2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3503-7770-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IROS58592.2024.10802725

Learning Safe Locomotion for Quadrupedal Robots by Derived-Action Optimization

Deye Zhu[1], Chengrui Zhu, Zhen Zhang, Shuo Xin, Yong Liu*

***Abstract*— Deep reinforcement learning controllers with exteroception have enabled quadrupedal robots to traverse terrain robustly. However, most of these controllers heavily depend on complex reward functions and suffer from poor convergence. This work proposes a novel learning framework called derived-action optimization. The derived action is defined as a high-level representation of a policy and can be introduced into the reward function to guide decision-making behaviors. The proposed derived-action optimization method is applied to learn safer quadrupedal locomotion, achieving fast convergence and better performance. Specifically, we choose the foothold as the derived action and optimize the flatness of the terrain around the foothold to reduce potential sliding and collisions. Extensive experiments demonstrate the high safety and effectiveness of our method.**

I. INTRODUCTION

Quadrupedal locomotion controllers [1]–[5] have made significant progress in recent years. Unlike wheeled robots, quadrupedal robots can autonomously select footholds and adaptively traverse various challenging terrains. In complex application scenarios such as patrols and deliveries, robots need to carefully control their feet to navigate through various staircases and steps safely, preventing dangerous behaviors such as getting stuck and falling.

Most existing methods construct the robot-centric elevation map with LiDAR or cameras as the form of exteroception [5]–[7]. Model-based approaches [8]–[10] explicitly select safe footholds from the elevation map and generate joint commands to achieve the desired footholds. However, model-based methods can only solve some corner cases, and model mismatch or violation of assumptions leads to poor robustness [7]. Controllers based on model-free reinforcement learning (RL) are suitable for addressing such a problem.

Incorporating terrain observation in learning-based controllers leads to better dynamic responsiveness. Some learning-based methods [11]–[13] directly use raw sensor data like depth images as environmental perception. The disadvantage is that it is usually expensive to obtain dense depth during simulated training. Using height sampling to describe the terrain, Rudin et al. [14] trained thousands of robots in parallel on different terrains, obtaining strong terrain traversal capabilities but still prone to dangerous foot collision behavior. This issue arises because the RL policy does not have a fine-grained perception of the terrain, suggesting that the neural network does not fully understand the terrain information.

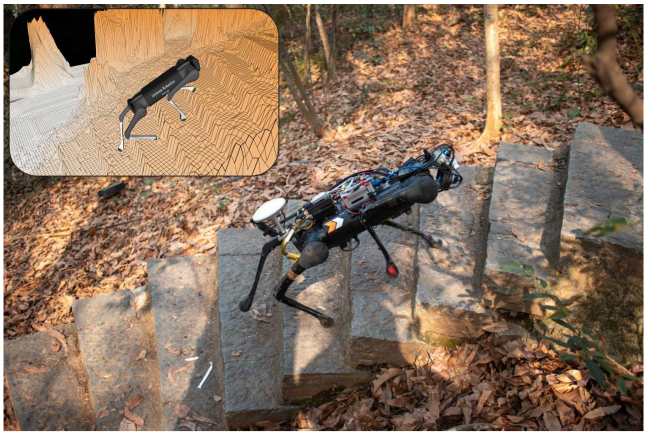

Fig. 1. Aliengo travels outdoors, with the robot autonomously selecting safe and reliable contact positions. The terrain consists of steps of varying widths and heights. Uneven surfaces and irregular steps pose a challenge for Aliengo.

Some methods [15], [16] introduce actions output by an external planner as a reference to shorten the training time and improve the stability of the policy. Jenelten et al. [17] introduced a model-based trajectory optimization method into the RL training process to achieve adaptive foothold locomotion in complex terrain. For a simpler implementation of safe locomotion, Shi et al. [18] introduced a discrete safety reward function in reinforcement learning. The policy's training speed is improved by introducing an adjustable trajectory generator. Meanwhile, the learning-based policy has also been proven to have the ability to generate footstep plans [19], which reveals the great potential of the policy for planning macro-actions. Research on how to improve policy understanding of terrain and learn more efficient and safe locomotion is crucial.

With prior knowledge, animals can select safe footholds ahead of time to avoid collision in challenging terrain. During locomotion, the muscle-generated action and expected footholds are alternately optimized. Inspired by this process, we propose a **derived-action optimization** framework to solve the problem that learning-based methods have encountered. In this framework, we define derived action as a high-level representation of policy and introduce a derived-action reward function to reinforcement learning. This allows for the alternating optimization of both action and derived action, which speeds up policy convergence and enables safe locomotion in complex terrain.

