude of the joint motors. A negative torque reward is to punish excessive torque of servomotors while walking. The purpose is to lower walking energy and to make walking trajectories smoother. The same torque reward function has proved to be effective in the Robotschool simulator of OpenAI Gym [14]. We have:

$$r_t^t = -w_3 \sum_{i=1}^{I} |f_t^i| \tag{13}$$

where I is the number of joints, and f_t^i is the torque on the ith joint. Meanwhile, the pose of the robot is one of the important indicators for assessing whether the robot is walking normally, which is evaluated by the height and orientation of the pelvis. The *height reward* and the *orientation reward* are defined as:

$$r_t^h = -w_4|h_t - h_0|, r_t^o = -w_5(|\alpha_t^r| + |\alpha_t^p|) \tag{14}$$

where h_t is the current height of the pelvis, and h_0 is the desired height. In Eq. (14), α_t^r is the roll angle and α_t^p is the pitch angle of the pelvis. Finally, the *fall down reward* r_t^d is set as -50 for the robot that fell to the ground.

5 Bipedal Robot Experiments

This section elaborates on experimental setups and results of bipedal locomotion tasks. We first describe the robot construction and implementation details of HDPG in Sections 5.1 and 5.2, respectively. We then introduce a bipedal motion benchmark for comprehensive evaluation of the proposed method in Section 5.3. The simulation results and hardware results of biped walking experiments are illustrated in Sections 5.4 and 5.5. Experimental results from both simulation and transfer-to-real robots are presented in this video: https://youtu.be/YlznGvG2xAg.

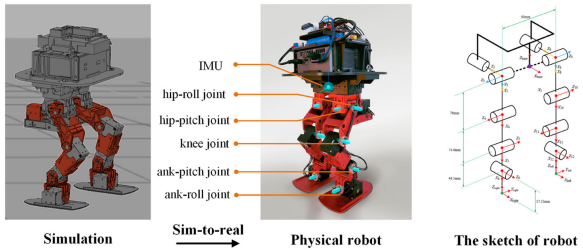


Fig. 2. *Left*: The simulation bipedal robot. *Middle*: The physical bipedal robot. *Right*: The sketch of robot.

5.1 AIDA Robot

To verify the effectiveness of the proposed method, we conduct our simulator, named AIDA, to support the development, training, and validation of biped locomotion. The corresponding physical robot are shown in Fig. 2. The AIDA robot contains 10 degrees of freedom (DoF), and each leg has 5 DoF. The center of mass of the pelvis is maintained at the center of the pelvis, similar to humans.

Simulation. We built our AIDA bipedal robot on Robot Operating System (ROS) framework and Gazebo simulation environment. We use Open Dynamics Engine (ODE) in our simulation, which is the default physical engine of Gazebo. The biped robot is modeled with SolidWork, the sketch is shown in Fig. 2. We will release the AIDA simulator and the code of HDPG later.

Physical robot. The physical robot is equipped with a mini-computer (Intel NUC8i7BEH), an IMU (YIS100-A-DK), 10 servo motors for joint control (DYNAMIXEL XH430-W210-T actuator), and a USB communication converter (U2D2). At testing time, the trained policy is evaluated on NUC in real-time. The policy network predicts appropriate controls based on only the current joint angles and IMU data. The U2D2 receives controls from NUC and transfers them to the motor controllers of the robot.

5.2 Implementation Details

This section presents the implementation details of HDPG. The robot state s_t, as shown in Fig. 1, contains the IMU data and the angular position of each joint. In order to extract temporal features, 3 state frames (s_{t-2}, s_{t-1}, s_t) are concatenated and fed into the policy network to predict an action. The combined states yield 69-dimensional state space (23D for each state). The predicted action is the angular position of each joint, which is a 10-dimensional vector.

Actor. The actor network consists of 3 fully connected layers. The concatenating state (s_{t-2}, s_{t-1}, s_t) passes through 3 fully connected layers of size (69,128), (128,256) and (256,10) to predict an action.

Multi-head critic. Multi-head critic aims to learn separate value functions via a multi-head network, similar to HRA [11]. The action a_t and the state s_t are fed to two fully connected layers of size (10,128) and (69, 128), respectively. They are then concatenated and fed to two fully connected layers of size (256,256) and (256,6) to predict 6 Q-values, each associated with a reward component (see Fig. 1).

