Investigating The Influence Of Curriculum Learning On Quadrupedal Parkour

Leon Magnus, Oleg Arenz, Nico Bohlinger

Abstract— **Recent advances in massively parallelized simulation have driven rapid progress in quadrupedal locomotion, enabling human-like parkour scenarios that push both hardware and computational limits. Solving such complex tasks benefits from curriculum learning, most often applied to terrain generation so that an agent progresses from simple locomotion to more challenging skills such as box climbing, jumping, and obstacle avoidance. However, most existing approaches keep the ranges for domain randomization and the magnitude of reward penalties fixed throughout training which can lead to suboptimal learned behaviors and poor sample efficiency. In this work, we extend performance-based curriculum learning beyond terrain generation to also scale domain randomization parameters and reward penalty terms. We evaluate four training configurations on a challenging simulated box climbing task and demonstrate real world deployment. Our results show that curriculum-based scaling of both domain randomization and reward penalties strongly increases sample efficiency and improves the final reached curriculum level by 31%.**

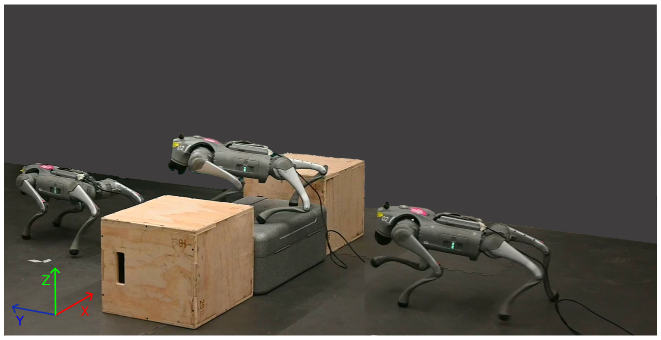

Fig. 1: Lab box-climbing setup. The agent must climb over the gray box, which is 27.5 cm high. Wooden boxes are used to stabilize the setup and are not observed by the agent.

I. Introduction

Quadrupedal robots have achieved remarkable progress in recent years, demonstrating agile locomotion across a wide range of challenging terrains. Their applications extend from hiking in mountainous environments [12] to navigating parkour-like scenarios [4], [6], [7], [21], where the ability to traverse complex settings such as stairs or obstacles is essential. Among the most demanding skills in this context is box climbing, which involves stepping or jumping onto elevated surfaces of varying heights. Mastering this ability requires precise coordination, robust perception, and adaptability to changes in both the environment and the commanded motion.

Many existing approaches employ curriculum learning for terrain generation, enabling an agent to progress from simple locomotion tasks to increasingly complex challenges such as box climbing. However, domain randomization, which is used to enable robust zero-shot transfer, and reward penalties, which discourage undesired behaviors, are typically kept constant throughout the training process. While some studies [10] have explored curriculum-based scaling of domain randomization ranges and reward penalties, to the best of our knowledge these methods have been applied only to basic locomotion tasks.

In this work, we extend the use of curriculum learning to also scale domain randomization parameters and reward penalty terms over the course of training. Our experiments show that this approach improves performance and increases sample efficiency for the challenging task of box climbing.

II. Related Work

In recent years, quadrupedal locomotion has made significant progress by leveraging Deep Reinforcement Learning (DRL) methods to solve complex locomotion tasks. Heavily parallelized simulation frameworks such as Mujoco MJX [1] and Isaac Sim [15] have made data collection significantly faster, enabling the learning of locomotion policies within minutes [18]. These methods have allowed agents to acquire robust and agile policies in highly complex environments [12], [18]. Most approaches employ Proximal Policy Optimization (PPO) [19]. Typically, an RL policy is trained entirely in simulation and then deployed directly to the real robot. This process is referred to as zero-shot transfer, as the agent does not observe any real-world data during training. To enable successful transfer, simulations are heavily randomized—a technique known as Domain Randomization (DR) [20], in which physical parameters, model properties, and perturbations are varied. As zero-shot transfer of model-free RL policies has shown promising results, recent works have introduced even more complex locomotion environments, requiring agents to master parkour-like skills such as jumping over gaps, climbing boxes, or avoiding obstacles [4], [6], [7], [21]. These challenging scenarios are referred to as Quadrupedal Parkour tasks. Such works have demonstrated impressive capabilities, with agents successfully climbing boxes significantly taller than their own body height. To tackle such complex tasks, agents often rely on structured guidance during training, especially in early stages when rewards are sparse and exploration may be insufficient before failure states occur. One effective strategy to provide such guidance in RL is called Curriculum Reinforcement Learning (CRL), where the agent first learns to solve simpler tasks and then leverages the acquired knowledge to tackle progressively harder ones. According to [14], a curriculum in reinforcement learning can be represented as a directed acyclic graph that specifies the order in which task samples or transitions should be presented to the agent during training. Curricula can be constructed either offline or online. In an offline setting, the graph is predefined before training based on task properties. In an online setting, edges are added dynamically according to the agent's learning progress and performance. [9] interpret curricula as interpolations between task distributions and propose updating the current distribution via a constrained optimal transport formulation, using Wasserstein distances to adapt tasks according to agent performance. An early example of incorporating online curriculum learning to scale DR for improved zero-shot transfer is the Rubik's Cube solving work by OpenAI [16]. In this work, the authors introduce Automatic Domain Randomization (ADR), where both visual and physical randomization parameters are automatically adjusted based on the agent's performance. Training begins with an easier distribution of environment parameters and, as the agent's proficiency improves, the randomization ranges are progressively expanded to include more challenging variations. This performance-driven progression effectively implements a curriculum, enabling the policy to adapt gradually to increasingly complex conditions while maintaining robustness for sim-to-real transfer.

