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

CaT: Constraints as Terminations for Legged Locomotion Reinforcement Learning

Elliot Chane-Sane*[1], Pierre-Alexandre Leziart*[1], Thomas Flayols[1],
Olivier Stasse[1,2], Philippe Souères[1], Nicolas Mansard[1,2]

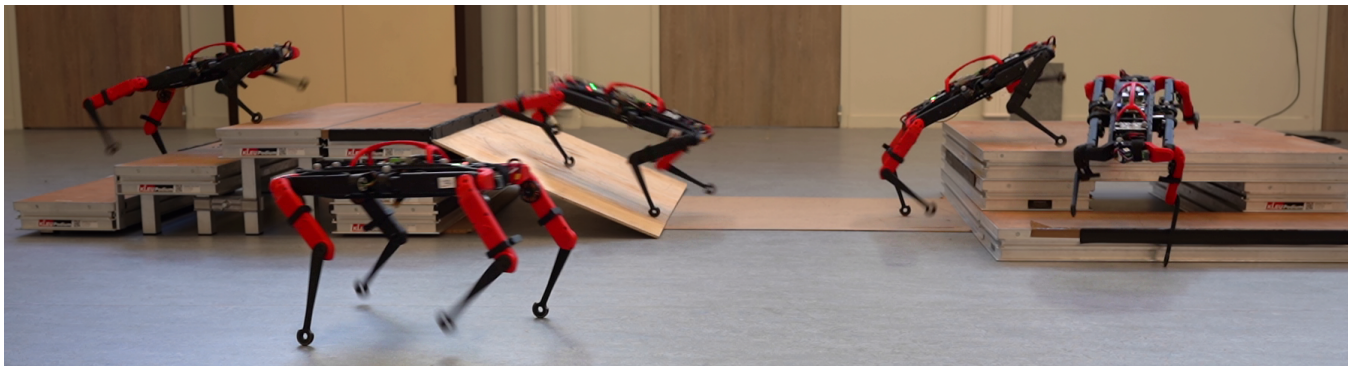

Fig. 1: The open-hardware quadruped robot Solo-12 trained with CaT performing agile locomotion over challenging terrains while satisfying safety and style constraints. The robot can walk up stairs, traverse slopes, and climb over high obstacles.

***Abstract*— Deep Reinforcement Learning (RL) has demonstrated impressive results in solving complex robotic tasks such as quadruped locomotion. Yet, current solvers fail to produce efficient policies respecting hard constraints. In this work, we advocate for integrating constraints into robot learning and present *Constraints as Terminations (CaT)*, a novel constrained RL algorithm. Departing from classical constrained RL formulations, we reformulate constraints through stochastic terminations during policy learning: any violation of a constraint triggers a probability of terminating potential future rewards the RL agent could attain. We propose an algorithmic approach to this formulation, by minimally modifying widely used off-the-shelf RL algorithms in robot learning (such as Proximal Policy Optimization). Our approach leads to excellent constraint adherence without introducing undue complexity and computational overhead, thus mitigating barriers to broader adoption. Through empirical evaluation on the real quadruped robot Solo crossing challenging obstacles, we demonstrate that CaT provides a compelling solution for incorporating constraints into RL frameworks. Videos and code are available at constraints-as-terminations.github.io.**

I. Introduction

Deep reinforcement learning (RL) has proven highly effective in crafting control policies for complex robotic tasks. In quadruped locomotion, RL approaches have demonstrated high performances to train policies capable of traversing challenging terrains [1], [2], [3], [4] and generating natural, animal-like motions [5], [6], [7]. In this work, we follow recent successful approaches based on model-free RL [8] to train policies on a curriculum of increasingly difficult settings [9], [10] in simulation and directly transfer the learned policy on the physical robot [11], [12], [13] to overcome challenging obstacles. Compared to previous approaches in robot motion [14], [15], [16], this workflow requires minimal design choices, relying on generic algorithms and simulations that allow to generate a wide variety of tasks.

Yet, reward shaping remains a meticulous endeavor as it demands a delicate balance between accomplishing the desired task, adhering to physical limitations, enabling seamless sim-to-real transfer, and ensuring natural and efficient motions. Many of these terms could be more effectively and intuitively formulated as constraints. For instance, joint torque and velocity limits have clear physical meanings that should not be considered through a hyperparameter search. While incorporating such constraints aligns with common practices in model-based control [17], [18], [19], widespread adoption in robot learning has been limited. Although some recent constrained RL methods have been applied to locomotion [20], [21], they often simplify reward engineering at the cost of algorithmic complexity, as additional critic networks and terms in the policy loss function have to be implemented.

In this work, we propose *Constraints as Terminations (CaT)*, a streamlined approach for constrained RL that prioritizes simplicity and flexibility. We introduce constraints through stochastic terminations during policy learning: any violation of a constraint leads to a probability of terminating the future rewards the RL agent could have achieved. To do so, we down-scale all the future rewards based on the magnitude of the constraint violations during policy learning through the discount factor. This naturally encourages the agent towards satisfying the constraints to maximize future rewards, while providing an alternative reward signal to recover from constraint violations. This principle can be seen as a refined extension of the common practice of using a

*Equal contribution
[1]LAAS-CNRS, Université de Toulouse, Toulouse, 31400, France first.last@laas.fr
[2]Artificial and Natural Intelligence Toulouse Institute, Toulouse, France.

straightforward termination function, leveraging stochastic termination to yield a dense feedback to the policy.