Training. The discount factor is set as 0.99. The learning rate of the actor network and critic network are $1e^{-5}$ and $1e^{-4}$, respectively. The training batch size is 64. The priority-based dynamic weight for the hybrid policy gradient is

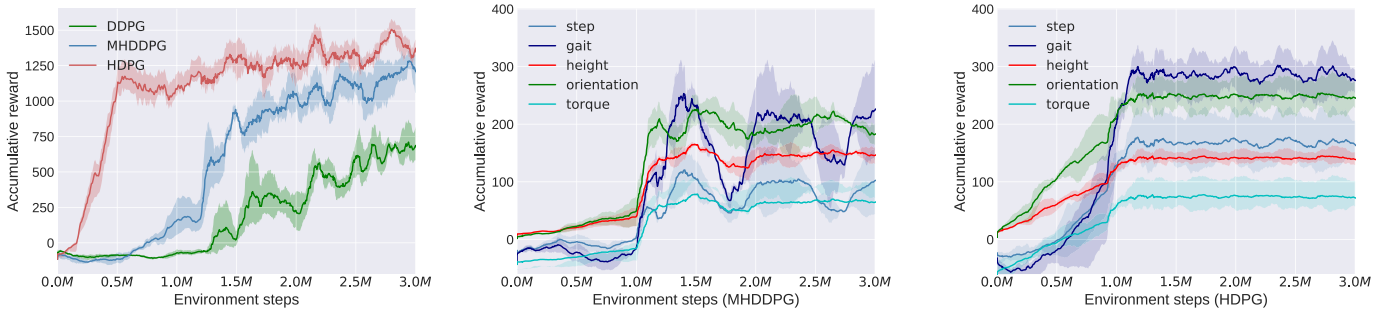


Fig. 3. Comparison of DDPG, MHDDPG and HDPG training curves. *Left*: The comparison of DDPG, MHDDPG and HDPG. Our HDPG achieves a much higher reward than MHDDPG and DDPG. It performs the highest learning efficiency at the same time. *Middle*: The training curve of each reward of MHDDPG. *Right*: The training curve of each reward of HDPG.

updated every 20 episodes. All training uses Adam optimizer [38].

Reward Normalization. The weights of torque reward and step reward are referred to Walker2d of OpenAI Gym [14], where w_1 and w_3 are set as 1.0 and 0.001, respectively. The other weights (e.g., w_2, w_4, w_5) are set as 1.0. Before calculating the dynamic weight, we normalize the reward to ensure that the reward value of each channel is in the same order of magnitude. The max-min normalization is applied to each component. To clarify how to normalize rewards, the gait reward is given as an example. The maximum achievable gait length d^g_{max} is calculated according to the physical limitations. So the maximum and minimum gait rewards are $r^g_{max} = w_1 d^g_{max}$ and $r^g_{min} = -w_1 d^g_{max}$, respectively (according to Eq. (11)). Thus, any gait reward r^g can be normalized to (0,1) by $(r^g - r^g_{min})/(r^g_{max} - r^g_{min})$. This also applies to other rewards.

5.3 Biped Locomotion Benchmark

To comprehensively evaluate the proposed method, we further define a biped locomotion benchmark in simulation that contains 3 walking tasks, which simulate 3 corresponding real-world challenging problems: walking under random disturbances, walking over obstacles, and walking on the slope. The obstacle and slope terrains are modelled with SolidWorks [1] (see Fig. 4). The details of 3 tasks are as follows:

- Walking under random disturbance: the robot walks under disturbances from different directions (see Fig. 4a). The disturbance range is from 6N to 14N, and the duration is 0.2s. The directions contain *forward*, *backward* and *sideward* (either on the left or right side). We measure the success rate that the robot recovers from push disturbances.
- Walking over obstacles: we randomly placed static obstacles of different heights on the ground (see Fig. 4b). We separately measure the success rate of the robot crossing obstacles of different heights.
- Walking on slopes: we build slopes with different angles to evaluate the terrain adaptability of the robot (see Fig. 4c). We separately measure the success rate of the robot climbing slopes of different angles.