Motivated by the impressive results of such online performance-driven curricula, researchers have applied similar methods in quadrupedal locomotion. [11] employ an adaptive velocity-command curriculum that starts with low-speed, low-turn-rate motions and progressively expands toward high-speed, agile behaviors, while simultaneously applying DR over physical parameters such as mass, friction, restitution, and actuator strength. The policy first masters feasible, easier motions before facing the full range of dynamics. [13] focus on learning the DR distribution itself, steering it toward parameter regions where the current policy achieves high reward, while retaining enough diversity to preserve robustness. [10] extend the curriculum concept beyond task commands and physical parameters. Their setup jointly scales command ranges, scope of DR parameters, terrain generation, and the magnitude of reward penalties for inefficient, unstable, or unsafe motion. In this way, the curriculum progresses along multiple axes, so that as policy competence increases, the agent is exposed to harder tasks, more diverse dynamics, and stricter performance requirements. However curriculum updates are performed using separate evaluation runs each after 100,000 training steps.

In other recent quadrupedal locomotion works, curricula have been used for terrain generation [4], [18], [21]. These methods employ a game-inspired curriculum in which the agent's current performance determines whether the difficulty is increased or decreased. This approach is particularly well-suited for heavily parallelized simulations, as the curriculum can be adjusted on the fly without requiring separate evaluation runs. In our work, we extend this online, game-inspired curriculum beyond terrain generation to also encompass reward penalties and DR ranges, following the approach of [10], but applying it in an online RL setting with PPO [19], specifically targeting quadrupedal box-climbing. Furthermore, we investigate the influence of each curriculum-scaled component and analyze their potential benefits. In the following section we describe our approach.

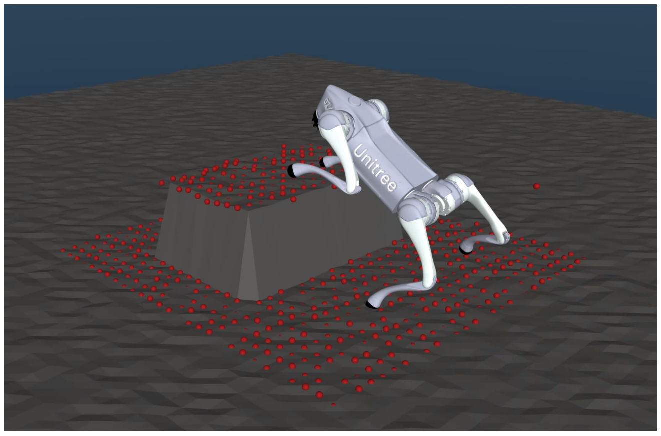

Fig. 2: Visualization of the used height map as exteroceptive information. The policy receives the z-values of each grid point. The initial position of each grid point is the same. We apply some noise on xyz-position of each grid point

III. Methodology

We want to enable a quadrupedal robot to perform the skill of *box climbing*—overcoming a box that is at least the robot's standing height or higher. To achieve such a skill, we train a policy that maps proprioceptive and exteroceptive observations directly to target joint positions. The section first outlines the problem formulation following the agents' architecture. Afterwards, we discuss observations, reward function and the overall learning setup. Finally we introduce our game-inspired performance-based curriculum.

A. Problem Formulation

We model our problem as a Partially Observable Markov Decision Process (**POMDP**), in which the objective is to determine actions that maximize the expected cumulative reward. Formally, a POMDP is defined as

$$\mathscr{P} = (\mathbf{S}, \mathbf{A}, \mathbf{O}, \mathbf{P}(s' \mid s, a), \mathbf{O}(o \mid s', a), \mathbf{R}(s, a)),$$

where $\mathbf{S}$ denotes the set of states, $\mathbf{A}$ the set of actions, and $\mathbf{O}$ the set of possible observations. The function $\mathbf{P}(s' \mid s, a)$ represents the state transition probabilities, while $\mathbf{O}(o \mid s', a)$ specifies the probability of observing o given that the system transitions to state s' after taking action a. Finally, $\mathbf{R}(s, a)$ defines the reward function associated with executing action a in state s.

In a POMDP the agent does not have direct access to the state in which it is rather it is provided by observation of this state. Our goal is to learn a policy $\pi_\theta(a \mid o)$ that maps observations of the current state directly to actions. In our case an action consists of 12 joint position deviations from a given nominal joint position. One target joint position deviation for each joint of the GO2 quadruped. We transform the

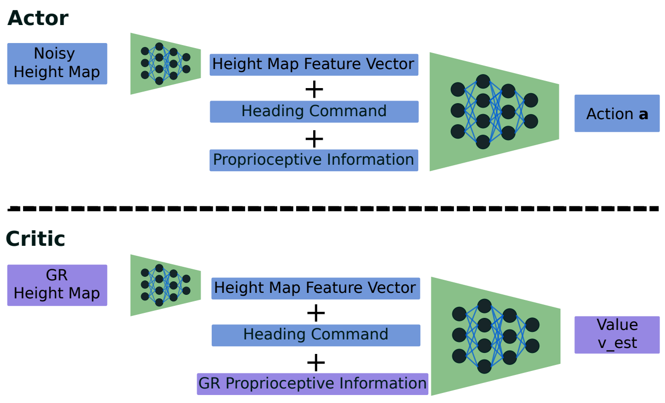
Actor
Noisy Height Map
Height Map Feature Vector
+
Heading Command
+
Proprioceptive Information
Action **a**
Critic
GR Height Map
Height Map Feature Vector
+
Heading Command
+
GR Proprioceptive Information
Value v_est