Our approach is simple to implement and seamlessly integrates with existing off-the-shelf RL algorithms. In our experiments, we instantiate CaT with Proximal Policy Optimization [8] (PPO), a model-free on-policy algorithm widely used in robot learning. We design a set of constraints to ensure that the learned policy can be safely deployed to the real robot, and a set of style constraints to exhibit natural motions. We demonstrate the effectiveness of our approach by deploying locomotion policies on a Solo quadruped robot with height-scan observations, producing agile locomotion skills capable of traversing challenging terrains composed of stairs, a steep slope and a high platform (see Fig. 1).

In summary, our contributions are the following:

1) we introduce stochastic terminations as a way to shape the behavior of the policy to satisfy constraints in a minimalist fashion,
2) we propose constraint designs to enforce safe behaviors and make the policy adhere to a specific walking style on flat terrains, while letting RL adapt the style on rougher terrains,
3) and we validate our approach on a real Solo quadruped robot to overcome diverse obstacles in a parkour while satisfying safety and style constraints.

II. Related Work

Reinforcement learning has emerged as a particularly effective method for obtaining agile and adaptive policies for quadruped robots. While some approaches attempt to train RL locomotion policies directly on physical quadruped robots by leveraging sample-efficient RL techniques [22], [23], a popular approach entails training policies in simulation before transferring them to the real world [24], [25], [26], [27]. This transfer relies on accurate physics simulators and domain randomization to ensure policy transferability to the physical robot [28], [29], [30]. Recently, GPU-based simulators capable of simulating thousands of robots in parallel [31], [32], [33] have streamlined this process [11]. The resulting policies exhibit natural, animal-like motions and can adapt to challenging terrain configurations [34], [35], [2], [1], [4], [3], [36]. In our experiments, we follow this sim-to-real approach and deploy our policies on the Solo-12 robot [16], [37] for challenging terrain traversal.

Incorporating constraints is a common practice in model-based control, where their importance to ensure robot safety is commonly accepted [38], [39], [40]. Yet constraints have garnered limited attention in the RL community, where the main effective solvers do not readily consider them [8], [41] and achieving policies that comply with constraints is often done through intricate reward shaping. In legged locomotion, this approach typically results in reward functions comprising numerous terms that are labor-intensive to tune. For instance, the reward functions used in [11], [26] comprise a dozen of terms. Moreover, the resulting policy, being a compromise among maximizing each of these terms, is not guaranteed to satisfy constraints in all situations [21].

Prior works have explored the imposition of constraints or safety mechanisms in addition to rewards within the learning process to ensure safety guarantees. Recovery policies have been learned jointly with the locomotion policy to address safety violations [42], [43]. [44], [45] proposed to shield the learning agent by directly substituting policy actions by safe actions whenever necessary to prevent constraint violations. Other approaches incorporate constraint satisfaction directly into the policy optimization algorithms by adjusting the policy update rules to discourage violations. For instance, Lagrangian methods [46], [47] approach constrained problems as unconstrained ones by introducing Lagrange multipliers, but this often leads to instability due to hyperparameter sensitivity [48]. More closely related to our work, [20] modifies the Interior-point Policy Optimization algorithm [49] and demonstrate quadruped locomotion skills on rough-terrain whereas [21] implements a modified Penalized Proximal Policy Optimization (P3O) [50] algorithm on a wheeled quadruped robot, both showcasing enhanced safety in the learned policies and facilitating the tuning of reward terms at the cost of additional algorithmic complexity. By contrast, our approach is simple to implement, requiring minimal changes to existing locomotion RL pipelines and introducing no additional computational overhead.

Terminating the future rewards and resetting the episode is ubiquitously used in reinforcement learning to avoid certain behaviors. For instance, [11] terminates the episode with a low reward when the robot base or knees touch the ground. [51] further showed that learning policies for early-terminated Markov decision processes (ET-MDP), i.e. terminating future rewards on constraint violations without necessarily resetting the environment, is an effective way to learn constraint-satisfying policies. However, our experiments highlight that this approach does not readily scale to complex systems such as quadruped robots with dozens of constraints. We propose in the next section to capitalize on this common practice to design a novel approach to enforce generic hard constraints in RL. To that end, we first reformulate the constraint as a probability of satisfaction. Then we introduce stochastic terminations as a way to downscale the sum of future possible rewards while keeping a dense feedback to the policy, in particular by keeping informative direction from the domain outside constraint satisfaction.

III. Method

A. Problem Formulation

We consider an infinite, discounted Markov Decision Process $\mathcal{S}, \mathcal{A}, r, \gamma, \mathcal{T}$ with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function r, discount factor γ and dynamics $\mathcal{T}$. RL aims to find a policy π that maximizes the discounted sum of future rewards:

$$\max_{\pi} \mathbb{E}_{\tau \sim \pi, \mathcal{T}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \tag{1}$$

In the following, we assume positive rewards $r \geqslant 0$ for simplicity (without loss of generality w.r.t. any other lower

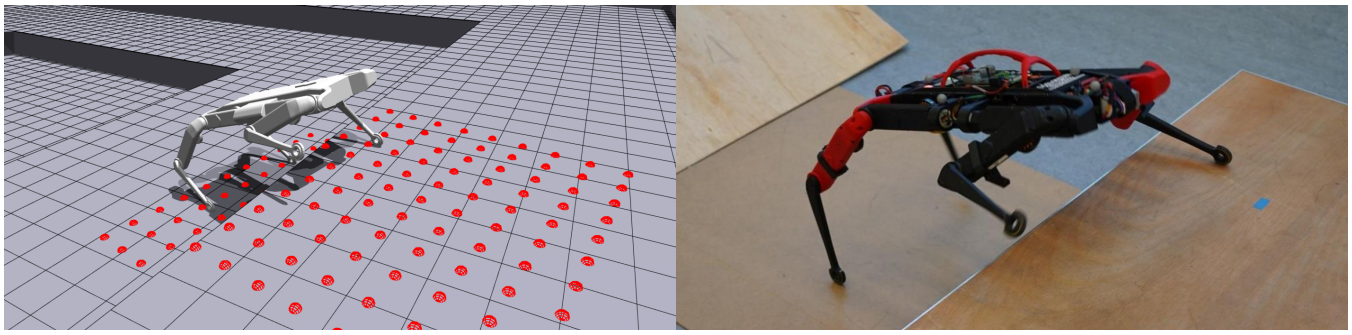

Fig. 3: (Left) The quadruped robot is trained with CaT in simulation using height-map scan. (Right) The learned policy is directly deployed on the real robot. Knowing the obstacle course on which the robot is placed, we use external motion capture cameras to reconstruct the height-map of its surroundings based on its position and orientation in the world.