Our contributions are as follows:

Deye Zhu, Chengrui Zhu, Zhen Zhang, Shuo Xin and Yong Liu are with the College of Control Science and Engineering, Zhejiang University, Hangzhou, 310027 China {zhudy, jewel, zhenz, 22232036}@zju.edu.cn, yongliu@iipc.zju.edu.cn
Yong Liu is the corresponding author.

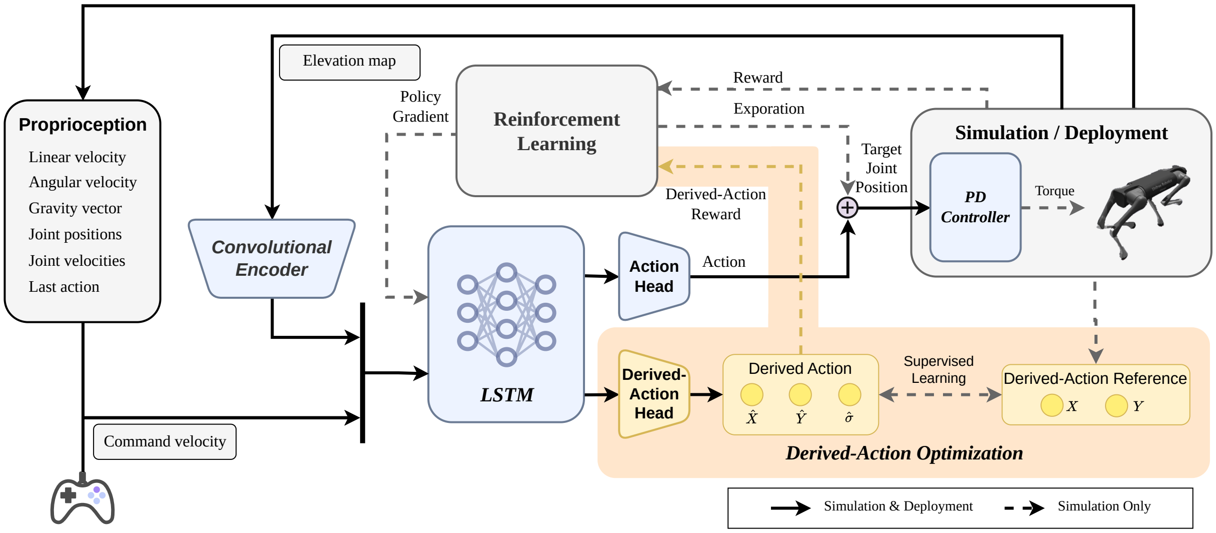

Fig. 2. An overview of training method and deployment. The policy takes proprioception and elevation map as inputs and outputs joint action as well as derived action. Supervised learning is used to train the derived action head, and then the derived-action reward function is introduced to accelerate the convergence of the policy.

- We propose a novel framework for policy training. The derived action is explicitly represented by the policy and participates in the optimization of RL.
- We choose the foothold as the derived action to learn safe locomotion. By optimizing the terrain flatness around the foothold, the policy automatically selects safe footholds to reduce unintended collisions.
- The effectiveness of our method is demonstrated in simulation and real-world experiments. Compared to baseline methods, our proposed method converges faster and achieves fewer collisions in challenging terrains.

II. METHOD

A. Problem Formulation

Quadrupedal locomotion and control can be defined as a Markov Decision Process (MDP), which is defined by a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{R}, P >$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition probability distribution, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function. At time t, the agent gets a state s_t from the environment and selects an action a_t. Then environment transits to the next state s_{t+1} with the probability $P(s_{t+1}|s_t, a_t)$ while earning reward $r_t = \mathcal{R}(s_t, a_t)$. Given the discount factor $\gamma \in [0, 1)$, its objective is to find a parameterized policy $\pi^*(a_t|s_t)$ that maximizes the expected long-term reward:

$$\pi^*(a_t|s_t) = \arg\max_{a_t} \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t] \tag{1}$$

In our derived-action optimization framework, a MDP can be represented by a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{A}^d, \mathcal{R}, P >$, where $\mathcal{A}^d$ is the derived-action space and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{A}^d \mapsto \mathbb{R}$ is the reward function. At time t, the policy also emits a derived action $a_t^d \in \mathcal{A}^d \sim \pi(\cdot|s_t)$ along with the action a_t, and the reward is calculated by $r_t = \mathcal{R}(s_t, a_t, a_t^d)$.

B. Policy Training

We trained a policy network to accomplish the task of crossing challenging terrain. As shown in Figure 2, the policy network combines the robot's proprioceptive and exteroceptive information, as well as the main action head and derived-action head of the policy output both joint action and derived action. The joint action is used to control robot locomotion and participate in the interaction with the simulation environment. The derived action is associated with the action through probabilistic supervised learning [20]. It is used to enrich the reward function and accelerate the convergence of the policy and the locomotion effect. Finally, the policy network is transferred to a physical robot with [21], [22].

