Gain Tuning Is Not What You Need: Reward Gain Adaptation for Constrained Locomotion Learning

Arthicha Srisuchinnawong[1] and Poramate Manoonpong[1,2]
[1]Vidyasirimedhi Institute of Science and Technology, Rayong, Thailand.
[2]The University of Southern Denmark, Odense, Denmark.

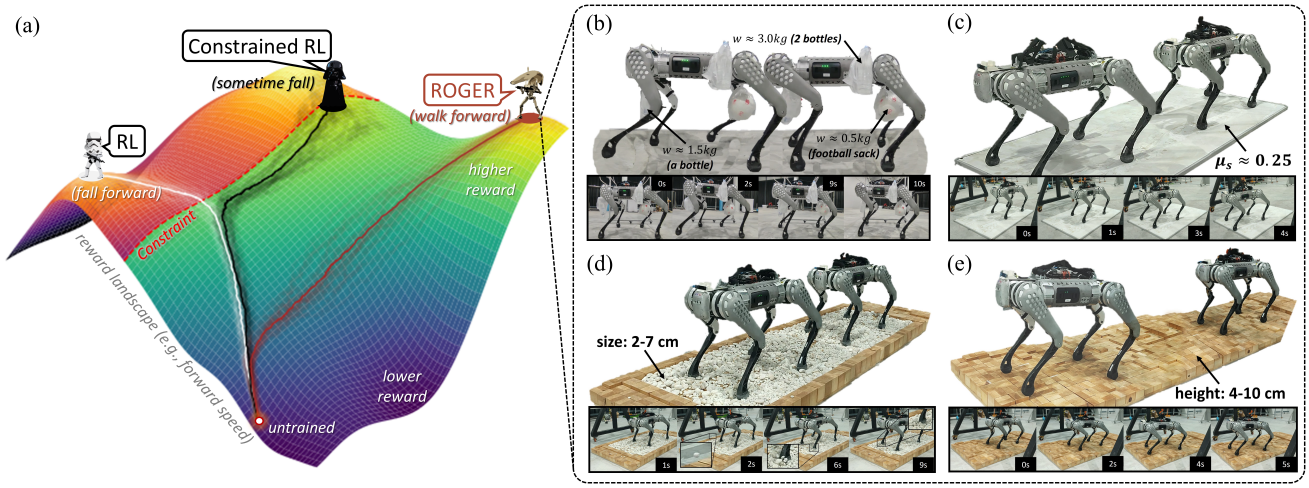


Fig. 1: (a) Parameter trajectories from (white) RL, (black) constrained RL, and (brown) ROGER on a simulated reward landscape with their (transparent) explorations. Brighter regions indicate higher rewards, while darker regions indicate lower rewards. The red areas highlight violations with a red dashed line indicating the constraint threshold. RL and constrained RL consistently violate constraints, possibly during exploration, while ROGER effectively avoids violations. Physical quadruped locomotion learning from scratch (b) with dynamic load, (c) on slippery terrain, (d) on a loose gravel field, and (e) on a random step field. A video of this experiment is available at https://youtu.be/Cqu7vLT_Piw?si=jtzJCpRubbFHx06w.

***Abstract*—Existing robot locomotion learning techniques rely heavily on the offline selection of proper reward weighting gains and cannot guarantee constraint satisfaction (i.e., constraint violation) during training. Thus, this work aims to address both issues by proposing Reward-Oriented Gains via Embodied Regulation (ROGER), which adapts reward-weighting gains online based on penalties received throughout the embodied interaction process. The ratio between the positive reward (primary reward) and negative reward (penalty) gains is automatically reduced as the learning approaches the constraint thresholds to avoid violation. Conversely, the ratio is increased when learning is in safe states to prioritize performance. With a 60-kg quadruped robot, ROGER achieved near-zero constraint violation throughout multiple learning trials. It also achieved up to 50% more primary reward than the equivalent state-of-the-art techniques. In MuJoCo continuous locomotion benchmarks, including a single-leg hopper, ROGER exhibited comparable or up to 100% higher performance and 60% less torque usage and orientation deviation compared to those trained with the default reward function. Finally, real-world locomotion learning of a physical quadruped robot was achieved from scratch within one hour without any falls. Therefore, this work contributes to constraint-satisfying real-world continual robot locomotion learning and simplifies reward weighting gain tuning, potentially facilitating the development of physical robots and those that learn in the real world.**

I. Introduction

Robot locomotion is a challenging task involving the embodied interaction between the robot and the environment [1, 2]. To properly exploit this embodied interaction, reinforcement learning (RL) has been employed as a promising framework, enabling robots to discover effective control policies on their own [3]. As a result, this technique has demonstrated remarkable success [4, 5, 6]. Nevertheless, traditional RL approaches are influenced by random exploration and are prone to constraint violation. Desired characteristics, such as being stable without falling and staying within physical limits, cannot be guaranteed either during [7] or after [6, 8] the learning process (white path in Figure 1), leading to instability, danger, and degraded performance [9].

To deal with this challenge, constrained RL emerges as a technique to enforce the constraints through mathematical formulations describing the desired characteristics of robot behaviors [9]. This technique can be divided into model-based and model-free approaches. The model-based approach relies on accurate models for learning, making model-free

more practical. Without an environment model, model-free constrained RL represents undesired behaviors using penalty terms (R_{it}), resulting in the following reward function (R_t):

$$R_t = \lambda_{0t} R_{0t} - \sum \lambda_{it} R_{it}, \tag{1}$$

where R_t, R_{0t}, and R_{it} denote the total reward, primary reward (i.e., the weighted summation of all positive reward terms), and i^{th} semi-positive definite constraint penalty terms at time t, while λ_{0t} and λ_{it} denote the corresponding weighting gains. This approach can be categorized further into two groups: fixed-weighting, where the gains are pre-tuned and fixed; and adaptive weighting, where the gains are adapted or learned.

A. Fixed-Weighting Constrained RL

In fixed-weighting constrained RL, λ_{0t} and λ_{it} are carefully tuned offline, either empirically or by hyperparameter searching, and kept fixed throughout the learning. Most works employ fixed weighting gains with error-based penalties, e.g., a squared error relative to the desired states or a highly negative penalty given at undesired states to enforce constraint satisfaction and the desired results. Others model the penalty terms as control barrier functions (CBFs) [6, 10], providing high penalties near safety boundaries only to obtain improved performance. These techniques could include as many as 16 terms with 11 distinct properly selected values [4], where adjusting a single hyperparameter may require four additional training repetitions [11]. Choosing improper values may result in unnatural motion, undesired gaits, or poor performance [6], making the selection process crucial and time-consuming.

B. Adaptive Weighting Constrained RL

To tackle the issue of fixed-weighting constrained RL, in adaptive weighting constrained RL, λ_{0t} and λ_{it} are continuously adjusted throughout the learning. Adaptive weighting constrained RL can be further divided into two categories: (1) primal-dual approaches [12, 13, 14, 15], which optimize the network parameters in the primal update while numerically adjusting the penalty regularization gains in the dual update, and (2) primal-only approaches [16, 17], which explicitly compute the penalty regularization gains rather than using iterative optimization.