bounded definition). Constrained RL additionally introduces a set of constraint functions $\{c_i : \mathcal{S} \times \mathcal{A} \to \mathcal{R}, i \in I\}$ and aims to maximize rewards while limiting the discounted sum of constraints over the trajectories generated by the policy:

$$\mathbb{E}_{\tau\sim\pi,\mathcal{T}}\left[\sum_{t=0}^{\infty}\gamma^t c_i(s_t,a_t)\right] \leq \epsilon_i \ \forall i \in I. \tag{2}$$

While standard in the RL literature [48], [20], this formulation includes a notion of budget for the constraints. We consider instead maximizing rewards while avoiding constraint violation at each time step: $\mathbb{E}_{\tau\sim\pi,\mathcal{T}}\left[\sum_{t=0}^{\infty}\gamma^t 1_{c_i(s_t,a_t)>0}\right] \leq \epsilon_i$, where $1_{\gamma^t c_i(s_t,a_t)>0}$ indicates whether the i-th constraint has been violated at time t. This is equivalent to:

$$\mathbb{P}_{(s,a)\sim\rho_\gamma^{\pi,\mathcal{T}}}\left[c_i(s,a) > 0\right] \leq \tilde{\epsilon_i} \ \forall i \in I, \tag{3}$$

where $\rho_\gamma^{\pi,\mathcal{T}}$ corresponds to the discounted state-action occupancy distribution of the policy π. While this corresponds to a special case of the more general constrained RL setting, this formulation, akin to chance-constrained optimization [52], [53], encompasses many practical applications of RL for robotic control.

B. Constraints as Terminations

1) Reformulation: Instead of directly solving (1) under the constraints (3), we propose to reformulate it as:

$$\max_{\pi} \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\left(\prod_{t'=0}^{t}\gamma(1-\delta(s_{t'},a_{t'}))\right)r(s_t,a_t)\right], \tag{4}$$

where we introduce a random variable $\delta_t : \mathcal{S} \times \mathcal{A} \to [0,1]$ indicating whether the episode terminates and the future rewards are terminated from time step t. Importantly, we propose to design δ_t as a function of the constraint violations c_i. Note that episode terminations are **not** environment resets, but merely future reward terminations from a policy learning perspective. Under the expectation, the Bernoulli variable and its probability are the same. In the rest of the paper, δ_t will refer directly to the probability of termination.

2) Naive termination: A naive approach is to terminate the future rewards if any constraint is violated [51] with the following binary function for δ:

$$\delta = 1 - \prod_{i\in I} 1_{c_i\leq 0}. \tag{5}$$

[51] showed that if the minimum value of the rewards is high enough, which can be easily obtained by adding a high enough constant value, the learned policy will satisfy the constraints. However, terminating the episode if any constraint is violated might be overly conservative with respect to the constraints and impair exploration and learning. Moreover, such a termination condition offers a sparse signal for recovering from constraint violations: once the agent enters a region of constraint violation, the episode always terminates and the agent does not learn anything.

3) Stochastic terminations: We propose that δ_t can take values beyond 0 or 1 depending on the constraint violations at time t. As a result, any violation of a constraint leads to a probability of terminating the future rewards the RL agent could have achieved. If no constraints are violated, then the episode terminates with a probability of zero, whereas if one or more constraints are violated, δ may take positive values between 0 and 1. In that case, the sum of all future rewards at t and after time are re-scaled by $(1-\delta_t)$. Therefore, in order to maximize the sum of future rewards, the agent naturally gravitates towards satisfying the constraints. Allowing δ to take values in $]0,1[$ enables the agent to learn to recover from constraint violations. Moreover, depending on the value of δ, this allows some exploration inside the region of constraint violation.

By designing δ such that it is increasing with c_i, the termination probability will provide a dense signal to the learning algorithm to recover from constraints. Driven by simplicity, we propose the following termination probability function:

$$\delta = \max_{i\in I} p_i^{\max}\text{clip}\left(\frac{c_i^+}{c_i^{\max}}, 0, 1\right), \tag{6}$$

where $c_i^+ = \max(0, c_i(s,a))$ is the violation of constraint i, $c_i^{\max}$ is an exponential moving average of the maximum constraint violation over the last batch of experience collected in the environment:

$$c_i^{\max} \leftarrow \tau^c c_i^{\max} + (1-\tau^c)\max_{(s,a)\in\text{batch}} c_i^+(s,a), \tag{7}$$

with decay rate $\tau^c \in]0,1[$ and $p_i^{\max}$ a hyperparameter that controls the maximum termination probability for the constraint i, indirectly setting $\tilde{\epsilon_i}$ in (3). We found directly using

the maximum over the batch of experience without exponential moving average to be slightly less stable. Hence, the termination probability for each constraint is proportional to the magnitude of the constraint violation, while the dynamic update of $c_i^{\max}$ makes sure that the termination function always provides a relevant learning signal throughout training. We found that this design was simple to implement while achieving effective constraint satisfaction.