1) State, Action and Derived-Action Spaces: The state is defined as $s_t = (s_t^p, s_t^e) \in \mathbb{R}^{658}$, where $s_t^p = [v_t \ \omega_t \ g_t \ c_t \ \theta_t \ \dot{\theta}_t \ a_{t-1}^j]$ refers to the proprioception, where $v_t \in \mathbb{R}^3, \omega_t \in \mathbb{R}^3, g_t \in \mathbb{R}^3, c_t \in \mathbb{R}^3, \theta_t \in \mathbb{R}^{12}, \dot{\theta}_t \in \mathbb{R}^{12}, a_{t-1} \in \mathbb{R}^{12}$ denote body linear velocity, body angular velocity, gravity vector, velocity commands, joint positions, joint velocities, and the previous action, respectively. The exteroception s_t^e is a robot-centric elevation map, which is a rectangular area of 1m×1.4m with a resolution of 0.05m.

The action $a_t \in \mathbb{R}^{12}$ is defined as the target positions of the robot's 12 joints, which are then used to calculate the joint torques by the proportional-derivative (PD) controller. The derived action a_t^d is chosen as the distribution of the foothold of each leg at the moment of landing. At time t, it is assumed that the foothold of leg i follows a Gaussian distribution $\mathcal{N}((x_{i,t}, y_{i,t}), \sigma_{i,t}^2)$, where $(x_{i,t}, y_{i,t})$ denotes the expected foothold of leg i expressed in the robot coordinate system, and $\sigma_{i,t}$ denotes the standard deviation of the foothold.

2) Probabilistically Supervised Learning of the Derived Action: As shown in Figure 3, the locomotion process of one foot can be divided into two phases: the contact phase and

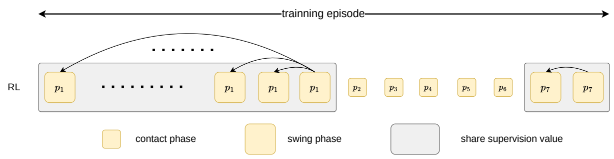

Fig. 3. The foot locomotion process can be divided into two phases. The small box indicates that the foot is in contact with the ground, while the large box indicates that the foot is airborne. The grey box indicates that the derived action shares a supervised value.

the swing phase. In the contact phase, we take the position and variance of the foot in contact with the ground as the derived action. In the swing phase, we take the position and variance of the contact moment in the future as the derived action. More intuitively, Shared supervised value ensures the derived action always focuses on the safety of the footfall, which can optimize the policy more clearly and efficiently.

The derived-action head of the policy is trained using probabilistic supervised learning. For instance, when crossing stairs, a foot may get stuck because of colliding with the vertical surface of a step or stepping up to the next one. Thus, the derived action represents this uncertainty through variance. We define the loss of the derived action prediction as follows:

$$L = \frac{\|y - \hat{y}\|^2}{2\hat{\sigma}^2} + \log\left(\hat{\sigma}^2\right) \tag{2}$$

where y denotes the actual position of the foothold, $\hat{y}$ denotes the mean position of the derived action, and $\hat{\sigma}^2$ denotes the standard deviation of the derived action. Based on the Mean Squared Error (MSE) loss, this approach introduces variance that allows the policy to learn a probability distribution for each foothold rather than just an expected position. The variance will increase significantly at the beginning of the swing phase, and the foot contact location becomes fuzzy due to the uncertain foothold. Towards the end of the leg swing phase, the variance will decrease rapidly and the foot contact location becomes precise.

3) Reward Function: As shown in Equation 3, in reinforcement learning rewards, we categorize the rewards into ordinary rewards r_i^j and derived-action rewards r_i^d, where i is the ordinal index of the reward function and ω denotes the reward function coefficients. The ordinary reward function r_i^j follows the setup of the work [14]. The primary rewards are accurate velocity tracking rewards and rewards that can increase the naturalness of robot locomotion, such as joint velocity, acceleration, smoothness, and so on. The derived-action reward function is shown in Equation 4:

$$r_t(s_t, a_t^j, a_t^d) = \sum \omega_i^j r_i^j(a_t^j) + \omega_i^d r_i^d(a_t^d) \tag{3}$$

$$\begin{gathered} r^d = \tfrac{1}{\exp(k_\sigma \hat{\sigma})} \iint_S p(x,y) V(x,y) \mathrm{d}x\mathrm{d}y, \\ (x,y) \sim \mathcal{N}((\hat{x}_{i,t}, \hat{y}_{i,t}), \hat{\sigma}_{i,t}^2) \end{gathered} \tag{4}$$