One early technique, primal-dual optimization (PDO) [12], maximizes the total reward in the primal update while simultaneously adjusting the penalty gains in the dual update, as summarized in Appendix A. This method provides a foundation for later works like constrained policy optimization (CPO) [13], which resets the penalty gains and incorporates trust regions for each dual update to achieve higher performance with greater computational complexity. Although a recent quadruped robot trained with CPO managed to progressively reduce its orientation deviation during learning, its exploration trials still experienced constraint violations [14]. Building upon CPO, inertia-point policy optimization (IPO) [15] replaces penalty terms with logarithmic CBF; however, a quadruped robot trained with IPO experienced around 1,300 constraint violations per 1,000 episodes, according to [8].

One possible reason for constraint violation during learning is the delay in adjusting the regularization gains in primal-dual approaches [16]. In other words, multiple updates are required to increase a penalty gain from nearly zero to a high proper value to ensure constraint satisfaction. To investigate this, constraint-rectified policy optimization (CRPO) [16], a primal-only approach, introduces a reward-switching mechanism, which performs policy updates solely with the constraint penalties after a certain tolerance is exceeded, as summarized in Appendix A. Although [8] reported that CRPO outperformed previous techniques, like CPO and IPO, a quadruped robot trained with CRPO still experienced around 960 violations per 1,000 episodes; besides, its effectiveness relied on the switching threshold, i.e., tolerance parameter. Following that, according to [17], a variant of CRPO incorporates a distortion-based risk measure to modify reward distribution, highlighting high-penalty actions; however, a quadruped robot trained with this technique still experienced around 300 constraint violations per 1,000 episodes. Despite simplifying much of the tuning process, constraint violations persist to some extent (black path in Figure 1).

Recent techniques proposed in early 2025 are based on these approaches. For instance, QRSAC-Lagrangian [18] performs dual updates as in PDO with Adam optimizer to handle shift in value functions and improve stability, Constraint-Rectified Multi-Objective Policy Optimization (CR-MOPO) [19] modifies CRPO for multiple constraint objectives along with incorporating a conflict-averse technique optimized with natural policy to deal with conflicting objectives, while [20] uses neural network-based CBFs trained to adjust CBFs for specific conditions.

Recent techniques proposed in early 2025 build on these approaches. For instance, QRSAC-Lagrangian [18] performs a dual update similar to PDO, using the Adam optimizer to handle shifts in value functions and improve stability. Constraint-Rectified Multi-Objective Policy Optimization (CR-MOPO) [19] extends CRPO to multiple constraint objectives and incorporates a conflict-averse technique optimized with a natural policy gradient to handle conflicting objectives. Meanwhile, [20] employs neural network-based CBFs that are trained to adapt to specific conditions.

C. Current Stage of Real-World Locomotion Learning

Although several constrained RL techniques have been proposed, the guarantee of near-zero constraint violations, especially during learning, has yet to be achieved. Only two robots have demonstrated locomotion learning in the real world by using simple reward functions with some workarounds [7, 21]. In one work, a physical hexapod robot achieved locomotion learning within 20 minutes using a single-term speed reward function without falling, thanks to the stability provided by leg redundancy [21]. In another work, a physical quadruped robot achieved this under a

similar time scale using a reward function combining speed and orientation plus a fall recovery policy for resetting the robot [7]. Therefore, when foot contact redundancy cannot be obtained or the consequence of falling is destructive, safety constraints defined using certain robot state variables, such as orientation, must be predefined, included in the reward function, and satisfied at all times.

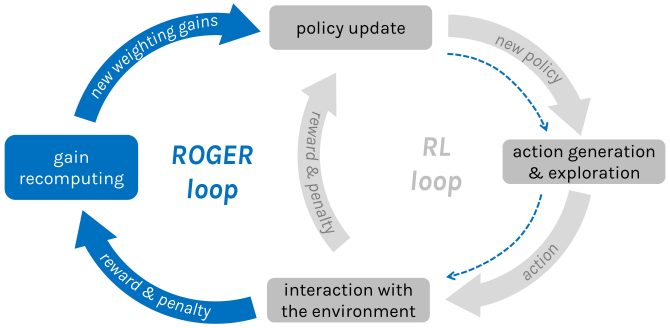


Fig. 2: Illustration of how embodied interaction between the robot and the environment can be used to train a control policy. The traditional RL loop is shown in gray, and the additional ROGER loop is shown in blue.

To this end, this work hypothesizes that robot-environment interaction can be leveraged for both policy updates (RL loop) and dynamic reward gain adjustments (online weighting gain adaptation loop), as illustrated in Figure 2, without introducing additional hyperparameters that require tuning. Building on this idea, this work proposes Reward-Oriented Gains via Embodied Regulation (ROGER), dynamically balancing reward weightings based on proximity to the constraint thresholds. These weightings are continuously refined through embodied interaction, where insufficient penalty gains drive the robot toward unsafe states, prompting stronger penalties and reducing the primary gain to prevent violations. In summary, ROGER has the following key advantages:

1) Ensuring constraint satisfaction throughout learning (brown path in Figure 1).
2) Elimination of extensive gain tuning by utilizing intuitive hyperparameters, e.g., constraint thresholds.

This paper is organized as follows: Section II describes ROGER in more detail; Section III presents the experimental results, including real-world RL of a quadruped robot; Section IV discusses the results and future potential; and finally, Section V discusses the limitations.

II. Reward-Oriented Gains via Embodied Regulation (ROGER)

The core idea of Reward-Oriented Gains via Embodied Regulation (ROGER) is to ensure safe operation while maximizing performance by dynamically adjusting the reward structure through real-time interaction with the environment, thereby eliminating the need for additional non-intuitive hyperparameters. When the robot is far from the constraint thresholds, ROGER uses the high primary reward weighting gain with low penalty weighting gains to encourage task-specific optimization. As the robot approaches the constraint thresholds, ROGER automatically increases the penalty weighting gains and reduces the primary reward weighting gain, discouraging unsafe actions and reducing the penalties. This process continues iteratively: when penalties reach a critical point, they steer the robot back into safe regions, after which the penalty weighting gains decrease, allowing for improved performance.