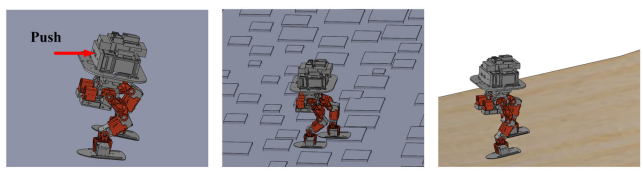
Note that these disturbances, obstacles, and slopes are only defined for the testing phase *without*, but NOT the training phase, i.e., we directly test them *without* any training to verify the robustness and generalization ability of the policy.

5.4 Simulation Results

This section qualitatively and quantitatively analyzes the performance of the proposed HDPG with ZMP[2] method and DDPG [13], for DDPG is the baseline of our method.

- ZMP: ZMP-based algorithm [2], [19] is a traditional classical control method. It has been proven effective in many bipedal walking tasks.
- DDPG: DDPG [13] is an advanced actor-critic model-free RL algorithm. It is also considered the basic model with a single-head critic.
- MHDDPG: MHDDPG represents the DDPG [13] model equipped with the proposed multi-head critic. We keep the experimental settings and parameter settings consistent with DDPG. The only difference is that we learn hybrid policy gradients with multi-head critic, and each policy gradient is assigned the same static weight to optimize the policy (e.g., $m_1 = m2 = ... = m_K = 1$).
- HDPG: HDPG is a full version of the proposed method. It can be seen as MHDDPG incorporated with a dynamic weight for each policy gradient.

We train the bipedal robot with DDPG [13], MHDDPG, and HDPG, respectively (see Fig. 3 (Left) for training curves). The accumulative reward of HDPG starts to rise after training for 0.25 million environment steps, while MHDDPG and DDPG require 1.0 and 1.5 million environment steps, respectively. The performance of MHDDPG is much better than vanilla DDPG [13]. Since vanilla DDPG sums the rewards up to a scalar, it is hard to derive each reward from this scalar. This results in a single Q-value not giving insight into factors contributing to policy, as mentioned in Section 4. Some reward channels are much easier



(a) recover from force (b) obstacle (c) slope

Fig. 4. Illustrations of biped locomotion tasks: (a) walking under random disturbance; (b) walking over obstacles; (c) walking on the slope.

1. https://www.solidworks.com/

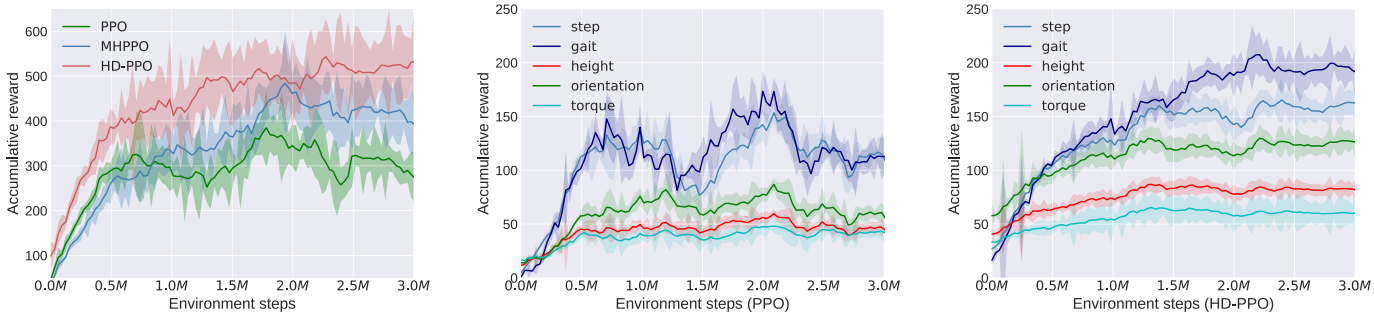


Fig. 5. *Left*: The comparison of PPO, MH-PPO, and HD-PPO. HD-PPO can achieve the best performance, indicating the effectiveness of HDPG architecture in the PPO framework. *Middle*: The training curve of each reward of MH-PPO. *Right*: The training curve of each reward of HD-PPO.