Fig. 3: PPO Architecture. Both the actor and critic share a similar network architecture. First, an MLP encoder processes the height map and encodes it into a feature vector. This feature vector is concatenated with proprioceptive information and the heading command. The concatenated input is subsequently passed through a second MLP, which maps it to either actions (for the actor) or estimated values (for the critic). Purple highlights differences in input and output between the actor and critic. Importantly, the actor and critic each have their own set of trainable MLP parameters. The abbreviation GR stands for Ground-truth.

policy output into torque commands using a PD-Controller.

$$\tau = [p \cdot ((\mathbf{q}_{\text{nom}} + a \cdot s) - \mathbf{q}) - d\dot{\mathbf{q}}] \tag{1}$$

Equation 1 shows the transformation from action output a to torque commands τ which are sent to the robot. $\mathbf{q}_{\text{nom}}$ is a given nominal joint position, s is a scaling factor with which we scale the action having a value of $s = 0.3$. The constant scaling mainly impacts early training, where it promotes exploration by tempering random high outputs that would otherwise trigger failure states. $\mathbf{q}$ are the current joint positions and $\dot{\mathbf{q}}$ are the current joint velocities. p and d are the proportional and derivative gains with the values $p = 20.0$ and $d = 0.5$,

B. Architecture

We leverage a model-free asymmetric actor-critic Reinforcement Learning setup similar to [17] in which the critic has access to ground-truth information which is only available in simulation. Specifically we use an own implementation of the PPO algorithm which is inspired from the Brax [5] PPO implementation. We leverage ADAM [8] as our optimizer. All hyperparameters used in our learning setup can be found in Table II. Figure 3 illustrates the architecture used in our PPO setting. The actor and critic each consist of two main components. First, a Multilayer Perceptron MLP encoder processes a given height map of the environment to a feature vector. This feature vector is concatenated with command inputs and proprioceptive information. Afterwards the concatinated input vector is passed through a second MLP, which maps the input to either actions or the estimated value v_{est}. While the overall structure is similar for both actor and critic, the critic has its own set of trainable parameters. Additionally, the critic benefits from access to additional and ground-truth proprioceptive information aswell as groundtruth exteroceptive information available in simulation. Table I provides a detailed visualization of each layer in both architectures, highlighting the differences.

C. Observations

In this section we describe the observations the agent has access to in the provided POMDP.

1) Exteroceptive Observation: To enhance robustness against real-world inaccuracies such as drift or sensor noise, we augment the observations with three distinct types of perturbations. The first is uniform XY-plane offset noise, which models scenarios in which the height map is slightly shifted due to drift or temporal delays. The second is Gaussian Z-height noise, which accounts for inaccuracies in the measured box heights. The third is uniform outlier noise, which improves the policy's resilience to spurious measurements. The specific hyperparameters for each noise type are listed in Table IV. The agent receives as input height values of each grid point, resulting in a height map vector of size $28 \times 28 = 784$.
In the laboratory experiments on the real robot, we leverage OptiTrack [2] to generate a 2D x–y grid of height values analogous to the simulation setup. The positions of both the robot and the box are obtained via OptiTrack which we use to reconstruct the local height map centered at the robots LIDAR position. A detailed description of the experimental setup is provided in Section IV-B.

2) Command: In recent quadruped locomotion approaches, the agent is typically provided with a randomly sampled local linear target velocity in x- and y-direction and a yaw angular target velocity [18] [12]. Such a command formulation is easy to interpret and allows the agent to be controlled via joystick. However, for box climbing and, more generally, robot parkour, following a local target velocity can be challenging. For instance, the agent may exploit a collision with the box as a means of reorienting toward an alternative global heading, thereby learning to circumvent the obstacle rather than attempting to climb it. Furthermore the agent should have some freedom in selecting a current target velocity. To address these limitations, recent work has shifted to a different way of commanding the robot. In [4], the authors leverage global waypoints that the robot is required to reach. Using the global positions of both the robot and the waypoint, a target heading h_t can be computed to continuously command the robot towards the goal waypoint. In addition to the target heading, a desired velocity is specified which acts as a lower bound.

$$\mathbf{C} = [\mathbf{h}_t, v_t]$$

$$\mathbf{h}_t = \frac{\mathbf{p}_g - \mathbf{p}_r}{\|\mathbf{p}_g - \mathbf{p}_r\|} \tag{2}$$

$$v_t \in \{0.4 \text{ m/s} - 1.0 \text{ m/s}\}$$

We adopt the commanding structure from [4] and define 3 waypoints in the the environment which the robot should reach. Hereby the target heading and desired velocity are concatenated leading to command $\mathbf{C}$. Equation 2 visualizes the computation of $\mathbf{C}$. $\mathbf{h}_t$ represents the target heading and v_t is the desired velocity.. The current waypoint the agent should reach is stated with $\mathbf{p}_g$. The current position of the robot is stated with $\mathbf{p}_r$.