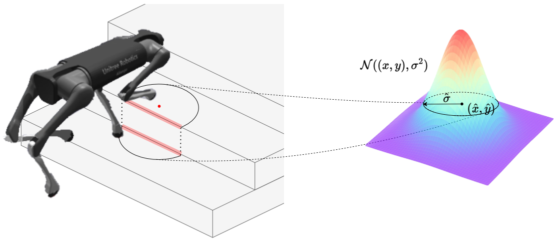

Fig. 4. Representing the foothold safety as an integral of terrain security and predicted foothold probability.

where $V(\cdot)$ is an evaluation function that computes the flatness of terrain at a point on the map. We select the variance of terrain height around the foothold as the evaluation function. As shown in Figure 4, the closer the foothold is to the edge of the step, the greater the variance, indicating a higher risk. Meanwhile, compared to calculating the distance from the foothold to the edge of the step, this evaluation function is more accessible to calculate in parallel. $p(\cdot)$ is the probability density of a two-dimensional Gaussian distribution centered on the derived action $(\hat{x}_{i,t}, \hat{y}_{i,t})$, with $\hat{\sigma}$ as the variance. The derived-action reward is the integral of the terrain safety value and represents the average foothold safety under consideration of the foothold probability. When the robot lifts its foot, the derived action shifts from the current foothold to the next. Therefore, the mean of the derived action will move to the next point, and the variance will become extremely large. To reduce the position bias and excessive variance error, we multiply the function by a variance decay coefficient, paying more attention to assessing footholds during the swing and the contact phase. It can be seen that the reward function designed with derived action can generate a continuous reward signal to optimize the derived action without interruption. Thus, the policy is optimized at a fast speed of convergence.

4) Curriculum Learning: We use a game-inspired curriculum [14] to ensure progressive learning on difficult terrain. Terrains include flat terrain, rough terrain, discrete steps terrain, continuous stair terrain, sloping terrain, and gap terrain. Rough terrain is a wave-shaped terrain with grid sides of 0.05 m and heights uniformly sampled between [-0.05, 0.05] m. Discrete step terrain consists of a group of rectangular squares, with heights uniformly sampled between [0.05, 0.25] m. On continuous stair terrain, we sample step widths from $\{0.25, 0.3, 0.35, 0.4\}$ m, and sample step heights from [0.05, 0.20] m to make the policy more robust. The slope of the sloping terrain was sampled between $[5°, 30°]$. The width and depth of the gaps of gap terrain are sampled between [0.0, 0.20] m. The proportion of continuous stairs is higher because it is common and challenging, while that of other terrains is roughly the same.

5) Domain Randomization: We randomize the mass of the robot's body and legs, the joint PD parameters, the initial joint positions and velocities, and the initial body positions for each episode to increase the robustness of the policy. The friction coefficient is also randomized to enhance adaptation to different surfaces. The parameters of domain randomization follow the settings of Rudin *et al.* [14].

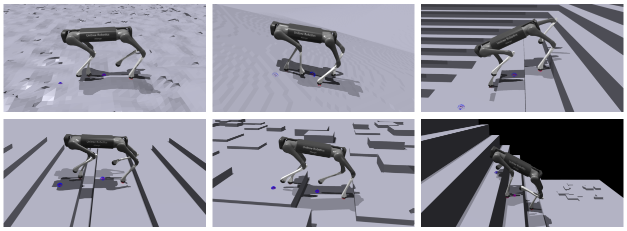

Fig. 5. The terrain settings for IsaacGym include flat terrain, rough terrain, sloping terrain, continuous stair terrain, gap terrain, and discrete steps terrain.

III. EXPERIMENTS

To evaluate the effectiveness of derived-action optimization, we compare the following three methods:

1) Baseline [14]: The policy is trained with regular reward functions, which do not include terms that constrain footholds.

2) DR (Discrete Reward): Based on the *Baseline*, the reward functions include a discrete form of Equation 4 , which is only valid for the first contact moment.

3) DAO (Derived-Action Optimization): The policy is optimized by the derived-action optimization framework.

Other experimental settings remain consistent across the three methods.

A. Policy Training in Simulation

We implement the environment with IsaacGym [23]. The policy network is trained by Proximal Policy Optimization (PPO) [24]. The policy network consists of four parts: a convolutional encoder for encoding elevation maps, a Long Short-Term Memory (LSTM) to keep historical encodings, an action head, and a derived-action head. The convolutional encoder consists of a two-layer Convolutional Neural Network (CNN) with channels of [8, 16] and a three-layer Multi-Layer Perceptron (MLP) with neurons of [128, 64, 32]. Both the action head and derived-action head are MLPs with three hidden layers with [512, 256, 128] neurons.