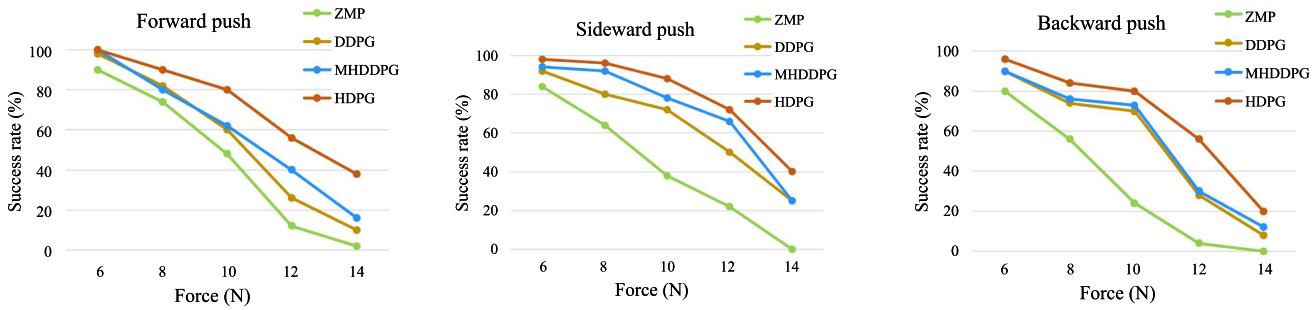


Fig. 6. Comparison of the success rate in the "walking under random disturbance" task. The robot is commanded to walk forward, and random push force will be applied to the robot's pelvis from different directions. We conducted 100 trials for each test. A trial is considered successful if the robot recovers stable walking from push disturbance.

to learn than others [11], then learning a separate value function is more effective. Remarkably, the proposed HDPG outperforms DDPG and MHDDPG. With the utility of dynamic weights, HDPG can adjust the learning priorities of each gradient component and learn more efficiently.

We also apply the proposed multi-head critic and dynamic weight to the PPO baseline to train the bipedal locomotion policy [2]. The training curves of PPO, MH-PPO (PPO with multi-head critic), and HD-PPO (PPO with multi-head critic and dynamic weight) are shown in Fig. 5 (Left). HD-PPO achieves the best performance, indicating that HDPG architecture is also effective in the PPO framework. We note that the PPO agent does not perform as well as the DDPG under the current experimental setting. For example, the cumulative reward for the best testing of PPO is only 400, while DDPG has 750, even better than HD-PPO. In the following bipedal locomotion benchmark tests, we use DDPG as the experimental baseline.

5.4.1 Walking Under Random Disturbance

To evaluate the robustness of the above methods, we let the robot walk under various push disturbances from different directions, as mentioned in Section 5.3. The success rate results are shown in Fig. 6. For 6N disturbances from 3 directions, all algorithms show good resilience. In 8N disturbance 8N from sideward tasks, only HDPG can maintain a success rate of over 90%. In comparison, the success rate of DDPG and MHDDPG declined sharply to about 80%. As the disturbance increases, the success rate of the ZMP-based method drops the fastest and exhibits the worst robustness. For example, when the disturbance increases to 10N, the success rate of ZMP in 3 directions is only 38%, 38%, and 24%, respectively, but the other RL-based methods can maintain a success rate of over 50%. As the disturbance increases to 12N and 14N, HDPG shows greater stability, which is at least about 10% higher than other methods in forwarding tasks. In general, the proposed MHDDPG and HDPG outperform DDPG on all walking under random disturbance tasks, and HDPG achieves the best performance.

5.4.2 walking Over Obstacle

We simulate the obstacle on the ground to evaluate the robustness of different algorithms further. In our experiment, obstacles are randomly placed on the ground. The success rate curve is shown on the left of Fig. 7. Since ZMP agents can hardly walk over obstacles, no comparison is made here. RL-based methods show better adaptability. In this task, HDPG performs the best robustness in walking

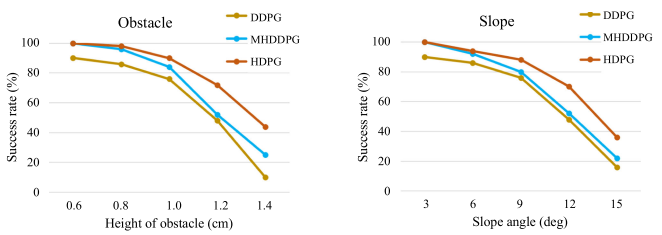


Fig. 7. Comparison of the success rate in the "walking over obstacles" task and "walking on the slope" task. A trial is considered successful if the robot is able to walk stably for 10 seconds. Left: The success rate of robot crossing obstacles with the height from range (0.6,1.4)(cm). Right: The success rate of the robot walking on the slope with the angle from range $(3, 15)$(deg).