3) Proprioceptive Observation: To successfully learn a locomotion policy, an agent requires an accurate estimate of its current state. For this purpose, the actor receives proprioceptive inputs comprising current joint positions and velocities, last action applied, trunk angular velocity, and projected gravity vector. These signals form a 42-dimensional proprioceptive feature vector. To account for sensor inaccuracies present on real hardware, noise is applied to each component of this vector, with specific types and magnitudes detailed in Table IV. In contrast, the critic has access to additional proprioceptive information, namely trunk linear velocity and trunk height above ground. When concatenated with the actor's proprioceptive vector, these features yield a 53-dimensional critic feature vector. No noise is applied to critic inputs.

D. Reward Function

$$r_{\text{tracking}} = \min\left(\langle \mathbf{v}, \mathbf{h}_t \rangle, v_t\right) \tag{3}$$

We leverage the same base reward function term as presented in [4]. Equation 3 defines the base tracking reward. The target heading $\mathbf{h}_t$ is computed as described in Equation 2. The dot product is calculated between the current velocity vector $\mathbf{v}$ and the target heading $\mathbf{h}_t$, yielding a value that is then compared to the current target velocity v_t, with the minimum of the two returned. Such a reward formulation enables the agent to exceed the target velocity when advantageous, such as when approaching obstacles that require higher speeds. In addition to the base reward, several penalty terms are employed to encourage a smoother gait. The complete set of penalty terms and their corresponding coefficients are detailed in Table III.

E. Domain Randomization

To enable zero-shot transfer from simulation to the real world, we employ Domain Randomization across five categories: action delay, control randomization, model randomization, perturbation randomization, and initial robot position randomization. Action delay is introduced by probabilistically postponing the application of actions in simulation, improving robustness to real-world latency. Control randomization varies the PD controller parameters in Equation 1, including p- and d-gains, joint offsets, and motor strengths. Model randomization alters both physical properties and robot parameters to increase resilience to modeling errors. Perturbation randomization applies random linear velocities to the trunk, simulating unmodeled external forces. Initial position randomization varies the robot's starting configuration at the beginning of each episode, promoting adaptability to diverse initial states. All parameter ranges are listed in Table V, and the DR setup follows the approach in [3].

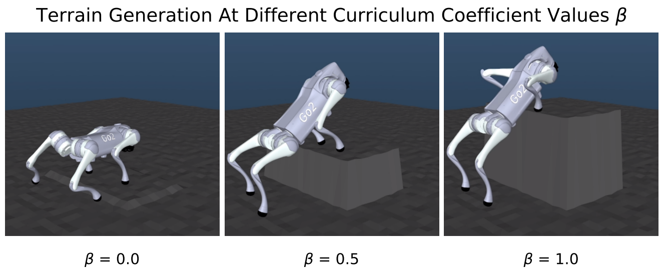
Terrain Generation At Different Curriculum Coefficient Values β
β = 0.0
β = 0.5
β = 1.0

Fig. 4: Training environment at different curriculum coefficients $\beta \in \{0.0, 0.5, 1.0\}$. The box height is increased with increasing curriculum coefficient. The different curriculum coefficient values correspond to box heights of $\{0.02m, 0.23m, 0.45m\}$.

F. Training Environment

We use MuJoCo MJX [1] as the simulator due to its strong parallelization capabilities and its ability to modify the underlying height field during simulation, which offers a key advantage over alternatives such as Isaac Sim [15]. This functionality enables fine control over terrain difficulty, allowing the agent to progress from simple to complex scenarios. To train box-climbing skills, we initialize a six-meter by six-meter height field containing a single box with an initial height of $0.02m$. Training uses two box orientations. In the width-oriented case, the longer side is perpendicular to the walking direction and acts as a hurdle. In the length-oriented case, the longer side is parallel to the walking direction and requires the robot to balance on top without falling. The maximum box height reaches $0.45m$ and increases over the course of training, as shown in Figure 4. To maintain proficiency on lower boxes, two environments are used. One contains the current maximum height, while the other has a height uniformly sampled between $0.02m$ and the current maximum. The maximum-height environment is selected with a probability of 70%, and the random-height environment with 30%. A new environment is sampled after each episode.

G. Game-Inspired Curriculum

We employ a game-inspired, performance-based curriculum to guide the policy from initially simple environments toward progressively more challenging scenarios over the course of training. The curriculum is controlled by a single coefficient $\beta \in [0,1]$, which simultaneously scales DR, reward penalties, and terrain difficulty. For DR, β determines the active sampling range for each randomized parameter. At $\beta = 0.0$, the range is minimal and centered around the nominal (midpoint) value of the parameter. As β increases, the range expands linearly toward its predefined maximum bounds, reaching the full DR range at $\beta = 1.0$. This ensures that the agent first learns under stable, near-nominal conditions before being exposed to the full variability of the environment. Reward penalties are scaled directly by β, such that penalties are zero at the start of training and gradually