The policies are trained in 1024 parallel environments for 20,000 iterations and updated every 24 steps. The terrain settings for the training are shown in Figure 5. All training is performed on a desktop PC equipped with an Intel Core i7-12700KF CPU, 64 GB RAM, and an NVIDIA RTX 3090 GPU. The algorithms are trained for 16 hours.

B. Behavior of Derived Action

We demonstrate the variation of the derived action on a continuous staircase. As shown in Figure 6, at time t_1, the robot's front right foot is just lifted, and the derived action indicates the estimated location of the front right foot and the distribution of the possible foothold. The large variance indicates the considerable uncertainty of footfall at this moment. At time t_2, the robot's front right foot is about to touch the ground, and the actual foothold is close to the derived action. The decrease in variance indicates that the foothold estimated by the policy is becoming more accurate. At time t_3, the front right foot of the robot completely lands on the ground, and the derived action converges to the actual contact point with a small variance. Changes in the position and variance of the derived action suggest that the policy can accurately represent its high-level behavior.

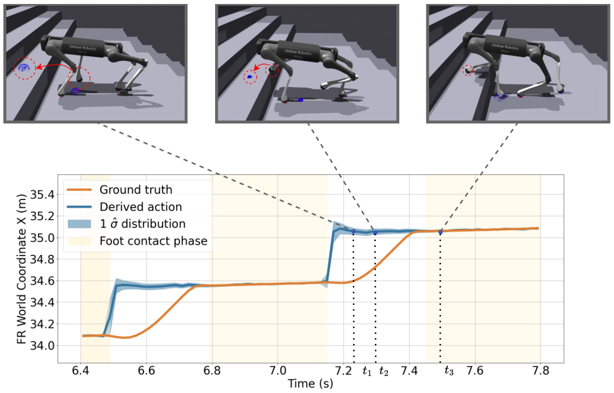

Fig. 6. Demonstration of the robot's derived action during the contact phase and the swing phase. The orange and blue curves, respectively, represent the x-coordinate of the derived action and the actual position of the front right foot. The blue area represents the 1-σ variance of the derived action. The three images from left to right at the top correspond to at time t_1, t_2, and t_3, indicating that the foot is just lifted, the foot is about to touch the ground, and the foot has already contacted the ground. The position and size of the blue sphere represent the mean and variance of the derived action.

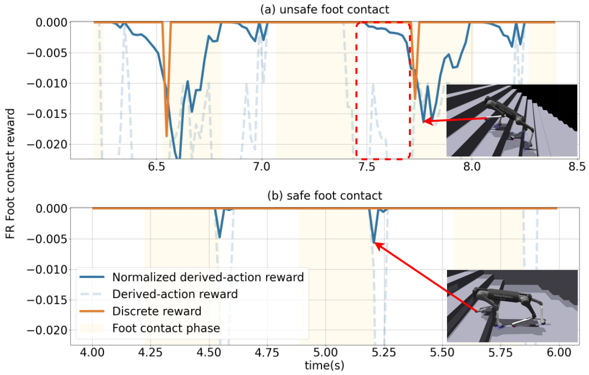

Fig. 7. These are the reward curves for derived-action and discrete action rewards during unsafe foot contact (a) and safe foot contact (b), respectively. The episodes of unsafe foot contact and safe foot contact are collected from two policies at the beginning and end of training, respectively. The foot contact reward is calculated by Equation 4. The blue curve shows the derived-action reward normalized using the variance decay coefficient, the transparent blue dashed line shows the original derived-action reward and the orange curve shows the discrete reward. The red dashed box demonstrates that derived action also generates incentives in the swing phase.

C. Reward of Derived Action

The derived-action reward accelerates the convergence of the training process and produces robust results by generating continuous rewards compared to the discrete reward function. At the beginning of training with DAO, as shown in Figure 7(a), the blue curve declines slowly at first, produces a larger peak at the moment of contact, and then rises to zero as the contact phase ends. As shown in Equation 4, the reward for the derived action at the start of the foot lift is maintained to a higher value after being reshaped by the large variance decay coefficient, even though the original reward for the derived action is relatively low. Unlike the discrete

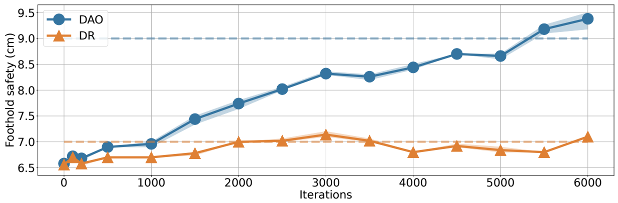

Fig. 8. Comparison of the convergence speed and effectiveness of DAO and DR in terms of foothold security from the 0th to the 6000th iteration. Vertical coordinates indicate the distance from the foothold to the edge.

foothold reward in orange, the derived-action reward has an effect during both the swing and contact phases. Figure 7(b) demonstrates that the derived-action reward remains high throughout the locomotion when the policy converges. It indicates that the policy automatically selects a safe foothold and plans a collision-free leg trajectory to reach the selected foothold.