Algorithm 1 Implementation of CaT with PPO, with alterations from the original RL algorithm highlighted in red.

for epoch $= 1$ **to** N **do**
  data $\leftarrow$ PPO.collect_trajectories()
  compute δ(data.constraints) using (6)
  data.rewards $\leftarrow$ data.rewards $\times (1 - \delta)$
  data.dones $\leftarrow \delta$
  PPO.update_policy(data)
end for

Our proposed approach, *Constraints as Terminations (CaT)*, can easily be incorporated into existing RL algorithms with minimal changes, by simply computing δ based on the constraint violations using (6), multiplying the rewards by δ and rewriting the terminations with δ. These modifications can be implemented with just a few lines of codes to existing RL algorithms. Algorithm 1 highlights the changes needed to implement our approach on top of PPO.

IV. Application to Legged Locomotion

We train a policy in simulation using CaT and directly transfer the policy to a real Solo-12 robot (see Fig. 3). For this quadruped locomotion problem, the state space $\mathcal{S}$ corresponds to the measured angular velocity ω, the gravity vector projected in base frame, the measured positions q_t and velocities $\dot{q}_t$ of all 12 joints of the robot, the previous action a_{t-1} and the linear and angular velocity commands v_{xy}^{des} and ω_z^{des} that the robot must track. For non-blind navigation, the robot also observes the height-scan h_{scan} of its surroundings. The action space $\mathcal{A}$ corresponds to desired joint position offsets $a_t = \Delta q_t^{\text{des}}$ with respect to a default joint configuration $q^\star$, that are then converted to torques through a proportional-derivative (PD) controller operating at a higher frequency than the neural policy. The derivative part of the controller aims to bring the joint velocity to zero.

One might consider that each reward and constraint can serve one of these three purposes:

- define the task to be achieved,
- ensure that the generated trajectories are safe and transferable to the physical robot,
- or impose a style to the generated motions.

The complete list of rewards and constraints used in our experiments is provided in Table I. We detail them below.

a) Task definition: The legged locomotion task is to track the linear velocity command in horizontal direction v_{xy}^{des} and yaw rate ω_z^{des}. We consider a velocity tracking reward function widely used in RL for legged locomotion (Option A) [11], [21]. Alternatively, we propose to define the velocity tracking task as a constraint to be satisfied (Option B).

Task formulation: through rewards (Option A)	
Reward function	$r = e^{-\frac{\left\|v_{xy}^{\text{des}} - v_{xy}\right\|_2^2}{0.25}} + \frac{1}{2} e^{-\frac{\left\|\omega_z^{\text{des}} - \omega_z\right\|^2}{0.25}}$
Task formulation: through soft constraints (Option B)	
Reward function	$r = 1$
Linear velocity tracking	$c_{\text{lin vel}} = \left\|v_{xy}^{\text{des}} - v_{xy}\right\|_2 - \epsilon_{\text{track}}$
Angular velocity tracking	$c_{\text{ang vel}} = \left\|\omega_z^{\text{des}} - \omega_z\right\| - \epsilon_{\text{track}}$
Hard constraints for safety	
Knee or base collision	$c_{\text{knee/base contact}} = 1_{\text{knee/base contact}}$
Foot contact force	$c_{\text{foot contact}_j} = \|f^{\text{foot}_j}\|_2 - f^{\text{lim}}$
Soft constraints for safety ($\forall k \in 1..12$)	
Torque limits	$c_{\text{torque}_k} = \|\tau_k\| - \tau^{\text{lim}}$
Joint velocity limits	$c_{\text{joint velocity}_k} = \|\dot{q}_k\| - \dot{q}^{\text{lim}}$
Joint acceleration limits	$c_{\text{joint acceleration}_k} = \|\ddot{q}_k\| - \ddot{q}^{\text{lim}}$
Action rate limits	$c_{\text{action rate}_k} = \frac{\left\|\Delta q_{t,k}^{\text{des}} - \Delta q_{t-1,k}^{\text{des}}\right\|}{dt} - \dot{q}^{\text{des lim}}$
Soft constraints for style (Active on flat terrains only, $\forall j \in 1..4$)	
Base orientation	$c_{\text{ori}} = \|\text{base ori}_{\text{xy}}\|_2 - \text{base}^{\text{lim}}$
Hip orientation	$c_{\text{hip}_j} = \|\text{hip ori}_j\| - \text{hip}^{\text{lim}}$
Foot air time	$c_{\text{air time}_j} = t_{\text{air time}}^{\text{des}} - t_{\text{air time}_j}$
Number of foot contacts	$c_{\text{n foot contacts}} = \|n_{\text{foot contact}} - n_{\text{foot contact}}^{\text{des}}\|$
Stand still if $v^{\text{des}} = 0$	$c_{\text{still}} = (\|q - q^\star\|_2 - \epsilon_{\text{still}}) \times 1_{v^{\text{des}}=0}$

TABLE I: Rewards and constraints used in our experiments.

b) Safety constraints: Safety constraints are defined to ensure the policy learned in simulation will transfer well and safely to the physical robot once training is complete. We prohibit collisions to the knee and the base of the robot to avoid dangerous behaviors that might destroy the robot. We limit the contact force of each foot n to prevent the robot from hitting the ground too harshly, and we limit the torque applied to each joint k to prevent damaging the actuators. To ensure that the generated motions are smooth for seamless sim-to-real transfer, we also limit joint velocities, joint accelerations and action rates.