To achieve this, ROGER adopts a multiple-reward-channel setup [22], where rewards and penalties are stored independently before being combined into total advantage scores for policy updates. This setup enables ROGER to dynamically adjust the reward and penalty weighting gains (Equation 1) across different learning episodes and timesteps. At each timestep, the weighting gains are computed according to:

$$\lambda_{0t} = 1 - \Delta_t, \tag{2}$$

$$\lambda_{it} = r_{\lambda_i t}\Delta_t, \tag{3}$$

where Δ_t denotes the clipped summation of constraint penalties, preventing $\lambda_{0t} < 0$ and ensuring that $\lambda_{0t} + \Sigma\lambda_{it} = 1$, and $r_{\lambda_i t}$ denotes the constraint contribution ratio of the i^{th} constraint penalty term. Δ_t and $r_{\lambda_i t}$ are computed from the ratio of the estimated constraint penalties ($\tilde{R}_{it}$) to the constraint thresholds (τ_i) as:

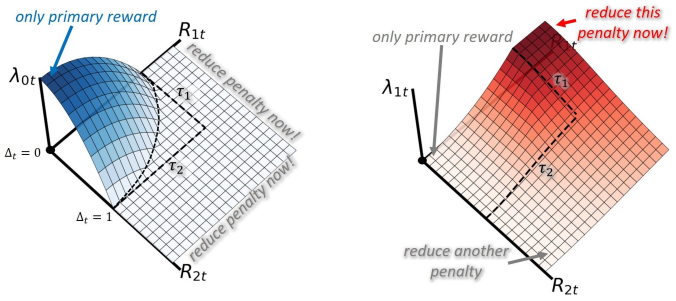
$$\Delta_t = \min\left\{\sum_j (\tilde{R}_{jt}/\tau_j)^2, 1.0\right\}, \tag{4}$$

$$r_{\lambda_i t} = \frac{(\tilde{R}_{it}/\tau_i)^2}{\sum_j (\tilde{R}_{jt}/\tau_j)^2}, \tag{5}$$

where τ_i and τ_j are selected intuitively based on the physical properties of the system, e.g., maximum hardware limits, together forming a safe region into which the robot should be bounded, as shown by the dashed lines in Figure 3, while $\tilde{R}_{it}$ is a statistical estimate of the i^{th} penalty at time t: $\tilde{R}_{it} = \text{average}[R_{it}] - k_\sigma \text{std}[\sigma_{it}]$. A preliminary result when varying k_σ is presented in Figure S8b in the Appendix.

This adaptation strategy is also proven to be partially stable in key conditions (i.e., near the constraint thresholds and convergence), while the expected primary reward is guaranteed to increase, as detailed in Appendix B [23]. As a result, if an optimal solution exists far from the constraint thresholds, such as body orientation in robot locomotion, ROGER will converge to that solution; otherwise, it chooses a safe alternative.

Intuitively, the robot initially maximizes the primary reward since all penalty terms and their gains are zero, i.e., $\Delta_t = 0$ and $\lambda_{0t} = 1$, as illustrated in Figure 3a. As the robot approaches a constraint threshold, the R_{it} terms increase, causing Δ_t to grow, reducing λ_{0t} and increasing λ_{it}, thereby distributing more and more attention to the penalty terms. This setup prevents the robot from approaching the constraint thresholds. However, when near a constraint threshold, i.e., some $R_{it} \approx \tau_i$, Δ_t saturates at 1.0, making $\lambda_{0t} \approx 0$



(a) (b)

Fig. 3: This adaptation strategy is also proven to be partially stable in key conditions (i.e., near the constraint thresholds and convergence), while the expected primary reward is guaranteed to increase, as detailed in Appendix B. As a result, if an optimal solution exists far from the constraint thresholds, such as body orientation in robot locomotion, ROGER will converge to that solution; otherwise, it chooses a safe alternative.

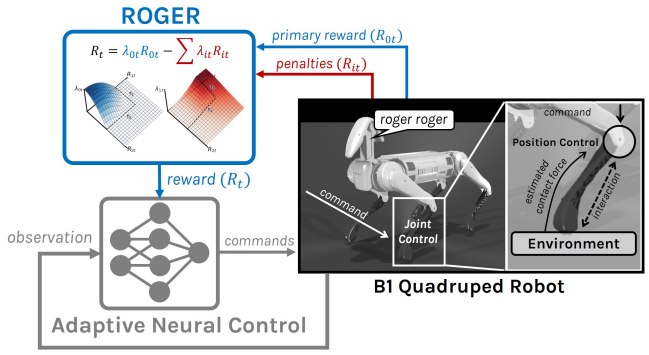


Fig. 4: Locomotion learning framework for the Unitree B1 quadruped with ROGER. An adaptive neural control produces joint position targets that are used by low-level controllers. After execution, the robot receives rewards and penalties, which are then combined using ROGER and subsequently used to train the neural control.

and $R_t \approx -R_{it}$. This setup neglects the primary reward, immediately enforcing the reduction of the penalty terms and ensuring constraint satisfaction.

Simultaneously, the penalty weighting gains are distributed according to the ratio of R_{it}, as illustrated in Figure 3b. As the first penalty term (R_{1t}) increases, it receives more weight and higher optimization priority. However, as other terms, e.g., R_{2t}, rise, the weighting gains are redistributed, lowering the priority of the first penalty term to increase those of the others. Thus, this setup automatically balances the contributions between different penalty terms online during locomotion learning. In total, ROGER[1] takes an additional computation time of 0.46±0.09 ms on an Intel i7 CPU with an Nvidia GTX1050 GPU, or approximately 0.03% of data collection/exploration time.

III. Experiments and Results

Two experiments were conducted to investigate and evaluate ROGER. The first compared ROGER with state-of-the-art techniques on a 60-kg quadruped robot in simulation, followed by the demonstration of real-world locomotion learning. The second further tested ROGER with a MuJoCo hopper, which violates the assumptions presented in Appendix B: zero-penalty optimality and gentle system and transition dynamics. Under this condition, the reward weightings adapted using ROGER were compared with the default reward weightings provided by OpenAI Gymnasium and CRPO. The results of other locomotion learning tasks are provided in Appendix D.

A. Quadruped Locomotion Learning

Unlike previous studies that used small, less sensitive quadruped robots, the first experiment performs a comparison on a 60-kg Unitree B1 (Figure 4), a platform more prone to instability and falling if improper policies are learned. To ensure safe and efficient locomotion learning, a state-of-the-art adaptive neural control framework, called Sequential Motion Executor-Adaptive Gradient-weighting Online Learning (SME-AGOL) [21], was adopted, as shown in Figure 4. This control framework has previously demonstrated real-world locomotion learning on a physical hexapod robot. Moreover, it supports stable and smooth locomotion by performing exploration in parameter space, e.g., avoiding noisy exploration as in PPO, and providing interpretability, e.g., mapping weights directly to a series of corresponding robot trajectories/configurations.

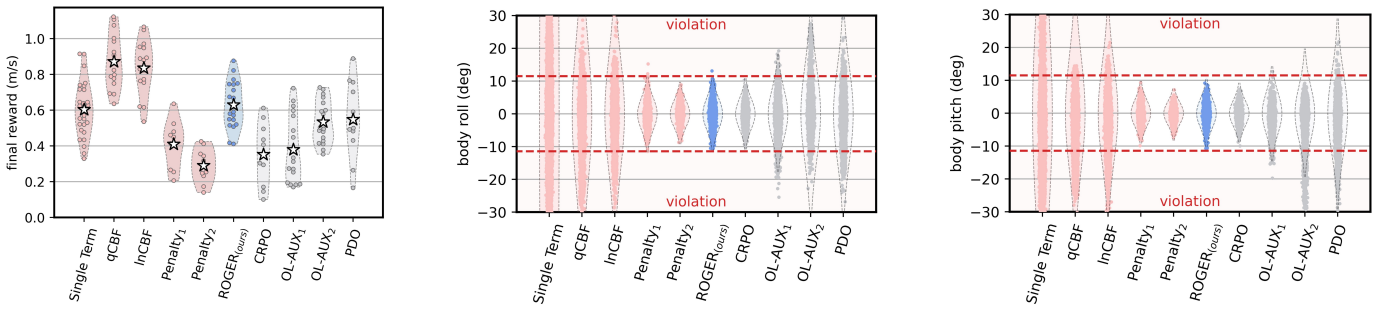
The SME network is a central pattern generator-based network. It consists of three layers: (1) a central pattern generator layer producing rhythmic activity, (2) a triangular basis layer forming locomotion bases, and (3) an output layer mapping the bases to eight joint position commands for the quadruped robot. It should be noted that the hip abduction/adduction joints were fixed, reserving them for future turning control. The mapping weights were initialized to zero before being optimized using the AGOL learning rule as follows:

$$\Delta\theta = \eta_\theta \sum_{i=0}^{n} |\nabla_\theta \tilde{a}_i| \left(\frac{\tilde{\theta}_i - \theta}{\sigma_\theta^2} \right) \left(\frac{G_i - \bar{G}_i}{\sigma_{G_i}} \right), \quad (6)$$

$$\Delta\sigma_\theta = \eta_\sigma \sum_{i=0}^{n} |\nabla_\theta \tilde{a}_i| \left(\frac{(\tilde{\theta}_i - \theta)^2 - \sigma_\theta^2}{\sigma_\theta^3} \right) \left(\frac{G_i - \bar{G}_i}{\sigma_{G_i}} \right), \quad (7)$$

where $\Delta\theta$ and $\Delta\sigma_\theta$ denote the updates to the parameters (θ) and the exploration standard deviations (σ_θ), respectively. η_θ and η_σ denote the learning rates. n denotes the length of the stored trajectories, i.e., 8 episodes × 70 timesteps (≈ 3 gait cycles per episode). $\nabla_\theta \tilde{a}_i$ denotes the gradient of the explored actions. $\tilde{\theta}$ denotes the explored parameter re-randomized at the beginning of every episode. $\bar{G}_i$ and σ_{G_i} denote the average and standard deviation of the return G_i, respectively, serving as the baseline and normalization gains for advantage estimation. Taking advantage of the rhythmic nature of locomotion, the

[1]Code is available at https://github.com/Arthicha/ROGER_ROGER_public.



(a) (b) (c)

Fig. 5: (a) Final primary reward values obtained from the last training episode and (b-c) roll and pitch angles recorded throughout the locomotion learning of the simulated Unitree B1 quadruped robot. The robot was trained using six techniques: two fixed-weighting techniques in red (fixed-gain penalty and fixed-gain CBF), three adaptive weighting techniques in gray (PDO, CRPO, and OL-AUX), and ROGER in blue. All conditions are presented along with their kernel density estimation. In (b-c), red dashed lines indicate constraint thresholds at $\pm$ 0.2 rads, or approximately $\pm$ 10°; therefore, the data points exceeding these lines indicate violations. A video of this experiment is available at https://youtu.be/cZ5qOw0i_T4.

return G_i is computed from the average of the total reward performed over a horizon of 20 timesteps, or approximately a gait cycle. Here, ROGER was applied as an add-on mechanism to the AGOL learning mechanism by adapting its reward weighting gains.

The total reward function in Equation 1 includes a forward velocity reward ($R_{0t} = v_{\text{fwd}}$, encouraging forward movement) and orientation penalties for roll ($R_{1t} = |\alpha_t|$, penalizing sideways oscillation) and pitch ($R_{2t} = |\beta_t|$, penalizing longitudinal oscillation). The constraint thresholds, e.g., τ_j in Equations 4 and 5, are selected as 0.2 rad, or approximately 10°. To include 99.9% of the exploration uncertainty in the process, ROGER recomputes the reward weighting gains using $k_\sigma = 3$. In other state-of-the-art methods, $\tilde{R}_{it}$ is used as summarized in Appendix A.

1) Simulated Robot Experiment: The comparison between ROGER and state-of-the-art constrained RL techniques was conducted using a Mujoco-based simulation of a 60-kg Unitree B1 quadruped robot. The state-of-the-art techniques include two fixed-weighting techniques: fixed-gain penalty (Penalty) and fixed-gain CBF (qCBF: quadratic CBF and lnCBF: logarithmic CBF); three adaptive-weighting techniques: PDO, CRPO, and OL-AUX; and ROGER. Note that OL-AUX [24] is included here as an additional baseline because it dynamically adapts the auxiliary weighting gains (e.g., penalty), with the aim of exploiting the auxiliary terms to maximize the primary reward. The summarized details of all techniques are provided in Appendix A.

The key metrics for the evaluation are the final primary reward (i.e., robot forward speed), the constraint violation rate throughout the learning process obtained from kernel density estimation, and the maximum roll and pitch deviations. For each technique, hyperparameters were selected through a grid search [11], with those yielding high rewards and low constraint violations being selected for comparison. Some of the preliminary selection results are presented in Appendix C. The robot was subjected to 500 training episodes under each condition, with over 10 repetitions per condition.

Figures 5b and 5c present that, in this case, the fixed gain penalty is the only fixed-weighting approach capable of satisfying the constraints throughout learning. When properly selected, the fixed gain penalty resulted in below 10° of orientation deviation with an estimated violation probability of 2×10^{-12}, or three violations in 25,000 timesteps. In contrast, training the quadruped robot solely with the primary reward did not satisfy any constraints: the robot experienced over 30° orientation deviations with an estimated violation probability of 0.85, both of which were significantly higher than those of the fixed-gain penalty *(p-value $\ll$ 0.01, two-proportion test)*. Similarly, CBF-based approaches, whether logarithmic or quadratic, reduced the estimated violation probability to 0.60–0.16, with most of the deviation being below 20°; yet, the numbers were still significantly higher than those of the fixed gain penalty *(p-value $\ll$ 0.01, two-proportion test)*.

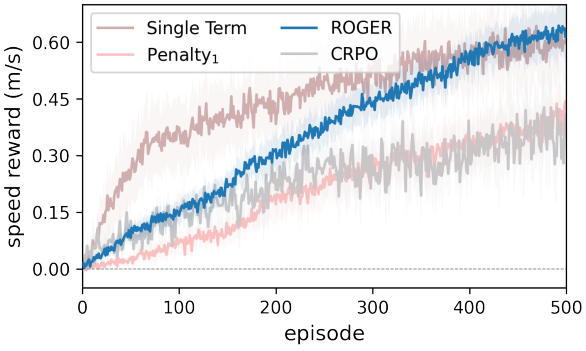


Fig. 6: Evolution of the main reward term across 500 learning episodes from the locomotion learning of the simulated Unitree B1 quadruped robot trained with (dark red) only the primary reward term, (light red) fixed gain penalty, (blue) ROGER, and (gray) CRPO.