D. Speed and Performance of Convergence

Continuous incentives of the derived-action reward function can significantly improve the efficiency of convergence. We select a checkpoint of Baseline and fine-tune it for 6000 iterations using the derived-action reward function and the discrete reward function, respectively. We take the policies with 0 to 6000 iterations and compare their average foothold safety and the number of foot collisions on continuous staircase terrain. For a more intuitive comparison, we define the average foothold safety as follows:

$$R_{safety} = \frac{1}{N}\sum_{i=0}^{N} \|p_i - p_{edge}\|, \tag{5}$$

$$R_c = \sum_{t=0}^{T} \sum_{i=1,2,3,4} \max(sgn(\|F^i_{xy,t}\| - 5|F^i_{z,t}|), 0), \tag{6}$$

where N represents the total times the four feet contact the terrain in an episode, p_i, p_{edge} denotes the actual foothold and the nearest step edge locations, respectively. R_{safety} is used to compute the average distance of footholds from the step edges during the motion process, which is more intuitive compared to Equation 4. Ideally, the robot will choose the midpoint of the edges of the two steps as the foothold for each locomotion procedure. R_c is used to evaluate the number of unintended foot collisions within an episode. When traversing on challenging terrain, such as stairs with narrow steps, robots may stumble, resulting in a dangerous body posture.

We construct a continuous staircase with a width of 0.35 m and a height of 0.15 m in IsaacGym, and select DR and DAO policies with iterations between 0 and 6000 to evaluate the average foothold safety. Each policy controls the robot to locomote on the terrain for 10 minutes with randomly sampling velocities between [-1.0, 1.0] m/s. We test each policy five times.

Figure 8 shows that as training progresses, DAO converges faster, indicating that the introduction of derived-action rewards leads to better training effects than the discrete reward.

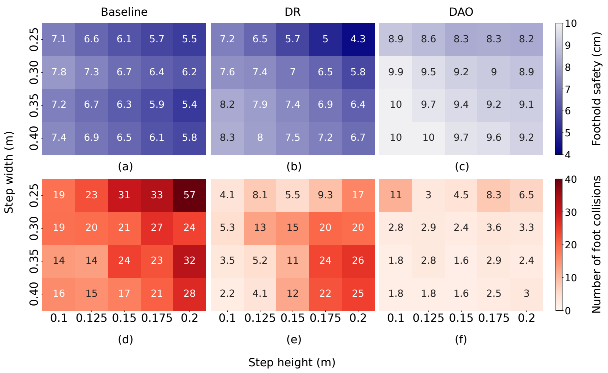

Fig. 9. Performance of Baseline (a, d), DR (b, e), and DAO (c, f) on staircases of different heights and widths. The horizontal coordinate denotes the height of the stair step, and the vertical coordinate denotes the width of the stair step. The blue heatmap is the average foothold safety calculated by Equation 5, and the red heatmap is the average number of foot collisions. Darker colors for all heat maps indicate poorer performance.

TABLE I

ROBUSTNESS TEST WITH SUCCESS AND COLLISION RATE.

	terrain	DR	DAO
Survival rate (%)	regular staircase	90	**100**
	floating staircase	80	**100**
	wide staircase	**100**	**100**
Collision rate (%)	regular staircase	5	**0.3125**
	floating staircase	2.5	**0.2083**
	wide staircase	0.2083	**0**

DAO can take a foothold safely up to 10 cm from the edge of the steps, while the DR can only improve slightly.

To quantify the final performance of the Baseline, DR, and DAO, we construct continuous staircases in IsaacGym with widths ranging from 0.25 m to 0.4 m and heights ranging from 0.1 m to 0.2 m. We evaluated each policy using the average foothold safety and the number of foot collisions.

As shown in Figure 9 (a-c), DAO has a higher average foothold safety on continuous steps. Even on the most challenging terrain with a width of 25 cm and a height of 20 cm, the DAO can traverse the terrain with an average foothold safety of 8.2 cm. In comparison, on DR and Baseline, these values are only 4.3 cm and 5.5 cm, respectively. On all staircase terrain, the DAO selects safe footholds and avoids stepping on edges as much as possible. At the same time, on a 40 cm wide stair, DAO does not blindly choose the safest footholds but considers the appropriate ones to achieve robust locomotion. In Figure 9 (d-f), DR and DAO have fewer collisions than Baseline. Throughout the episode, DAO can easily keep the number of collisions less than ten, while DR's foot collision times increase significantly as the step height increases. Baseline is highly prone to stumble on any terrain since it does not have a foothold safety reward. In summary, these results demonstrate the superior effectiveness of our proposed method.