c) Style constraints: Style constraints are used to guide learning towards natural-looking motions. However, defining relevant style constraints in any terrain configuration is difficult. We propose to enforce style constraints only on flat surfaces while deactivating them (i.e. set them to 0) otherwise. This allows us to define a precise style to follow on flat terrains while providing room for the RL algorithm to adapt the learned behavior on more challenging terrains. In our implementation, the terrain is considered flat if the variance of the scan dots is below a certain threshold $\text{var}(h_{\text{scan}}) < \text{var}_{\text{scan}}^{\text{lim}}$. We limit the orientation of the base and the angle of the hips. When the velocity command is above a threshold, we additionally ground the flying phase duration of each foot and limit to two the number of foot contacts with the ground whereas, if no velocity is provided, we force the robot to go back to its default pose.

d) Soft and hard constraints: Our method introduces a hyperparameter $p_i^{\max}$ for each constraint which trades off exploration with constraint satisfaction. A high value of $p_i^{\max}$ will ensure that the constraint is strictly satisfied during

Method	Rewards	Cstr.
Hard constraints only	0	$n.a.$
ET-MDP [51]	0	$n.a.$
N-P3O [50], [20], [21]	593.2 ($\pm$ 49.5)	8% ($\pm$ 1%)
CaT (Tracking Rewards)	**682.9** ($\pm$ 5.8)	**0.5%** ($\pm$ 0.3%)

TABLE II: Average sum of rewards (*Rewards*) and average time proportion of torque constraint violation for any joint (*Cstr.*) achieved by the policies on flat terrain in simulation. Results are averaged over 4 training seeds.

training but might lead to overly conservative exploration, whereas lower values of $p_i^{\max}$ will allow the learning agent to discover higher reward regions of the behavior space. Motivated by simplicity, we propose to classify constraints into two groups: *hard constraints* with $p_i^{\max} = 1.0$ for constraints that should never be violated, and *soft constraints*, where $p_i^{\max}$ increases from 0.05 to 0.25 throughout the course of training, that the RL algorithm might violate during exploration and learn to recover from. We found that this design allowed the agent to maximally learn complex locomotion skills while further enforcing the constraints in the later stage of training. In our experiments, base or knee contact collisions and foot contact forces are defined as hard constraints and the rest of the constraints as soft ones.

This set of constraints results in a large constraint vector comprising more than 60 terms. While prior approaches group constraints together [20], [21], we found that this additional engineering burden was unnecessary for CaT.

V. Experiments

A. Experimental setup

To train our policies, we leverage the PPO algorithm [8] using the implementation from rl-games [54], which we slightly modified to accommodate non-boolean terminations, alongside massively parallel simulation of Isaac Gym [33]. Hyperparameters are provided in Appendix VI-C. Blind policies for flat terrains are trained for 2000 epochs whereas policies with height-scan map are trained for 20000 epochs for agile terrain traversal. This amounts to respectively 1 hour and 10 hours of training on a single V100 GPU. Except for CaT specific implementations, the resulting training procedure is similar to [11].

After training in simulation, the controller is directly deployed on a real Solo-12 robot. The policy runs at 50 Hz on a Raspberry Pi 4 Model B using a custom C++ implementation. Target joint positions are sent to the onboard PD controller running at 10 kHz. PD gains are kept low to obtain a compliant impedance controller that will achieve a behavior close to torque control and will be able to dampen and absorb impacts [26]. This is further made possible thanks to the transparent actuation of Solo-12. For more details on the hardware, please refer to [16], [37]. Instead of directly capturing a height-scan map of the robot's surrounding terrain, we use motion capture to track the position of the robot and sample the corresponding height map points.

To validate the agility of the learned policies in diverse scenarios, we evaluate our approach on a challenging obstacle parkour comprising a set of stairs, a slope and a platform roughly the height of the robot (see Fig. 1). Following Table I, we consider two versions of CaT: one with the task defined through rewards (*CaT (Tracking Rewards)*) and one with the task defined through constraints (*CaT (Tracking Constraints)*). We compare CaT to the following baselines:

- *ET-MDP*: a modification of our method designed to resemble [51] by using (5) to compute δ.
- *N-P3O*: our reproduction of P3O [50] using techniques from [20], [21].
- *Hard constraints only*: an ablation of our approach where we use $p_i^{\max} = 1.0$ for all constraints.
- *Style always active*: an ablation of our approach where style constraints are always enforced.

For N-P3O, we group constraints of the same type together following [20], [21], use dense constraint functions as in CaT as opposed to indicator functions used in [20], [21], and employ foot phase duration and number of foot contacts as rewards rather than constraints, as N-P3O struggles to converge otherwise. Solo-12 is a light robot with dynamic, but limited actuators that should avoid applying a torque of more than 3Nm. To evaluate the capabilities of our approach to enforce constraints, we focus on the torque constraint satisfaction and report the proportion of time where this constraint is violated for one or more joints.

B. Results and Analysis

We first compare *CaT (Tracking Rewards)* to N-P3O, ET-MDP and *Hard constraints only* trained on a flat terrain for blind locomotion in simulation. Table II reports the rewards and the torque constraints satisfaction achieved by the policies. ET-MDP entirely fails to learn locomotion policies in our high-dimensional constraint problem. This may be due to the fact that at the beginning of training, the robot always violates some constraints, preventing any reward or constraint feedback to allow policy learning. Similarly, when the constraints are enforced too roughly (*Hard constraints only*), learning fails completely, as overly stringent enforcement of constraints hinders exploration and learning. Despite being simpler, CaT outperforms N-P3O, in both the sum of tracking rewards attained and the satisfaction of torque constraints after 2000 epochs of training. We hypothesize that the integration of rewards and constraints into a unified RL framework allows CaT to learn faster.