Figures 5b and 5c also reveal that CRPO and ROGER,

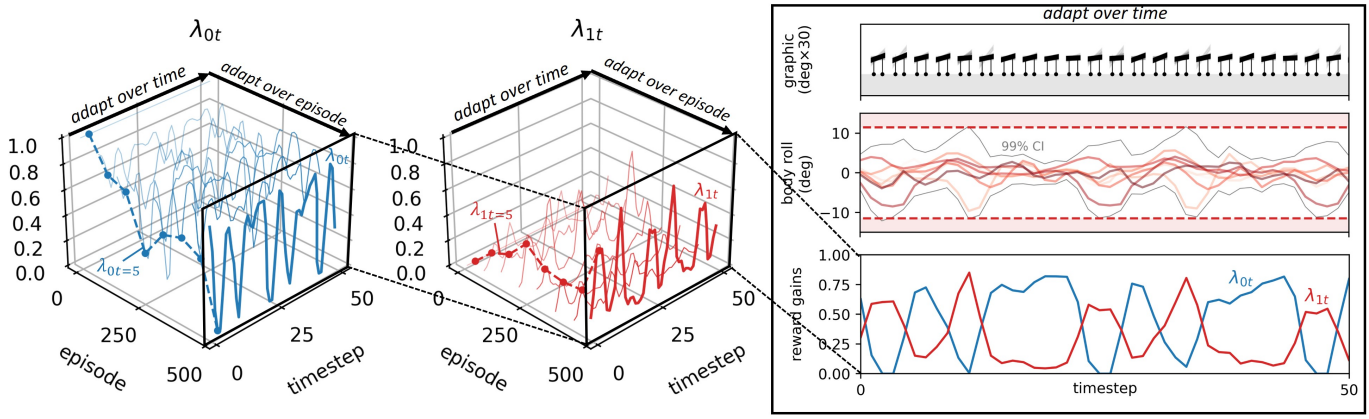


Fig. 7: (Left) Evolution of the primary weighting gain (λ_{0t}) and body roll penalty weighting gain (λ_{1t}) across 50 timesteps and 500 episodes, obtained from the locomotion learning of a simulated quadruped robot. (Top right) Graphical illustration showing the robot body from the back view, with the values amplified 30 times for visualization purposes. (Middle right) Body roll values collected over eight previous episodes (in red) along with their 99% confidence interval (in gray). (Bottom right) Evolution of the weighting gains: λ_{0t} in blue and λ_{1t} in red.

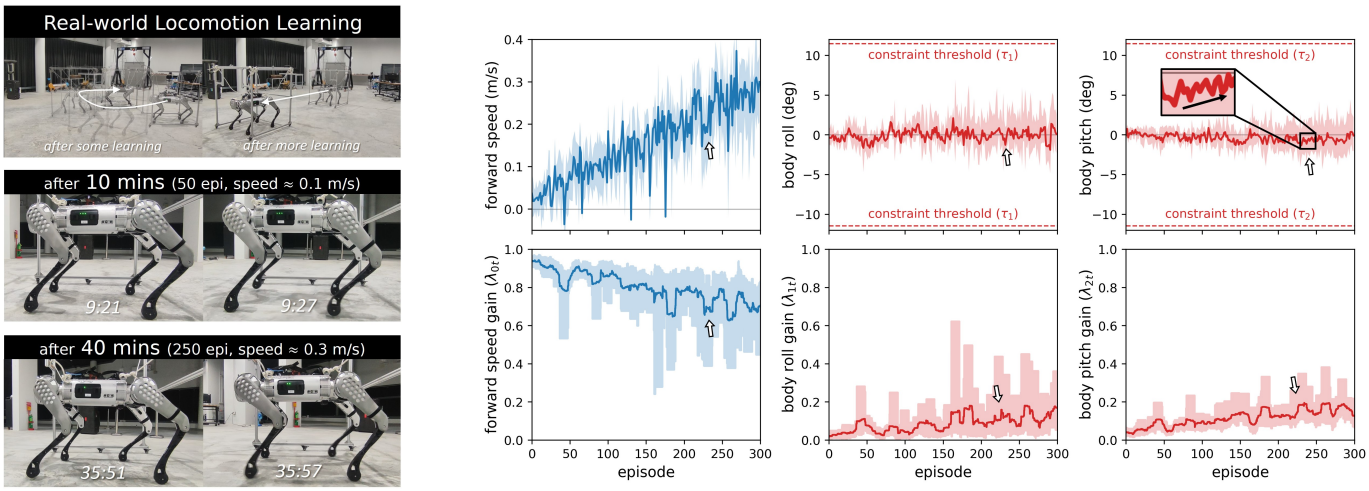
which recompute weighting gains for each iteration, are the only two adaptive weighting approaches capable of satisfying constraints throughout learning. CRPO had zero violations in 25,000 timesteps, while ROGER merely exhibited one in 50,000 timesteps (without any falls; the violation was caused by exploration, as no violation was observed during reruns without exploration). This corresponds to an estimated violation probability of around $2 \times 10^{-12}\%$. In comparison, PDO exhibited $\approx 20^\circ$ in orientation deviations and an estimated violation probability of 0.02, significantly higher than those from CRPO and ROGER *(p-value $\ll$ 0.01, two-proportion test)*. Interestingly, OL-AUX, designed to maximize the increase in the primary reward, exhibited similar 20° deviations with a lower estimated violation probability of 0.0006–1.6% compared to PDO *(p-value $\ll$ 0.01, two-proportion test)*. Nevertheless, the deviation and estimated violation probability of OL-AUX were still significantly greater than those of CRPO and ROGER *(p-value $\ll$ 0.01, two-proportion test)*.

Among the three techniques with fewer than three violations in over 25,000 training timesteps presented in Figure 5, ROGER achieved the highest final primary reward. The primary rewards reached 0.1 m/s and 0.3 m/s after 50 and 250 learning episodes, respectively, before the final value of 0.6 m/s at 500 episodes, as shown in Figure 6. This final value of 0.6 m/s was 50% greater than those of the fixed gain penalty and CRPO, which were $\approx$ 0.3 m/s *(p-value $\ll$ 0.01, t-test)*. Interestingly, the final primary reward from ROGER matched that achieved when using the primary reward solely *(p-value = 0.54, t-test)*, but with near-zero constraint violation. This highlights that ROGER can autonomously and properly adapt and balance different reward weighting gains while ensuring constraint satisfaction.

Finally, Figure 7 presents the underlying mechanism of ROGER in more detail, revealing that the primary weighting gain λ_{0t} and roll penalty weighting gain λ_{1t} are dynamically adjusted based on the estimated body roll (gray lines), both across and within episodes. When the robot tilts to one side, λ_{0t} decreases and λ_{1t} increases to prevent tilting at that specific stage. Conversely, when the robot is stable and its body roll exhibits minor deviations, λ_{0t} increases and λ_{1t} decreases to prioritize the primary reward at that stage. Given the rhythmic nature of robot locomotion in this case, where the robot walks for two gait cycles in 50 timesteps, the weighting gains exhibit two periodic repetitions, offering insights into the learning objectives across different stages.

2) Physical Robot Experiment: According to the results of the simulation experiment, ROGER was then tested on a 60-kg physical quadruped robot. The robot was trained for 300 learning episodes with all mapping weights initialized as zero and the reward and penalty estimated from the outputs of an Intel RealSense T265 camera installed at the front. Training started from a standing posture—i.e., the home configuration—which has a high center of mass and is prone to falling. A wheeled 15-kg support structure with slack ropes was employed as a safety mechanism. The support was set up such that it did not assist in maintaining the stability of the robot during learning. However, it added complexity to the task, as the robot had to drag the structure while learning from an imperfect pose estimation. Due to the limited testing space, the robot waited for three seconds after each episode to allow the experimenter to pause the main program and reorient the robot after it went near the testing space boundaries. In total, the entire trial, including locomotion learning and reset, lasted almost one hour.