2. A brief discussion on stochastic policy is presented in Section 7.

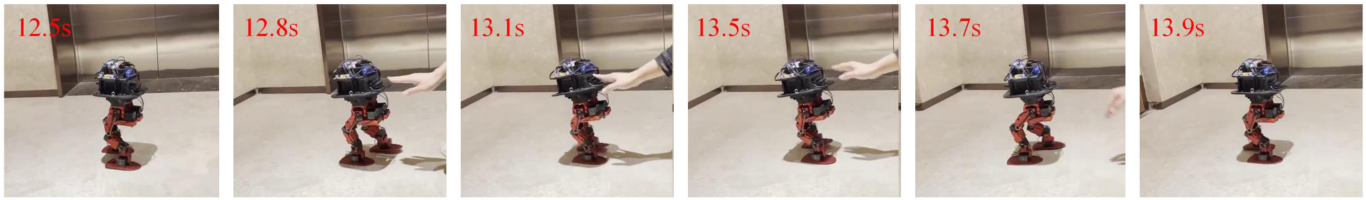


Fig. 8. Simulation to real experiment. The HDPG policy trained with dynamics randomization in simulation is successfully transferred to physical robot. The bipedal robot is able to recover stable walking from the forward push force. Please see the YouTube link for a video version.

over obstacles of all heights, while MHDDPG achieves the second performance. For 0.6cm obstacles, HDPG and MHDDPG are able to maintain a 100% success rate of recovery. For 1.2cm obstacles, only HDPG achieves a success rate over 60%, while MHDDPG and DDPG 50% drop to about 50%. For 1.4cm obstacles, although HDPG has the best performance, its success rate is only about 40%. This is due to the limitation of the height of the robot's leg lift.

5.4.3 Walking on the Slope

Walking on undulating terrain is another challenging task for a biped robot. We test the success rate of walking on the slope for 10 seconds (see Fig. 7 *right*). The performance of DDPG and MHDDPG is close, implying that multi-head critic doesn't have a strong effect on this task. HDPG also has the best performance. When the slope angle is 12 degrees, HDPG is more robust, and its success rate is nearly 20% higher than the other two algorithms.

5.4.4 The Effectiveness of Dynamic Weight

Cumulative rewards of each component. To further reveal the effectiveness of dynamic weights, we recorded the cumulative reward of each reward during training (see Fig. 3, *middle* and *right*). To observe the training characteristic of torque reward, we calculate the average torque reward for each time step and scale it up so it can be visible compared to other rewards. For MHDDPG, the learnings of the *height* and *orientation* rewards are relatively stable. However, the learnings of the step and gait rewards are quite unstable, indicating dependency and mutual effects between different reward components during training. In comparison, HDPG shows stable training of all five components (see Fig. 3, *right*). It is worth noting that its *height* and *orientation* rewards rise at the beginning of training, while its *gait* and *step* rewards then rise at about 0.5 million environment steps. This means that the robot first learns how to keep body balance and then learns to walk forward, proving that priority weights can allow the robot to learn different components more orderly. Similar results are also found for MH-PPO and HD-PPO (see Fig. 3, *middle* and *right*), i.e., the learning of HD-PPO is more stable than that of MH-PPO.

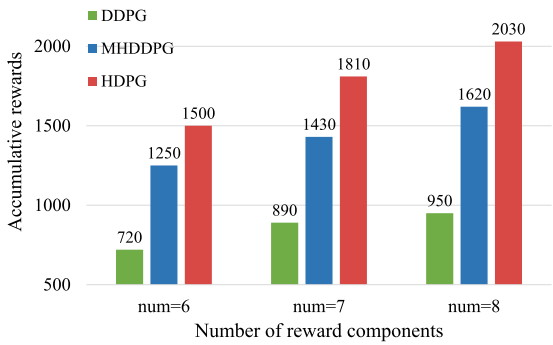


Fig. 9. Results for different number of reward components.