Next, we deploy CaT with height-scan map on the real robot. In Table III, we report the success rate of traversing each obstacle in the parkour. CaT with both sets of rewards and constraints successfully learns agile locomotion skill to overcome each obstacle of the parkour. Fig. 1 shows a full traversal of the obstacle parkour, demonstrating natural motions on flat surfaces while achieving agile skills on more challenging obstacles. Notably, CaT successfully learns to overcome all the obstacles while satisfying the torque constraint. Fig. 4 shows that, while climbing on the platform almost as high as the robot, the torque remains within the limit set during training. Interestingly, *CaT (Tracking Constraints)*, where the locomotion task is defined entirely

Method	Front Stairs		Sideways Stairs		Slope		Platform		Average	
	Succ.	Cstr.	Succ.	Cstr.	Succ.	Cstr.	Succ.	Cstr.	Succ.	Cstr.
Style always active	50.0%	2.5%	40.0%	4.8%	30.0%	2.0%	17.5%	5.4%	34.4%	3.7%
CaT (Tracking Rewards)	**100.0%**	**0.08%**	42.5%	**0.3%**	**97.5%**	**0.3%**	77.5%	**1.1%**	79.4%	**0.5%**
CaT (Tracking Constraints)	**97.5%**	0.5%	**85.0%**	1.8%	**95.0%**	1.2%	**85.0%**	3.4%	**90.6%**	1.7%

TABLE III: Average success rate (*Succ.*) and average time proportion of torque constraint violation for any joint (*Cstr.*) achieved by the policies on the different obstacles of the parkour on the real robot: walking up the stairs from the front (*Front Stairs*) and sideways (*Sideways Stairs*), walking up the slope (*Slope*) and walking up the platform as high as the robot's base (*Platform*). Results are averaged over 4 random training seeds and 10 attempts per obstacle per seed.

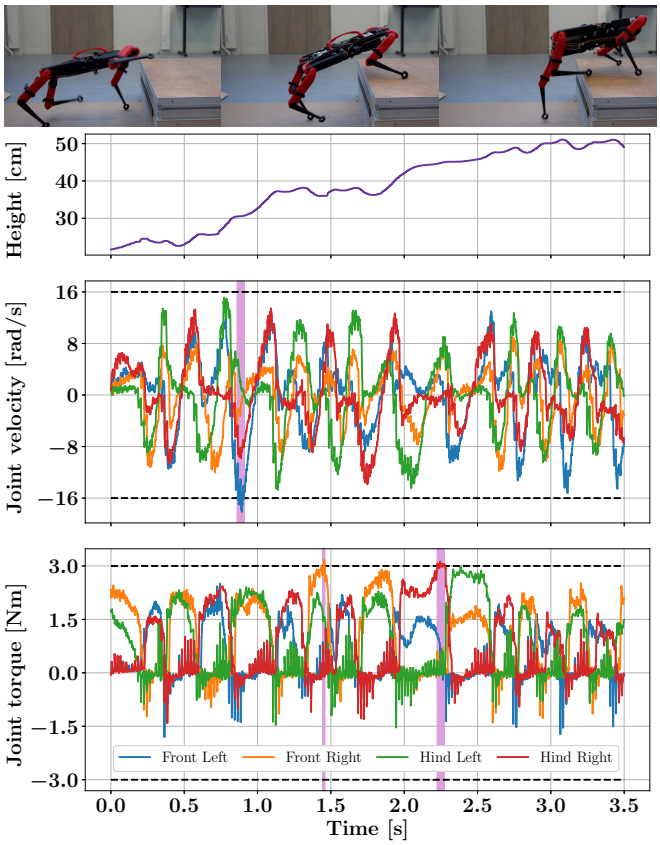

Fig. 4: Joint torques and velocities during the climb of a 24 cm platform. For clarity, we only report data for the knee joints, which had the highest torque peaks.

through constraints, learns agile locomotion skills. In particular, it outperforms *CaT (Tracking Rewards)* on climbing the stairs sideways, a difficult task where the height-scan map provides less visibility. By contrast, *CaT (Tracking Rewards)* often refuses to walk over the stairs sideways while achieving similar performances on other obstacles. We hypothesize that *CaT (Tracking Constraints)* is more prone to explore unsafe behaviors to fulfill the task constraints, resulting in better success rates at the expense of more constraint violations. This highlights how stochastic termination functions can be used to appropriately shape the behavior of the robot policy, either to ensure the controller is safe and adhere to a certain style, but also to fully define the intended task for the robot.

We then compare CaT to always enforcing style constraints, even on challenging terrains (*Style always active*). While this approach successfully learns walking skills on flat and rough terrains, it struggles on more difficult obstacles. This occurs because adhering strictly to certain style constraints, as defined on flat surfaces, may not be compatible with other scenarios. For example, imposing the constraint that the robot's base must remain horizontal is incompatible with scenarios involving stair climbing. This is particularly striking when attempting to climb the platform, which requires tilting the base and lift the shoulders, as illustrated in Fig. 4 (top).

Fig. 5: CaT trained with a constraint that limits the height of the base learns crouching locomotion skills.

In Fig. 5, we illustrate how simply adding a constraint to limit the height of the base ($c_{\text{height}} = \text{height}_{\text{base}} - \text{height}^{\text{max}}_{\text{base}}$) can learn crouching locomotion skills on the quadruped. Videos of the robot traversing the parkour and crouching are available in the supplementary video.

VI. Conclusion

In this study, we introduce *CaT*, a novel and minimalist algorithm addressing constraints in reinforcement learning. We formulate the problem so that the probability of constraint violation is bounded and use stochastic termination to seamlessly integrate it on top of standard algorithms such as PPO. On a Solo-12 quadruped robot, CaT successfully manages to learn agile locomotion skills on challenging terrain traversals, showcasing its utility in enforcing safety and stylistic constraints within quadruped locomotion. Future work could explore more principled ways to define the termination conditions based on the constraints.