Figure 8 presents the results of physical locomotion learning. The robot successfully completed all five trials without falling, demonstrating the capability of ROGER to train a heavy quadruped robot in real-world conditions. By episode 50 ($\approx$ 10 minutes), the robot had achieved a forward



(a) (b)

Fig. 8: (a) Snapshots capturing the locomotion learning of a physical robot after 50 and 250 episodes. (b, top) Forward speed (i.e., primary reward), robot roll, robot pitch, (b, bottom) the reward weighting gains, and their min-max ranges, recorded throughout 300 learning episodes. The gray solid line indicates zero forward speed, while the red dashed lines indicate the constraint thresholds. A video of this experiment is available at https://youtu.be/F1olq7W6J9g.

speed of 0.1 m/s, with roll and pitch angles remaining below 3°. By episode 250 (≈ 40 minutes), the average forward speed had increased to 0.3 m/s, matching the performance observed in simulation experiments and potentially suggesting a minimal sim-to-real learning performance gap.

Notably, even under continuous random exploration, the robot effectively maintained its roll and pitch angles below the constraint threshold of 10° throughout 300 episodes, as shown in the top row of Figure 8b. This stability was achieved through the dynamic adjustment of reward weighting gains. In general, the orientation deviation increased slightly as the robot learned, reducing the primary weighting gain and increasing the roll and pitch penalty weighting gains, as shown in the bottom row of Figure 8b. As shown in the video (https://youtu.be/F1olq7W6J9g), the robot learned relatively short steps with low foot lift while walking on flat terrain, a strategy that helps maintain stability under continuous exploration. Longer steps and higher foot lifts, while potentially more efficient, are more likely to violate stability constraints during such stochastic movements, as shown in https://youtu.be/cZ5qOw0i_T4. Nevertheless, Figure 1 illustrates that a variety of complex gaits emerged under more challenging conditions, supporting that physical interaction with the environment shapes the resulting behavior [2].

Apart from the long-term trend, specific adjustments in the weighting gains were also observed throughout the learning in Figure 8b. For example, an interesting event occurred around episode 220 (≈ 2:36 minutes in the video; https://youtu.be/F1olq7W6J9g), when the robot started oscillating in both roll and pitch angles. In response, ROGER lowered the primary weighting gain and increased the penalty weighting gains, as shown in the bottom row of Figure 8b. The robot then spent the next five episodes adjusting its locomotion pattern to reduce the oscillation. This is reflected in a slight decrease in forward speed, accompanied by roll and pitch values changing toward zero shortly after 220 episodes. After 3:00 minutes, the robot had successfully reached a stable gait, demonstrating the autonomous balancing process of the primary reward and constraint penalty terms in real time.

To further investigate this mechanism, the robot was trained under four challenging conditions, as shown in Figure 1(b–e): (1) a step field with step heights varying between 4 and 10 cm; (2) a gravel field with gravel diameters ranging from 2 to 7 cm; (3) a slippery surface, created by applying machine lubricant to a whiteboard (static friction coefficient ≈ 0.25); and (4) dynamic loading, involving a 1.5-kg bottle of water placed at the front, two 1.5-kg bottles at the rear, and a football sack—together comprising nearly 10% of the robot's weight.

Under dynamic loading, the robot successfully maintained balance while walking forward. On the slippery surface, initial difficulty in maintaining ground contact was observed, but the robot eventually learned to slide forward while preserving body posture. On the loose gravel terrain, the robot leaned slightly forward and learned to kick the top-layer gravel to clear a path. In the step field, the terrain's rigidity restricted both foot placement and terrain manipulation, initially leading to unsuccessful attempts. To deal with this, training was initialized with a trotting-in-place gait featuring increased foot lifting. As a result, the robot developed a hopping-like gait to overcome the steps. Notably, all these behaviors emerged without the use of any exteroceptive terrain sensing, suggesting that feasible control policies should be reachable within the exploration while learning should satisfy constraints throughout.

B. Hopper Locomotion Learning

This experiment extends the evaluation of ROGER to further investigate a condition of assumption violations. A simulated MuJoCo hopper was used, which requires a highly dynamic hopping gait and is more sensitive than a quadruped robot. Due to being feedback-dependent, which cannot currently be achieved with the SME architecture [21], the hopper was instead controlled by a three-hidden-layer neural network with 256 hyperbolic tangent neurons per layer, trained using a standard PPO algorithm [25], as shown in Figure 4. The experiment followed the default OpenAI Gymnasium setup, except that the reward was decomposed into multiple channels as required by ROGER. The reward/penalty terms included forward velocity plus a healthy reward as the primary reward, an absolute torque penalty to penalize torque usage, and an absolute body orientation penalty to encourage an upright body. The torque penalty represents a non-zero constraint in the optimal solution (i.e., zero-penalty optimality violated), while the orientation penalty represents a sensitive constraint (i.e., gentle system and transition dynamics violated). The penalty thresholds were set at $\tau_1 = 1.0$ (maximum value) for torque and $\tau_2 \approx 10^\circ$ for orientation. Other hyperparameters, e.g., those of the neural network and learning algorithm, were empirically selected such that the robot could complete the task with the default reward function.

Three testing conditions were evaluated: (1) the baseline reward functions with fixed weighting gains obtained directly from the OpenAI Gymnasium, i.e., default; (2) CRPO; and (3) ROGER. Each condition was repeated for over 20 trials, with each trial lasting one million timesteps and 2048 timesteps per episode. Four performance metrics were employed: hopping distance (i.e., related to the main reward), absolute joint torque, orientation deviation, and the percentages of constraint violations.

Figure 9 demonstrates that overall, ROGER significantly outperformed the baselines, achieving greater hopping distances, reduced torque usage, and smaller orientation deviations. Specifically, ROGER achieved a hopping distance of 6.6 m, representing a 20% improvement over the default reward function *(p-value* < 0.01*, t-test)*, which was similar to that of CRPO. With ROGER, the average torque usage was 0.33 Nm, representing a 57% reduction from the default reward function *(p-value* < 0.01*, t-test)* and a 58% reduction from CRPO *(p-value* < 0.01*, t-test)*. Similarly, the average orientation deviation was 2°, representing a 59% decrease compared to the default reward function *(p-value* < 0.01*, t-test)* and 53% decrease compared to CRPO. In total, the percentages of constraint violation obtained with ROGER were lower than those with the default reward function and CRPO both throughout the learning process and in the last 10 episodes, as summarized in Appendix D. However, unlike the quadruped experiment, while ROGER ensures that the expected value of constraints is satisfied, the absence of falls was not guaranteed due to assumption violations (i.e., having sensitive system dynamics). Nevertheless, it offers improved performance and a simpler tuning process.