E. Real World Deployment

We deploy our controller on an Aliengo made by Unitree Robotics and modify the sensor configuration. The head

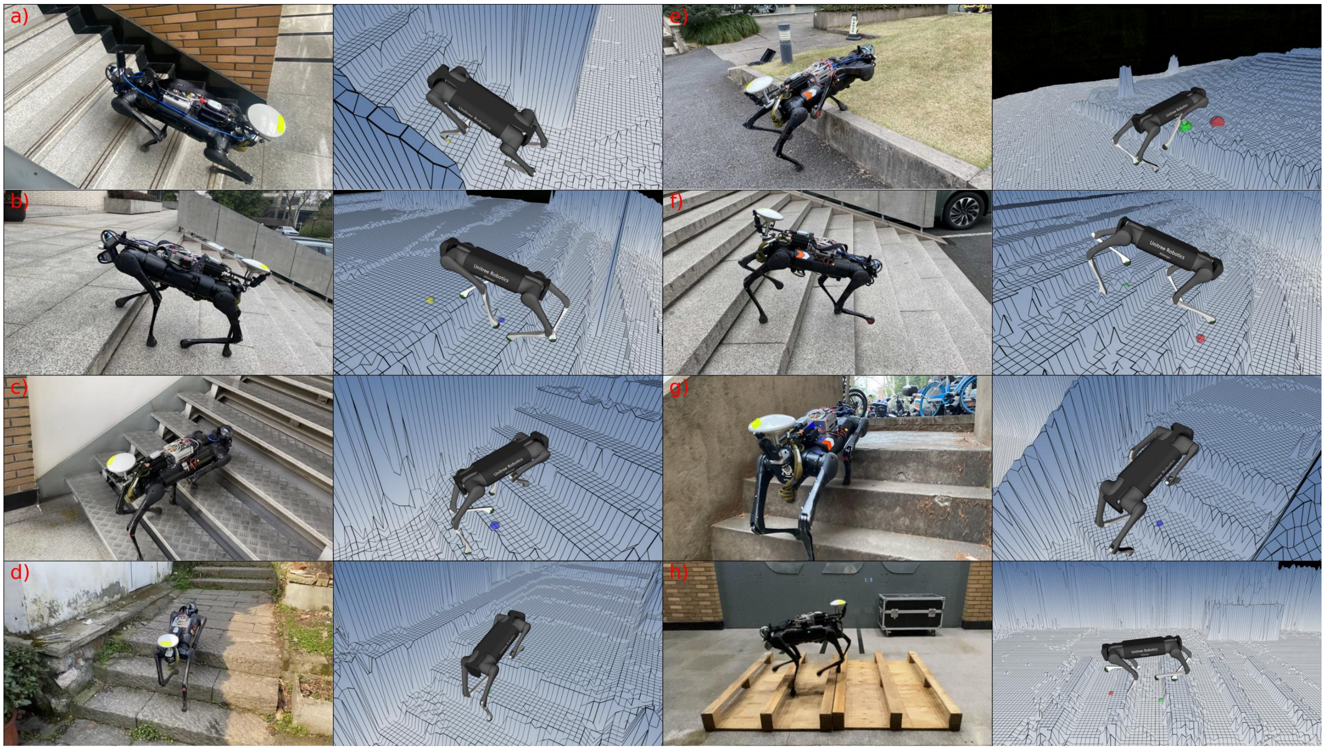

Fig. 10. Performance of the DAO on real-world environments and its corresponding elevation map. With our controller, the robot can traverse terrains like (a) regular indoor staircases with a step width of 25 cm and a height of 16 cm, (b, f) wide outdoor staircases with a step width of 35 cm and a height of 16 cm, (c) floating staircases with a step width of 25 cm and a height of 17 cm, (d) irregular steps in the wild, (e) single steps with a height of 25 cm, (g) high continuous stairs with a step width of 25 cm and height of 20 cm, and (h) wooden thresholds with a height of 10 cm. The red, yellow, blue, and green spheres represent the derived action of the FR, FL, RR, and RL foot, respectively.

of Aliengo is equipped with two LIVOX-MID360 LiDAR sensors for a 120-degree vertical field of view. We sampled the height of the terrain from a constructed elevation map as perceptual inputs to the controller. All terrain-sensing algorithms and control policy run on an onboard mini PC with an Intel 13700H processor and 16GB of RAM. The policy runs at 50 Hz, and we use a PD controller with proportional and derivative gains of $K_p = 40$ and $K_d = 1.5$, respectively, to track the desired joint positions at 1000 Hz.