Training With More Complex Rewards. To verify the advantages of HDPG in dealing with complex reward tasks, we add two more components to the original reward settings: *velocity* reward and *contact* reward. The velocity reward aims to encourage the robot to walk at the desired speed, defined as follows:

$$r_t^v = \begin{cases} w_7 v_t, & if\ v_t < v_0 \\ 0, & otherwise \end{cases}, \tag{15}$$

where v_t is the forward velocity at time step t, and v_0 is the desired speed. The contact reward helps to prevent the robot from stomping too hard and is defined as $r_t^c = w_8(f_0 - |f_t^c|)$ when the foot contacts the ground at time step t. f_t^c is the force at the moment when the foot contacts the ground, and f_0 is constant. The experiment of adding velocity reward is marked as num=7, and the experiment of adding both velocity reward and contact reward is marked as num=8. Experimental results are shown in Fig. 9. As the complexity of the reward increases, the advantage of learning a separate value method is more significant. When the number of reward components increases to 7 and 8, the proposed HDPG can also outperform DDPG and MHDDPG.

5.5 Physical Robot Results

The policies trained with dynamics randomization in the simulation were successfully transferred to physical robots. For dynamics randomization, we only randomized the pelvis center of mass in the horizontal directions (i.e., x-axis and y-axis). The results of physical robots are in Fig. 8. The biped robot was disturbed by push force at $13.1s$. Subsequently, the pelvis tilted at $13.5s$. As the policy adjusted the actions, the robot recovered to a stable gait at $13.9s$. This demonstrated the policies trained with HDPG were robust enough to handle the shift from simulation to reality.

6 MuJoCo Experiments

6.1 Experiment Setting

To further verify the generalization of the proposed method, we evaluate HDPG on 3 locomotion tasks on MuJoCo continuous control environment of OpenAI Gym [14], i.e., Walker2d, HalfCheetah, Hopper, and Humanoid. The Hopper and Walker2d tasks contain three reward components,

TABLE 1
Performance on MuJoCo Tasks

Methods	Hopper	Walker2d	HalfCheetah	Humanoid
DDPG	3476.5±98.0	2398.5±1169.7	12422.9±620.4	177.3±77.6
MHDDPG	3539.3±23.9	3162.0±1263.8	12748.9±501.8	513.6±131.0
HDPG	**3600.4±30.5**	3217.5±1144.8	**12895.3±528.1**	601.7±159.1
PPO	2609.3±700.8	3588.5±756.6	5783.9±1244.0	787.2±160.3
HD-PPO	3263.5±367.8	**4688.0±220.1**	6165.2±150.4	**996.4±149.6**

The results show the mean and standard deviation across 10 runs.

while HalfCheetah and humanoid contain two and four reward components, respectively. We conduct MuJoCo experiments using the publicly released implementation repository [3] as the baselines.

6.2 Results

Using the same experimental setting, we compare the performance of the DDPG baselines, MHDDPG, and HDPG. Table 1 shows that MHDDPG and HDPG outperform the DDPG baselines on all tasks within 1M environment steps, and HDPG achieves the best performance on all tasks, implying the proposed "learning a separate value function" and "dynamic policy gradient" can be applied to more general continuous control tasks.

Furthermore, we equip HDPG with proximal policy optimization algorithms (PPO) [15] to verify the generalization of HDPG, denoted as HD-PPO. The best results are highlighted in bold. The hybrid and dynamic policy gradient method significantly improves the performance of DDPG and PPO on all tasks. It is worth noting that HD-PPO achieves a breakthrough of more than 25% on Hopper, Walker and Humanoid tasks, and it increases by about 6.6% on the HalfCheetah task. This demonstrates that the proposed method obtains relatively insignificant improvement in tasks with simple rewards, e.g., HalfCheetah, which only has two components in its reward function. In comparison, the improvement is significant in Hopper, Walker2d, and Humanoid tasks which have more components in the reward functions.

7 Conclusion and Discussion

In this work, we propose a reward-adaptive RL method for bipedal locomotion tasks called hybrid and dynamic policy gradient (HDPG) optimization. It decomposes the commonly adopted holistic polynomial reward function and introduces priority weights, enabling the agent to learn each reward component adaptively. Experimental evaluation illustrates the effectiveness of HDPG by showing better performance on Gazebo simulation in perturbation walking challenges, walking over obstacles challenges, and walking on undulating terrain challenges. With dynamics randomization, the policies trained in the simulation were successfully transferred to a physical robot. In addition, we further verify the generalization of HDPG on 3 MuJoCo tasks. However, the policy trained in the simulator can only enable the physical robot to walk on flat ground and handle small perturbations. For terrains with obstacles and slopes, we believe it is necessary to construct more diverse and refined obstacles and terrains in the simulator to train the agent to achieve sim to real. This will be investigated in our future work. Our future work will aim to transfer the policies trained in the simulation into real physical robots in the challenges of walking across obstacles and undulating terrain.