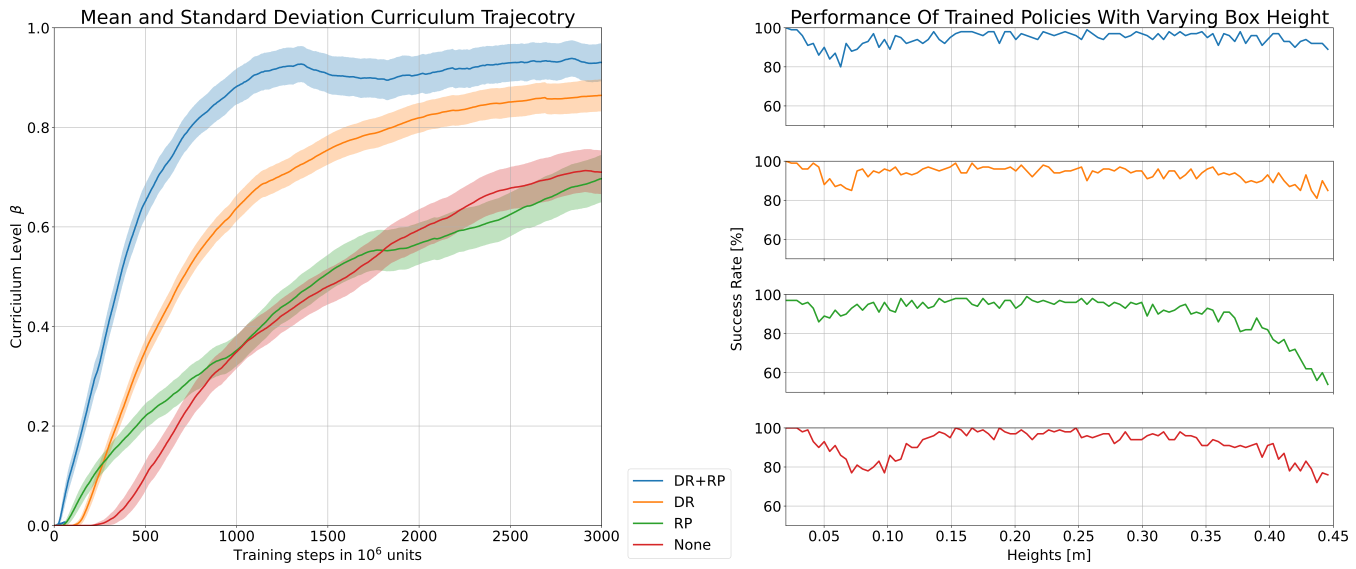

Fig. 5: Policy performance across four training configurations. **DR** denotes that Domain Randomization parameters were scaled using the curriculum coefficient β, while **RP** denotes that reward penalties were scaled using β. **None** refers to the baseline configuration, in which neither Domain Randomization nor reward penalties were scaled by the curriculum. The left plot shows the curriculum level trajectory during training for all four policies. The right plot presents the performance of the final trained policies in simulation for varying box heights. All policies were trained for a total of 3 billion steps.

increase in magnitude as the curriculum advances. This progression encourages the agent to first acquire core task skills before being pushed to optimize for efficiency, stability, or safety under stricter constraints. In terrain generation, β controls the maximum box height. Specifically, the maximum box height increases linearly from 0.02m at $\beta = 0.0$ to 0.45m at $\beta = 1.0$, allowing the agent to adapt to increasingly demanding climbing challenges as training progresses. All environments are initialized with $\beta = 0.0$. After termination of an episode in an environment its curriculum level is updated based on the agent's performance. β is increased if Equation 4 evaluates to true, and decreased if Equation 5 evaluates to true.

$$I: \big(w = 3 \ \vee \ (d > 0.9D \ \wedge \ \neg t)\big) \tag{4}$$

$$S: \big(w < 3 \ \wedge \ d < 0.9D\big) \tag{5}$$

Here, w denotes the number of waypoints reached, d the distance traveled in the terminated episode, and t indicates whether the episode was truncated. D is the total distance between the start and goal while following the provided waypoints. Curriculum updates are only applied if the target velocity in the terminated episode v_t was greater than zero and the agent was placed in an environment with the current maximum box height. The update step size is fixed at $\Delta\beta = \frac{1}{200}$. We hypothesize that jointly scaling DR, reward penalties, and terrain complexity via this curriculum improves sample efficiency and enables the agent to solve more complex environments.We evaluate the contribution of each scaling component in experiments that are described in the following section.

IV. Experiments

We evaluate our approach in simulation, examining whether scaling reward penalties and domain randomization improve the learning process. Next we asses the learned policy in real-world experiments to show zero-shot transfer capabilities. The experimental setups are described in the following sections.

A. *Simulation*

To evaluate the influence of each curriculum scaling component, we train multiple policies in which the elements scaled by the curriculum are systematically varied. Since scaling terrain complexity via a curriculum is already common practice and has been shown to significantly improve learning of complex behaviors, our investigation focuses on scaling Domain Randomization DR and reward penalties RP. We therefore consider four training configurations. In the first, neither DR nor RP are scaled by the curriculum. In the second, only DR parameters are scaled, while RP remains fixed. In the third, only RP is scaled, with DR kept constant. Finally, in the fourth configuration, both DR and RP are scaled simultaneously according to our proposed method. For each configuration, we compare the resulting curriculum trajectories during training and evaluate success rates across varying box heights. A trial is considered *successful* if the agent reaches all three waypoints within 14 s without falling or colliding with the box. For each box height we repeat the experiments 100 times.