We demonstrate the effectiveness of our model on regular staircases, floating staircases, wide staircases, high steps, and thresholds. As shown in Figure 10, scenes (a, b, f, g) are regular staircases with varying heights and widths, which shows that our policy can be widely applied to different staircases. Scene (c) is a special kind of floating staircase where the robot's foot can easily get stuck between two steps. Scene (d) is an outdoor hiking trail with masonry steps of varying widths and heights. Scene (e) is a high stepping stone to verify that our policy will choose appropriate footholds to prevent stumbling. Scene (h) is continuous wooden sills, where the robot is prone to stumble. In scenes (a, b, c), the robot is tested ten times by moving at a speed of 0.5 m/s, and then we recorded whether it successfully passed and the number of times the foot collided with the step wall. This challenging staircase verifies that our method ensures the safety of footholds and the stability of locomotion, as our policy avoids collisions with the vertical surface of the steps.

As shown in Table I, our model is superior in terms of survival rate and collision rate compared to DR. The robot is 100% guaranteed to traverse the terrain when running at a speed of 0.5 m/s. The foot collision probability also stays within 0.5% on each terrain, allowing almost collision-free passage on each terrain. This is particularly important for robots in application environments similar to scene (c), such as in industrial patrol, to improve the safety of robots when working on continuous and complex staircases.

IV. CONCLUSIONS

In this work, we introduce the derived-action optimization framework, which enables quadruped robots to utilize a policy to output a high-level derived action after developing natural locomotion capabilities. This derived action provides a more intuitive representation of the policy's locomotion ability, which can be targeted to accelerate the training convergence speed and convergence effect. The derived-action framework can introduce more substantial reward incentives than normal reinforcement learning frameworks. We demonstrate its effectiveness on Unitree's Aliengo robot. The limitations of the derived-action framework are how to select appropriate derived action and how to design the derived-action reward function to speed up the convergence speed and convergence effect of the model. In future work, we will implement it in more challenging tasks, such as jumping to high platforms, by combining robot state and action prediction.

References

[1] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, p. 10822–10825, Jan 2008.

[2] B. Katz, J. D. Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019.

[3] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, Jan 2019.

[4] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, Oct 2020.

[5] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[6] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim, "Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020.

[7] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive locomotion in rough terrain – online foothold optimization," *IEEE Robotics and Automation Letters*, p. 5370–5376, Oct 2020.

[8] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model predictive control," Aug 2022.

[9] O. Villarreal, V. Barasuol, P. Wensing, D. Caldwell, and C. Semini, "Mpc-based controller with terrain insight for dynamic legged locomotion," *arXiv: Robotics,arXiv: Robotics*, Sep 2019.

[10] K. Wang, T. Chen, J. Bi, Y. Li, and X. Rong, "Vision-based terrain perception of quadruped robots in complex environments," in *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2021.

[11] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on Robot Learning*, pp. 403–415, PMLR, 2023.

[12] S. Kareer, N. Yokoyama, D. Batra, S. Ha, and J. Truong, "Vinl: Visual navigation and locomotion over obstacles," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2018–2024, IEEE, 2023.

[13] R. Yang, M. Zhang, N. Hansen, H. Xu, and X. Wang, "Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers," *arXiv preprint arXiv:2107.03996*, 2021.

[14] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," *arXiv: Robotics,arXiv: Robotics*, Sep 2021.

[15] P. Brakel, S. Bohez, L. Hasenclever, N. Heess, and K. Bousmalis, "Learning coordinated terrain-adaptive locomotion by imitating a centroidal dynamics planner,"

[16] M. Bogdanovic, M. Khadiv, and L. Righetti, "Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization,"

[17] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "Dtc: Deep tracking control – a unifying approach to model-based planning and reinforcement-learning for versatile and robust locomotion," Sep 2023.

[18] H. Shi, Q. Zhu, L. Han, W. Chi, T. Li, and M.-H. Meng, "Terrain-aware quadrupedal locomotion via reinforcement learning," Oct 2023.

[19] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control.," *IEEE Transactions on Robotics*, p. 2908–2927, Oct 2022.

[20] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision," *Neural Information Processing Systems,Neural Information Processing Systems*, Dec 2017.

[21] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3019–3026, 2018.

[22] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "Robot-centric elevation mapping with uncertainty estimates," in *International Conference on Climbing and Walking Robots (CLAWAR)*, 2014.

[23] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu based physics simulation for robot learning," *Neural Information Processing Systems,Neural Information Processing Systems*, Aug 2021.

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