From a practical standpoint, constrained RL significantly simplifies the reward engineering process. However, unlike previous, more intricate methods, our approach is notably simpler to implement, necessitating minimal code adjustments and is devoid of any computational overhead. We hope the effectiveness and simplicity of our approach will foster the democratization of constrained RL in robotics.

ACKNOWLEDGEMENTS

This work was funded in part by ANITI (ANR-19-P3IA-0004), COCOPIL (Région Occitanie, France), PEP# O2R (AS2 ANR-22-EXOD-0006), Dynamograde (ANR-21-LCV3-0002) and ROBOTEX 2.0 (ROBOTEX ANR-10-EQPX-44-01 and TIRREX-ANR-21-ESRE-0015). It was granted access to the HPC resources of IDRIS under the allocations 2021-AD011012947 and 2023-AD011014301 made by GENCI.

REFERENCES

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

[2] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," *arXiv preprint arXiv:2309.14341*, 2023.

[3] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.

[4] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," in *Conference on Robot Learning (CoRL)*, 2023.

[5] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[6] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel, "Adversarial motion priors make good substitutes for complex reward functions," in *2022 IEEE/RSJ IROS*, 2022, pp. 25–32.

[7] T. Li, Y. Zhang, C. Zhang, Q. Zhu, J. Sheng, W. Chi, C. Zhou, and L. Han, "Learning terrain-adaptive locomotion with agile behaviors by imitating animals," in *2023 IEEE/RSJ IROS*, 2023, pp. 339–345.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[9] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[10] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, "Curriculum learning: A survey," *International Journal of Computer Vision*, vol. 130, no. 6, pp. 1526–1565, 2022.

[11] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*, 2022.

[12] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE, 2023, pp. 1–8.

[13] G. Bellegarda and A. Ijspeert, "Visual cpg-rl: Learning central pattern generators for visually-guided quadruped navigation," *arXiv preprint arXiv:2212.14400*, 2022.

[14] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *2003 IEEE ICRA*, vol. 2, 2003, pp. 1620–1626.

[15] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 93–100.

[16] P.-A. Léziart, T. Flayols, F. Grimminger, N. Mansard, and P. Souères, "Implementation of a reactive walking controller for the new open-hardware quadruped solo-12," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5007–5013.

[17] E. Dantec, M. Naveau, P. Fernbach, N. Villa, G. Saurel, O. Stasse, M. Taix, and N. Mansard, "Whole-body model predictive control for biped locomotion on a torque-controlled humanoid robot," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*. IEEE, 2022, pp. 638–644.

[18] F. Risbourg, T. Corbères, P.-A. Léziart, T. Flayols, N. Mansard, and S. Tonneau, "Real-time footstep planning and control of the solo quadruped robot in 3d environments," in *2022 IEEE/RSJ IROS*, 2022, pp. 12 950–12 956.

[19] P.-A. Léziart, T. Corbères, T. Flayols, S. Tonneau, N. Mansard, and P. Souères, "Improved control scheme for the solo quadruped and experimental comparison of model predictive controllers," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9945–9952, 2022.

[20] Y. Kim, H. Oh, J. Lee, J. Choi, G. Ji, M. Jung, D. Youm, and J. Hwangbo, "Not only rewards but also constraints: Applications on legged robot locomotion," *IEEE Transactions on Robotics*, 2024.

[21] J. Lee, L. Schroth, V. Klemm, M. Bjelonic, A. Reske, and M. Hutter, "Evaluation of constrained reinforcement learning algorithms for legged locomotion," *arXiv preprint arXiv:2309.15430*, 2023.

[22] L. Smith, I. Kostrikov, and S. Levine, "A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning," *arXiv preprint arXiv:2208.07860*, 2022.

[23] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, "Daydreamer: World models for physical robot learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 2226–2240.

[24] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE ICRA*, 2018, pp. 3803–3810.

[25] G. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," in *Robotics: Science and Systems*, 2022.

[26] M. Aractingi, P.-A. Léziart, T. Flayols, J. Perez, T. Silander, and P. Souères, "Controlling the solo12 quadruped robot with deep reinforcement learning," *scientific Reports*, vol. 13, no. 1, p. 11945, 2023.

[27] ——, "A hierarchical scheme for adapting learned quadruped locomotion," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE, 2023, pp. 1–8.

[28] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," in *Proceedings of Robotics: Science and Systems*, 2018.

[29] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[30] Z. Xie, X. Da, M. Van de Panne, B. Babich, and A. Garg, "Dynamics randomization revisited: A case study for quadrupedal locomotion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4955–4961.

[31] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[32] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, "Brax - a differentiable physics engine for large scale rigid body simulation," 2021. [Online]. Available: http://github.com/google/brax

[33] 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," 2021.

[34] Z. Fu, A. Kumar, J. Malik, and D. Pathak, "Minimizing energy consumption leads to the emergence of gaits in legged robots," *arXiv preprint arXiv:2111.01674*, 2021.

[35] G. Bellegarda, Y. Chen, Z. Liu, and Q. Nguyen, "Robust high-speed running for quadruped robots via deep reinforcement learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 364–10 370.

[36] H. Duan, B. Pandit, M. S. Gadde, B. J. van Marum, J. Dao, C. Kim, and A. Fern, "Learning vision-based bipedal locomotion for challenging terrain," *arXiv preprint arXiv:2309.14594*, 2023.

[37] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[38] W. Jallet, A. Bambade, E. Arlaud, S. El-Kazdadi, N. Mansard, and J. Carpentier, "Proxddp: Proximal constrained trajectory optimization," 2023.