B. *Real World*

We replicate the box-climbing task in a controlled laboratory environment. Figure 1 illustrates the setup, in which the primary objective is for the robot to climb over a gray

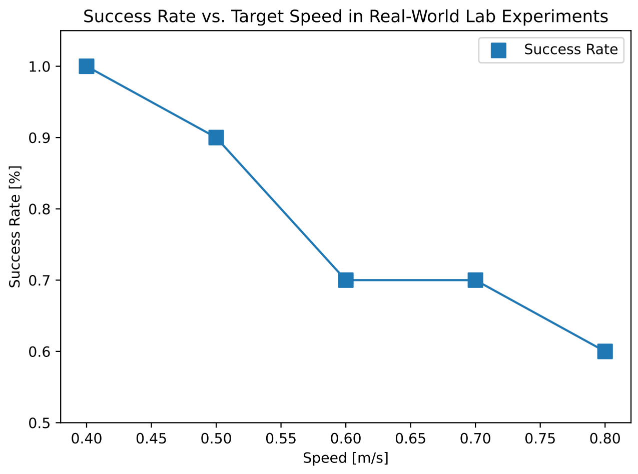

Fig. 6: Success Rate of the learned policy in the lab setting given different target velocities. For each target velocity 10 Runs were executed.

box with a height of 27.5 cm. Wooden boxes, positioned to the left and right of the gray box which are not visible for the agent, serve as stabilizers to prevent the box from shifting upon impact. The robot starts approximately 2 m in front of the box, with the setup aligned along the global y-axis. The robot positions shown in Figure 1 correspond to the waypoints provided as input for completing the climb. In each trial, the agent is tasked with reaching the same three waypoints. To assess the real-world robustness of our approach, we measure the success rate, defined as the percentage of trials in which the robot reaches the final waypoint after climbing the box without falling or losing stability. We evaluate target velocities of $[0.4, 0.5, 0.6, 0.7, 0.8]\,\mathrm{m/s}$, repeating each experiment ten times.

V. Results

In this section, we present the experimental results. We begin with evaluating the influence of the our proposed performance-based curricullum scaling of DR and PR. Afterwards we esimtate the real-world transfer of our method.

A. Simulation

Figure 5 illustrates the performance differences between four policies trained with distinct configurations, as described in Section IV-A. Comparing the mean curriculum trajectories during training, we observe distinct behaviors across the four policies. Scaling both Domain Randomization and reward penalties DR+RP yields the highest curriculum level, with a final mean coefficient of $\beta_{DR+RP} = 0.93$. Scaling only DR results in a final mean coefficient of $\beta_{DR} = 0.86$. The RP and None configurations achieve similar final coefficients of $\beta_{RP} = 0.69$ and $\beta_{None} = 0.71$, respectively. We observe that at the start of training the RP policy exhibits a faster initial increase in curriculum level compared to both DR and None.

From these results, we draw three main conclusions. First, scaling reward penalty terms does not have a direct impact on the final curriculum level, but it facilitates exploration in the early stages of training, as the agent can explore more freely without being heavily penalized. Second, scaling DR with a curriculum has the largest effect on the final maximum curriculum level. This is evident from the comparison between Policy None and Policy DR, where Policy DR outperforms Policy None by 0.15, corresponding to an improvement of 30 curriculum levels. Finally, combining scaled DR and scaled RP yields the highest mean curriculum level at the end of training. Moreover, the curriculum level of the DR+RP policy increases most rapidly, converging after approximately 1.35 billion training steps. In contrast, the DR policy never reaches the level achieved by DR+RP and, more importantly, appears not to have fully converged even after 3 billion steps. This finding further supports the conclusion that scaling RP is particularly beneficial for early exploration, thereby improving the overall sample efficiency of the approach.

Second, we evaluate the performance of the trained policies for varying box heights in simulation. The results are presented on the right side of Figure 5, where the success rate over 100 trials is plotted for each box height. Across the full range of heights Policy DR+RP and Policy DR achieve the highest performance. In particular, near the maximum box height of 0.45m, both policies significantly outperform Policy None and Policy RP. A noticeable drop in performance for RP and None occurs around 0.35m, which is expected given that the mean curriculum coefficients of Policy RP and Policy None are approximately equal ($\beta_{RP} \approx \beta_{None}$), corresponding to an average maximum box height of 0.32m. Interestingly, all policies exhibit a performance drop for small boxes with heights between 0.05m and 0.075m. We hypothesize that such heights require a behavior that lies between standard walking and box climbing. As training progresses and the curriculum advances, these intermediate scenarios are encountered less frequently, potentially leading to performance degradation. Further analysis is needed to confirm this hypothesis.

From these evaluations, we conclude that jointly scaling domain randomization and reward penalties with terrain complexity produces better-performing policies for box climbing. Our results further suggest that scaling domain randomization enables the agent to reach higher curriculum levels, while scaling reward penalties is particularly beneficial for early exploration, as it reduces the penalty for initially sub-optimal behaviors. Our approach DR+RP yields substantial gains in sample efficiency, enabling faster and more effective learning. Moreover, it increases the final curriculum level reached by approximately 44 levels compared to the None baseline, corresponding to a relative improvement of 31%.

B. Lab Experiments

To show that our approach is capable of zero-shot transfer we evaluate the agent in the box climbing setup shown in Figure 1. Hereby we test the policy for multiple target speeds. The results are shown in Figure 6. We observe that the learned policy successfully transfers to the real world. At 0.4 m/s, the success rate reaches 100%, but performance declines as the target velocity increases. Compared to the simulation results, this drop in success rate is significantly more pronounced. As higher target velocities make the task more complex, the robot must rely more heavily on accurate perception of the real-world environment. Analysis of the failed runs shows that the robot often hits the front edge of the box with a foot, either pushing the box or getting the foot stuck, preventing further movement. This behavior suggests that inaccuracies or delays in the OptiTrack-based height map estimation may contribute to these failures, as even small perception errors can lead to mistimed foot placement at higher speeds. However a more detailed analysis of the failed states should be made in subsequent work to check this hypothesis and potentially increase the success rates.