Discussion. The proposed hybrid and dynamic weights could also be applied to stochastic policies, such as PPO. In that case, the hybrid reward architecture would first learn a separate state value $V(s_t)$ instead of a state-action value $Q(s_t, a_t)$. Based on this, the multi-channel advantage function and the policy gradient of each channel could then be calculated. Finally, dynamic weights could be used to weight these policy gradients to optimize the policy.

3. https://github.com/thu-ml/tianshou

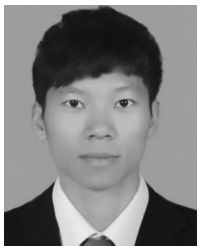
Acknowledgments

The authors would like to thank Jiang Su, Zhihong Zhang, and Dong Zhao for fruitful discussions and valuable comments on this paper.

References

[1] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames, "Dynamic humanoid locomotion: A scalable formulation for HZD gait optimization," *IEEE Trans. Robot.*, vol. 34, no. 2, pp. 370–387, Apr. 2018.
[2] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to Humanoid Robotics*, Berlin, Germany: Springer, 2014, vol. 101.
[3] A. Ames et al., "Dynamic humanoid locomotion: Hybrid zero dynamics based gait optimization via direct collocation methods," Ph.D. dissertation, Georgia Institute of Technology, 2016.
[4] J. Pratt et al., "Capturability-based analysis and control of legged locomotion, part 2: Application to M2V2, a lower-body humanoid," *Int. J. Robot. Res.*, vol. 31, no. 10, pp. 1117–1133, 2012.
[5] J. Siekmann et al., "Learning memory-based control for human-scale bipedal locomotion," 2020, *arXiv:2006.02402*.
[6] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Proc. Conf. Robot Learn.*, 2020, pp. 317–329.
[7] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Hybrid zero dynamics inspired feedback control policy design for 3D bipedal locomotion using reinforcement learning," 2019, *arXiv: 1910.01748*.
[8] T. Li, H. Geyer, C. G. Atkeson, and A. Rai, "Using deep reinforcement learning to learn high-level policies on the atrias biped," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 263–269.
[9] K. Zhang, Z. Hou, C. W. de Silva, H. Yu, and C. Fu, "Teach biped robots to walk via gait principles and reinforcement learning with adversarial critics," 2019, *arXiv: 1910.10194*.
[10] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne, "Feedback control for cassie with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1241–1246.
[11] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5392–5402.