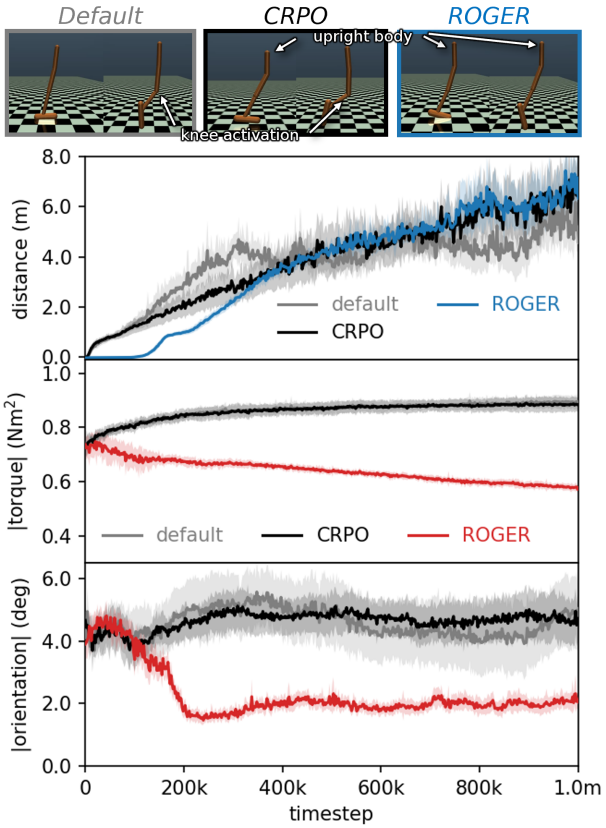


Fig. 9: Snapshots, hopping distance, joint torque usage, and absolute orientation deviation, obtained from a MuJoCo hopper trained using the default reward function from OpenAI Gymnasium, CRPO [16], and ROGER. A video of the experiment is available at https://youtu.be/SpL4awVgDZM.

Figure 9 further reveals the relationship between the three reward/penalty channels and the underlying mechanism of ROGER. During the first 50,000 timesteps, the hopper focused on reducing torque usage and orientation penalties, as indicated by their steady decline. Once these terms had been sufficiently minimized, the hopper shifted its focus toward maximizing the primary reward, resulting in a slight delay in primary reward maximization. Subsequently, the hopper showed a continuous performance increase while maintaining low torque usage and minimal orientation deviation.

Figure S10 in Appendix D suggests that, since ROGER utilizes embodied interaction between itself and the environment to adapt reward weighting gains, the results may be environment-specific. The torque usage obtained from ROGER$_\tau$, i.e., with only the primary reward and torque penalty, mirrored that obtained from the three-channel ROGER, with the orientation deviation being similar to that obtained from the default reward function. Conversely, the orientation deviation obtained from ROGER$_\beta$, i.e., with only the primary reward and orientation penalty, matched that of the three-channel ROGER, with the torque usage being similar to that obtained from the default reward function.

Interestingly, ROGER also demonstrated its success under stricter constraints, with the constraint thresholds being halved to $(\tau_1, \tau_2) = (0.5, 5^\circ)$, as shown in Figure S11 in Appendix D. In this scenario, the hopper initially prioritized the reduction of constraint penalties before maximizing the primary reward. Although this stricter configuration delayed reward optimization, the hopper maintained forward hopping without moving backward, i.e., exhibiting a negative hopping distance. These results reveal that given sufficient learning, ROGER can achieve comparable performance to finely tuned reward functions with constraint satisfaction.

IV. Conclusion

Nowadays, robot locomotion learning is not yet commonplace in the real world due to certain requirements, such as avoiding falling while learning in the case of quadruped robots [7]. Current state-of-the-art RL techniques often experience constraint violations in quadruped robots, especially during learning [8, 14], while selecting proper reward weighting gains is time-consuming [11] and risky, as shown in Appendix C. Therefore, this work introduces Reward-Oriented Gains via Embodied Regulation (ROGER), a simple rule that adjusts reward weighting gains and balances different objectives through the dynamic interaction between the robot and the environment. ROGER increases the penalty gains and decreases the primary weighting gain as the robot approaches the constraint thresholds, continuously enforcing constraint satisfaction. Conversely, it increasingly relaxes the constraints to prioritize primary reward maximization as the robot moves away from the constraint thresholds. Unlike most state-of-the-art techniques, which heavily rely on simulation-based training [10, 12, 13, 16], ROGER strictly enforces constraints, which potentially slows down learning in extreme conditions (e.g., Figure S11 in Appendix D) but is crucial for real-world learning where violations are catastrophic. Moreover, ROGER allows intuitive hyperparameter selection: constraint thresholds (τ_i; acceptable values) and uncertainty levels (k_σ, e.g., $k_\sigma = 3$ for 99.9% confidence), making the setting of unrealistic constraints, e.g., 1° body deviation, unlikely in practice.

ROGER demonstrates its effectiveness in both theory and practice. Theoretically, constraint satisfaction is achieved through partial stability under key conditions: near the constraint thresholds and when converged, while ensuring an increase in the primary reward, as detailed in Appendix B. Although ROGER is designed around three key assumptions—zero-penalty optimality, gentle system dynamics, and gentle learning dynamics—it still performs well even when these assumptions are violated. First, despite the torque constraint violation, where optimal torque values are non-zero for locomotion, ROGER remains effective, as shown in Figure 9. Second, in tests with the MuJoCo hopper and walker2D, which are less stable than the quadruped robot, ROGER outperforms state-of-the-art methods under sensitive system dynamics, although fall prevention isn't guaranteed (see Appendix D). Finally, an experiment with excessive learning rates on a quadruped robot, which has gentle system dynamics compared to the hopper and walker2D, reveals that sensitive learning dynamics can lead to failures, as constraint satisfaction cannot be guaranteed (Figure S8a in Appendix C).

Experimentally, ROGER exhibited near-zero violation throughout the multiple-seed locomotion learning of a heavy quadruped robot (i.e., five times heavier than Unitree A1 [7]), which is comparable to carefully tuned fixed weighting gains and state-of-the-art CRPO [16], but with a higher performance. Furthermore, the technique can be applied to DNN-PPO [4], outperforming the default reward functions in MuJoCo continuous locomotion learning tasks both in terms of the primary reward (e.g., distance), constraints where optimal values are expected to be non-zero (e.g., torque usage), and constraints where optimal values are expected to be near zero (e.g., orientation stability). Finally, ROGER offers stable real-world locomotion learning on a quadruped robot, both on regular terrain and under various challenging conditions, in less than an hour, with the robot neither falling nor requiring extensive tuning, even with limited training data (i.e., ≈ 500 timesteps per update, or rather 0.5% of that in [26]) and without exteroceptive terrain sensing.

In summary, this work highlights how embodied interaction can dynamically adapt reward weighting gains in real time, achieving near-optimal performance across all tests presented here while minimizing the need for reward weighting gain tuning. By enabling continuous improvement and simplifying tuning, this work contributes to physical AI research and development in terms of efficient continual robot learning in the real world, especially when parameter selection is high-stakes or resource-intensive. Thus, if constraint violations are acceptable during learning, simulation-based methods may be preferable; for real-world continual learning or fine-tuning, ROGER may be a better choice.