VI. Conclusion & Outlook

In this work we showed that extending a performance-based curriculum with domain randomization and scaled reward penalties improves both performance and sample efficiency in quadrupedal parkour, with a particular focus on box climbing. Domain randomization enables faster progression to higher curriculum levels, while scaled penalties promote early exploration by reducing the cost of suboptimal actions. Across evaluations, our method outperforms approaches that omit or partially apply these components, achieving steeper curriculum advancement and greater sample efficiency. Moreover, our approach transfers to the real world in a zero-shot manner. Furthermore there are several promising directions for extending this work. In our experiments, we used 200 curriculum levels linearly spaced between zero and one. Building on this, two potential avenues for further investigation emerge. First, the effect of varying the number of curriculum levels could be examined. Second, it is interesting to explore whether a nonlinear spacing of curriculum levels influences performance.

References

[1] MuJoCo XLA (MJX) - MuJoCo Documentation — mujoco.readthedocs.io. https://mujoco.readthedocs.io/en/stable/mjx.html. [Accessed 04-08-2025].

[2] OptiTrack - Motion Capture Systems — optitrack.com. https://www.optitrack.com/. [Accessed 07-08-2025].

[3] Nico Bohlinger, Grzegorz Czechmanowski, Maciej Krupka, Piotr Kicki, Krzysztof Walas, Jan Peters, and Davide Tateo. One policy to run them all: an end-to-end learning approach to multi-embodiment locomotion, 2025.

[4] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450, 2024.

[5] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021.

[6] David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots, 2023.

[7] Hyeongjun Kim, Hyunsik Oh, Jeongsoo Park, Yunho Kim, Donghoon Youm, Moonkyu Jung, Minho Lee, and Jemin Hwangbo. High-speed control and navigation for quadrupedal robots on complex and discrete terrain. *Science Robotics*, 10(102):eads6192, 2025.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[9] Pascal Klink, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. On the benefit of optimal transport for curriculum reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(11):7191–7204, November 2024.

[10] Sicen Li, Yiming Pang, Panju Bai, Zhaojin Liu, Jiawei Li, Shihao Hu, Liquan Wang, and Gang Wang. Learning agility and adaptive legged locomotion via curricular hindsight reinforcement learning, 2023.

[11] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning, 2022.

[12] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62), January 2022.

[13] Melissa Mozifian, Juan Camilo Gamboa Higuera, David Meger, and Gregory Dudek. Learning domain randomization distributions for training robust locomotion policies, 2019.

[14] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey, 2020.

[15] NVIDIA. Isaac Sim.

[16] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand, 2019.

[17] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

[18] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. *CoRR*, abs/2109.11978, 2021.

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[20] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 23–30. IEEE Press, 2017.

[21] Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher Atkeson, Soeren Schwertfeger, Chelsea Finn, and Hang Zhao. Robot parkour learning, 2023.

APPENDIX

Actor Architecture		
Layer	Type	In/Out Dim
Height Encoder	Dense(256) + ELU	$H = 784 \rightarrow 256$
Height Encoder	Dense(128) + ELU	$256 \rightarrow 128$
Height Encoder	Dense(150)	$128 \rightarrow 150$
Concat	-	$A + E = 45 + 150 = 195$
Dense 1	Dense(512) + LayerNorm + ELU	$195 \rightarrow 512$
Dense 2	Dense(256) + ELU	$512 \rightarrow 256$
Dense 3	Dense(128) + ELU	$256 \rightarrow 128$
Output	Dense(24)	$128 \rightarrow 24$
Critic Architecture		
Layer	Type	In/Out Dim
Height Encoder	Dense(256) + ELU	$H = 784 \rightarrow 256$
Height Encoder	Dense(128) + ELU	$256 \rightarrow 128$
Height Encoder	Dense(150)	$128 \rightarrow 150$
Concat	-	$V + E = 53 + 150 = 203$
Dense 1	Dense(512) + ELU	$203 \rightarrow 512$
Dense 2	Dense(256) + ELU	$512 \rightarrow 256$
Dense 3	Dense(128) + ELU	$256 \rightarrow 128$
Output	Dense(1)	$128 \rightarrow 1$

TABLE I: Actor and Critic architectures, each with their own trainable Height Encoder. H: height sample size, E: height encoder size, A: actor obs size, V: critic obs size, O: actor output size.