[39] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[40] S. Tonneau, D. Song, P. Fernbach, N. Mansard, M. Taïx, and A. Del Prete, "Sl1m: Sparse l1-norm minimization for contact planning on uneven terrain," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 6604–6610.

[41] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.
[42] T.-Y. Yang, T. Zhang, L. Luu, S. Ha, J. Tan, and W. Yu, "Safe reinforcement learning for legged locomotion," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2454–2461.
[43] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile but safe: Learning collision-free high-speed legged locomotion," in *arXiv*, 2024.
[44] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.
[45] K. Fan, Z. Chen, G. Ferrigno, and E. De Momi, "Learn from safe experience: Safe reinforcement learning for task automation of surgical robot," *IEEE Transactions on Artificial Intelligence*, 2024.
[46] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *Journal of Machine Learning Research*, 2018.
[47] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," *arXiv preprint arXiv:1805.11074*, 2018.
[48] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International conference on machine learning*. PMLR, 2017, pp. 22–31.
[49] Y. Liu, J. Ding, and X. Liu, "Ipo: Interior-point policy optimization under constraints," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 4940–4947.
[50] L. Zhang, L. Shen, L. Yang, S. Chen, B. Yuan, X. Wang, and D. Tao, "Penalized proximal policy optimization for safe reinforcement learning," *arXiv preprint arXiv:2205.11814*, 2022.
[51] H. Sun, Z. Xu, Z. Peng, M. Fang, T. Wang, B. Dai, and B. Zhou, "Constrained mdps can be solved by eearly-termination with recurrent models," in *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.
[52] A. Charnes and W. W. Cooper, "Chance-constrained programming," *Management science*, vol. 6, no. 1, pp. 73–79, 1959.
[53] A. Nemirovski and A. Shapiro, "Convex approximations of chance constrained programs," *SIAM Journal on Optimization*, vol. 17, no. 4, pp. 969–996, 2007.
[54] D. Makoviichuk and V. Makoviychuk, "rl-games: A high-performance framework for reinforcement learning," https://github.com/Denys88/rl_games, May 2021.

APPENDIX

A. Additional study

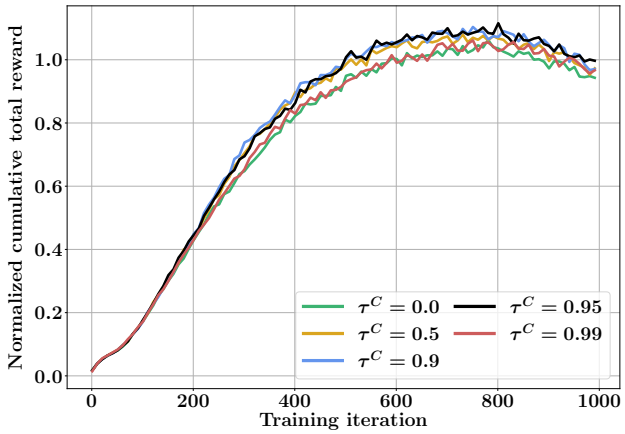

Fig. 6: Normalized cumulative reward during training for various decay rate τ^c in (7), averaged over 6 seeds. Learning performances are marginally impacted by this hyperparameter as extreme values only lead to a decrease of $\approx 5\%$. However, during the course of the project, we found that $\tau^c = 0$ sometimes led to instabilities during training.

B. Additional Limitations

While we only rarely observed failures of the policies on the real robot and the constraints were consistently satisfied, we did not test the locomotion skills at the robot's maximum agility. Pushing these limits, for instance by training the locomotion policies on terrains that include harder obstacles and that would require more dynamic movements, may challenge our approach's ability to satisfy all constraints. Additionally, our approach has not been tested beyond quadruped locomotion, and its effectiveness for manipulation and loco-manipulation is yet to be determined.

C. Hyperparameters

[54] details the meaning of some hyperparameters.

TABLE IV: Environment hyperparameters

Number of envs.	4096
Random v_x range	$[-0.3, 1.0]$ m/s
Random v_y range	$[-0.7, 0.7]$ m/s
Random ω_z range	$[-0.78, 0.78]$ rad/s
Proportional gain	4.0 Nm/rad
Derivative gain	0.2 Nm/(rad/s)
Action scaling	0.5
Default leg angles	[0.05, 0.4, -0.8] rad
Simulation time step	5 ms
Episode length	10 s
Height scan grid	13 x 11 points
Height scan step	8 cm

TABLE V: Learning hyperparameters

Actor network	[512, 256, 128]
Critic network	[512, 256, 128]
Activation	Elu
Discount factor	0.99
GAE coefficient	0.95
PPO clipping	0.2
Entropy coefficient	1e-3
Learning rate	3e-4
Learning rate schedule	Adaptive
KL threshold for adaptive schedule	8e-3
Maximum gradient norm	1.0
Horizon length	24
Minibatch size	16384
Mini epochs	5
Critic coefficient	2

TABLE VI: Constraints hyperparameters

Torque τ^{lim}	3 Nm
Joint velocity $\dot{q}^{\text{lim}}$	16 rad/s
Joint acceleration $\ddot{q}^{\text{lim}}$	800 rad/s^2
Action rate $\dot{q}^{\text{des, lim}}$	80 rad/s
Base orientation $base^{\text{lim}}$	0.1 rad
Contact force f^{lim}	50 N
Hip angle hip^{lim}	0.2 rad
Air time $t^{\text{target}}_{\text{air time}}$	0.25s
Number of foot contacts $n^{\text{target}}_{\text{foot contact}}$	2
Velocity tracking ϵ_{track}	0.2 m/s or rad/s