[12] Y. Flet-Berliac and P. Preux, "MERL: Multi-head reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst. Deep Reinforcement Learn. Workshop*, 2019, pp. 1–2.
[13] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Representations*, 2016, pp. 3–4.
[14] G. Brockman et al., "OpenAI gym," 2016, *arXiv:1606.01540*.
[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv: 1707.06347*.
[16] B. Ponton, M. Khadiv, A. Meduri, and L. Righetti, "Efficient multicontact pattern generation with sequential convex approximations of the centroidal dynamics," *IEEE Trans. Robot.*, vol. 37, no. 5, pp. 1661–1679, Oct. 2021.
[17] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1441–1460, Dec. 2018.
[18] H. Dai and R. Tedrake, "Planning robust walking motion on uneven terrain via convex optimization," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots*, 2016, pp. 579–586.
[19] S. Kajita et al., "Biped walking pattern generation by using preview control of zero-moment point," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2003, pp. 1620–1626.
[20] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Hybrid zero dynamics inspired feedback control policy design for 3D bipedal locomotion using reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 8746–8752.
[21] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 876–891, Apr. 2014.
[22] J. Englsberger, C. Ott, and A. Albu-Sch äffer, "Three-dimensional bipedal walking control based on divergent component of motion," *IEEE Trans. Robot.*, vol. 31, no. 2, pp. 355–368, Apr. 2015.
[23] J. W. Grizzle, C. Chevallereau, A. D. Ames, and R. W. Sinnet, "3D bipedal robotic walking: Models, feedback control, and open problems," *IFAC Proc. Volumes*, vol. 43, no. 14, pp. 505–532, 2010.
[24] Z. Li et al., "Reinforcement learning for robust parameterized locomotion control of bipedal robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 2811–2817.
[25] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 661–668.
[26] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "DART: Noise injection for robust imitation learning," in *Proc. 1st Conf. Robot Learn.*, 2017, pp. 143–156.
[27] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "DeepLoco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, 2017.
[28] W. Yu, G. Turk, and C. K. Liu, "Learning symmetric and low-energy locomotion," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–12, 2018.
[29] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez, "Explainable reinforcement learning via reward decomposition," in *Proc. IJCAI/ECAI Workshop Explainable Artif. Intell.*, 2019, pp. 1–4.
[30] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Phys. Rev.*, vol. 36, no. 5, 1930, Art. no. 823.
[31] K. J. Kyriakopoulos and G. N. Saridis, "Minimum jerk path generation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1988, pp. 364–369.
[32] C. R. Gil, H. Calvo, and H. Sossa, "Learning an efficient gait cycle of a biped robot based on reinforcement learning and artificial neural networks," *Appl. Sci.*, vol. 9, no. 3, 2019, Art. no. 502.
[33] A. Xi, T. W. Mudiyanselage, D. Tao, and C. Chen, "Balance control of a biped robot on a rotating platform based on efficient reinforcement learning," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 4, pp. 938–951, Jul. 2019.
[34] X. Wu, S. Liu, T. Zhang, L. Yang, Y. Li, and T. Wang, "Motion control for biped robot via DDPG-based deep reinforcement learning," in *Proc. IEEE WRC Symp. Adv. Robot. Automat.*, 2018, pp. 40–45.
[35] V. Mnih et al., "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[36] R. Laroche, M. Fatemi, J. Romoff, and H. van Seijen, "Multi-advisor reinforcement learning," 2017, *arXiv: 1704.00756*.
[37] M. Fatemi and A. Tavakoli, "Orchestrated value mapping for reinforcement learning," 2022, *arXiv:2203.07171*.
[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.



Changxin Huang received the BS degree in School of Automation Science and Engineering from South China University of Technology, Guangzhou, China, in 2015. He is currently working toward the PhD degree with Sun Yat-Sen University, advised by professor Liang Lin. His current research interests include reinforcement learning and robotics.

Guangrun Wang received the two BE degrees and one PhD degree from SYSU, in 2014 and 2020. He is currently a Postdoctoral Researcher in the Department of Engineering Science with the University of Oxford. He was a visiting scholar with the Chinese University of Hong Kong (CUHK). His research interest is representation learning. He is a Distinguished Senior Program Committee member for IJCAI, an outstanding reviewer of ICLR, NeurIPS, and ICCV. He is the recipient of the 2018 Pattern Recognition Best Paper Award, two ESI Highly Cited Papers, Top Chinese Rising Stars in Artificial Intelligence, and Wu Wen-Jun Best Doctoral Dissertation.

Zhibo Zhou received the BS degree in information and computing science from XiaMen University. He is currently working toward the master degree in computer science and technology in Sun Yat-sen University, advised by professor Liang Lin. His research interests include reinforcement learning and robotics.

Ronghui Zhang received the PhD (Eng) degree in Mechanical & Electrical Engineering from the Changchun Institute of Optics, Fine Mechanics and Physics, the Chinese Academy of Sciences, Changchun, China, in 2009. He is currently an associate professor with Sun Yat-sen University, China. His current research interests include computer vision, intelligent control, and ITS. He has published more than 20 papers in international journals.

Liang Lin (Senior Member, IEEE) is a full Professor with Sun Yat-sen University. He served as the Executive R&D Director and Distinguished Scientist of SenseTime Group from 2016 to 2018, taking charge of transferring cutting-edge technology into products. He has authored or co-authored more than 200 papers in leading academic journals and conferences with more than 12,000 citations. He is an associate editor of IEEE Trans. Human-Machine Systems and IET Computer Vision. He served as Area Chairs for numerous conferences such as CVPR, ICCV, and IJCAI. He is the recipient of numerous awards and honors including Wu Wen-Jun Artificial Intelligence Award, CISG Science and Technology Award, ICCV Best Paper Nomination, in 2019, Annual Best Paper Award by Pattern Recognition (Elsevier) in 2018, Best Paper Diamond Award in IEEE ICME 2017, Google Faculty Award in 2012, and Hong Kong Scholars Award, in 2014. He is a Fellow of IAPR and IET.