Hyperparamter	**Value**
Batch Size	480_000
Learning Rate	0.0004
Training steps	3_000_000_000
Number of Parallel Environments	8000
Minibatch Size	48_000
Discount Factor	0.99
Number of Updates Per Batch	10
Entropy Cost	0.0007
λ_{GAE}	0.8
ε_{clip}	0.2

TABLE II: Learning setup Hyperparamters

Name	**Coefficient**
Vertical Velocity Penalty	2
Pitch/Roll Angular Velocity Penalty	0.05
Pitch/Roll Angular Position Penalty	10.0
Joint Position Limit Penalty	100.0
Joint Velocity Penalty	5.0
Joint Acceleration Penalty	0.00000025
Joint Torque Penalty	0.0002
Action Rate Penalty	0.1
Air Time Penalty	1.0
Base Height Penalty	30.0
Symmetry Air Penalty	1.0
Feet Clearance Penalty	3.0
Geometry Below Ground Penalty	100.0
Yaw Angular Velocity Penalty	0.4
Front Heading Penalty	0.8
Reached Goal Reward	60.0
Lidar Contact Penalty	10.0

TABLE III: The table presents a comprehensive list of penalty terms used in the reward function, each designed to discourage specific unwanted behaviors in the robot's operation. The names of the penalties provide intuition about what they aim to penalize, while the coefficients indicate the magnitude of each penalty relative to others. All penalties are multiplied by the simulation time step $dt = 0.02$ seconds.

Parameter	**Value**	**Unit**
1. Proprioceptive Noise		
Uniform Joint Position Noise	0.01	rad
Uniform Joint Velocity Noise	0.5	rad/s
Uniform Trunk Angular Velocity Noise	0.2	rad/s
Uniform Gravity Vector Noise	0.05	rad
2. Exterocpetive Noise		
Uniform Z Offset Noise	0.005	m
Uniform Tilt Roll Noise	0.005	rad
Uniform Tilt Pitch Noise	0.005	rad
Gaussian Std Z Noise	0.005	m

TABLE IV: Observation Noise.
1. Proprioceptive Noise Parameters. All noises are applied each control step. The observation of joint velocities are given by the relation $\frac{\dot{q}_{curr}}{\dot{q}_{max}}$. To this, we add noise scaled by the joint velocity Noise parameter. Similarly, Trunk Angular Velocity Noise modifies the actual angular velocity with a random variation based on the given parameter. This same approach applies to gravity vector noise.
2. Exteroceptive Noise Parameters. Z Offset Noise is applied each run as an offset value onto each point of the height map. Tilt Roll and Pitch Noise applies a random orientation to the height map using roll and pitch angles each step. Gaussian Std Z Noise introduces noise by applying Gaussian noise to every grid point's Z value each step. In this application, the actual height serves as the mean for the respective Gaussian distribution, with the parameter 'Gaussian Std Z Noise' defining the Standard Deviation of that distribution

Category	**Range / Value**	**Description**
Action Delay		
Action Delay Steps	1	Maximum number of delayed steps
Delay Probability	0.05	Probability of action delay
Control Randomization		
Motor strength	0.5 – 1.5	Range of motor strength
p_gain_factor	0.8 – 1.2	Range of scaling factor for p_gain
d_gain_factor	0.8 – 1.2	Range of scaling factor for d_gain
asymmetric_factor	0.98 – 1.02	Range of asymmetry factor
position_offset	-0.02 – 0.02	Range of joint position offset
Model Randomization		
Trunk Mass	-2.0 – 2.0	Additional trunk mass
COM Displacement	-0.01 – 0.01	Center of mass displacement
Foot size	0.018 – 0.026	Range of foot size
Joint Damping	0.0 – 0.1	Range of joint damping
Joint Armature	0.005 – 0.015	Range of joint armature
Joint Stiffness	0.0 – 0.0	Range of joint stiffness
Joint Friction	0.0 – 0.4	Range of joint friction loss
Friction Tangential Factor	1.0	Tangential friction factor
Friction Torsional Factor	1.0	Torsional friction factor
Friction Rolling Factor	1.0	Rolling friction factor
Damping Factor	0.6	Damping scaling factor
Stiffness	0.5	Stiffness scaling factor
Improvement ration	1.0	Additional improvement ratio
XY Gravity	0.5	Gravity in XY direction
Z Gravity Factor	0.1	Gravity in Z direction
Trunk Geom size	0.05 – 0.1	Trunk geometry size range (approximated with capsule)
Trunk Geom Length	0.05 – 0.15	Trunk geometry length range (approximated with capsule)
Lidar Geometry Size	0.5	Lidar geometry size scaling
Perturbation Randomization		
Push Velocity In X	-0.7 – 0.7	Range of perturbation in X
Push Velocity In Y	-0.7 – 0.7	Range of perturbation in Y
Push Velocity In Z	-0.7 – 0.7	Range of perturbation in Z
Addition Probability	0.5	Probability to add push velocity to current velocity
Additive Multiplier	1.5	Scaling factor if velocity is added to current velocity
Initial Robot Position Randomization		
Joint Position Noise	0.5	Noise on initial joint positions
Joint Position Offset	0.01	Offset on initial joint positions
Joint Velocity Scale	0.5	Scaling of initial joint velocities
Linear Velocity Scale	0.5	Noise on initial linear velocity
Angular Velocity Scale	0.5	Noise on initial angular velocity
Roll Angle Noise	0.15	Noise on initial roll angle
Pitch Angle Noise	0.15	Noise on initial pitch angle
Yaw Angle Noise	3.14	Noise on initial yaw angle

TABLE V: Domain Randomization parameters grouped by category. There are two different ways of randomization. 1. Predefined Range: The parameter is randomized in between the range. Starting in the middle of the range at curriculum level 0 and using the full range at curriculum 1.0 2.Factor: The parameter is randomized using the initial value of the parameter and multiplied by the factor $[1.0 - c \times f, 1.0 + c \times f]$. Hereby is f the given factor. Hereby the factor is multiplied with the given curriculum factor c