V. Limitation

Currently, ROGER is limited to proprioceptive constraints within robot systems that satisfy zero-penalty optimality as well as gentle system and learning dynamics. It does not account for exteroceptive constraints like global positioning, which can accumulate over time. These limitations, including real-world quadruped locomotion learning on complex terrain with exteroceptive terrain sensing [27], will be addressed in future works.

Acknowledgments

This work is intended for research and academic purposes. The authors would like to thank Chaicharn Akkawutvanich for valuable better-late-than-never post-submission discussions and comments, Run Janna for assisting in fine-tuning the low-level control, and Kanut Tarapongnivat for suggesting the dynamic load experiment. We also appreciate the reviewers and area chair for their thoughtful feedback and constructive suggestions.

References

[1] Riccardo Manzotti. Embodied AI beyond embodied cognition and enactivism. *Philosophies*, 4(3):39, 2019.

[2] Rolf Pfeifer and Josh Bongard. *How the body shapes the way we think: a new view of intelligence*. MIT press, 2006.

[3] Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999.

[4] David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.

[5] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 43(4):572–587, 2024.

[6] Yunho Kim, Hyunsik Oh, Jeonghyun Lee, Jinhyeok Choi, Gwanghyeon Ji, Moonkyu Jung, Donghoon Youm, and Jemin Hwangbo. Not only rewards but also constraints: Applications on legged robot locomotion. *IEEE Transactions on Robotics*, 2024.

[7] Laura Smith, Ilya Kostrikov, and Sergey Levine. Demonstrating a walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *Robotics: Science and Systems (RSS) Demo*, 2(3):4, 2023.

[8] Joonho Lee, Lukas Schroth, Victor Klemm, Marko Bjelonic, Alexander Reske, and Marco Hutter. Exploring constrained reinforcement learning algorithms for quadrupedal locomotion. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2024)*, 2024.

[9] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, and Alois Knoll. A review of safe reinforcement learning: Methods, theories and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[10] Thanh Long Vu, Sayak Mukherjee, Renke Huang, and Qiuhua Huang. Barrier function-based safe reinforcement learning for emergency control of power systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 3652–3657. IEEE, 2021.

[11] Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them. In *International Conference on Machine Learning*, pages 9104–9149. PMLR, 2023.

[12] Santiago Paternain, Miguel Calvo-Fullana, Luiz FO Chamon, and Alejandro Ribeiro. Safe policies for reinforcement learning via primal-dual methods. *IEEE Transactions on Automatic Control*, 68(3):1321–1336, 2022.

[13] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

[14] Dohyeong Kim, Kyungjae Lee, and Songhwai Oh. Trust region-based safe distributional reinforcement learning for multiple constraints. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] Yongshuai Liu, Jiaxin Ding, and Xin Liu. Ipo: Interior-point policy optimization under constraints. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 34, pages 4940–4947, 2020.

[16] Tengyu Xu, Yingbin Liang, and Guanghui Lan. Crpo: A new approach for safe reinforcement learning with convergence guarantee. In *International Conference on Machine Learning*, pages 11480–11491. PMLR, 2021.

[17] James Queeney and Mouhacine Benosman. Risk-averse model uncertainty for distributionally robust safe reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[18] Yu Ishihara, Noriaki Takasugi, Kotaro Kawakami, Masaya Kinoshita, and Kazumi Aoyama. Constraints as rewards: Reinforcement learning for robots without reward functions. *arXiv preprint arXiv:2501.04228*, 2025.

[19] Shangding Gu, Bilgehan Sel, Yuhao Ding, Lu Wang, Qingwei Lin, Alois Knoll, and Ming Jin. Safe and balanced: A framework for constrained multi-objective reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[20] Weishu Zhan, Zheng Liang, Hongyu Song, and Wei Pan. Safe distributed learning-enhanced predictive control for multiple quadrupedal robots. *arXiv preprint arXiv:2503.05836*, 2025.

[21] Arthicha Srisuchinnawong and Poramate Manoonpong. An interpretable neural control network with adaptable online learning for sample efficient robot locomotion learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2025. doi: 10.1109/TNNLS.2025.3552793.

[22] Harm Van Seijen, Mehdi Fatemi, Joshua Romoff, Romain Laroche, Tavian Barnes, and Jeffrey Tsang. Hybrid reward architecture for reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[23] Aleksandr Mikhailovich Lyapunov. The general problem of the stability of motion. *International Journal of Control*, 55(3):531–534, 1992.

[24] Xingyu Lin, Harjatin Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[25] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[26] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[27] Arthicha Srisuchinnawong and Poramate Manoonpong.

Growable and interpretable neural control with online continual learning for autonomous lifelong locomotion learning machines. *The International Journal of Robotics Research*, 2025.

APPENDIX

A. Comparison Table

Table S1 presents the fundamental concept underlying different model-free constrained RL techniques, where $\tilde{x}_i$ denotes an observation/state variable, R_{it} denotes a penalty term, $\tilde{R}_{it}$ denotes the statistical estimate of the penalty term, τ_i denotes the constraint threshold, δ_i denotes the tolerance, $[x]_+$ denotes the positive clipping function that yields an output in the range $[0, \infty]$, η_λ denotes the learning rate of the penalty weighting gains, and $\nabla\mathbb{E}[R_{it}]$ denotes the gradient of the expected values of R_{it} with respect to the network parameters.

B. Mathematical Analysis

This study assumes that there exists only a pre-normalized constraint penalty term, that the optimal reward occurs when $R_{1t} \to 0$ (i.e., zero-penalty optimality), and that R_{1t} represents the estimated constraint penalty that is mostly in conflict with the primary reward ($0 \leq R_{it} \leq 1$; $\tau_1 = 1$). The total reward can now be expressed as:

$$R_t = R_{0t} - R_{0t}R_{1t}^2 - R_{1t}^3. \tag{S1}$$

1) Stability Analysis: To present the stability of ROGER, this work adopts a Lyapunov-based approach [23]. Consider the Lyapunov candidate function (R_{1t}), which measures the system's deviation from equilibrium:

$$V(R_{1t}) = 0.5R_{1t}^2.$$

Given that R_{1t} is a squared term, $V(R_{1t})$ is positive semi-definite, and $V(R_{1t}) = 0$ if and only if $R_{1t} = 0$. Its time derivative is:

$$\dot{V}(R_{1t}) = R_{1t}\dot{R}_{1t}.$$

Since the learning rule maximizes R_t, it follows that $\dot{R}_t \geq 0$, which can be expressed as:

$$\left(1 - R_{1t}^2\right)\dot{R}_{0t} \geq \left(2R_{0t} + 3R_{1t}\right)R_{1t}\dot{R}_{1t},$$

and thus,

$$\dot{V}(R_{1t}) \leq \gamma\dot{R}_{0t},$$

where $\gamma = (1 - R_{1t}^2)/(2R_{0t} + 3R_{1t})$.

This analysis can be divided into four key conditions.

1) *Near the constraint boundary:* When $R_{1t} \to 1$, $\gamma \approx 0$ and $\dot{V}(R_{1t}) \leq 0$, this indicates that ROGER remains stable near the constraint boundary. As the reward gains are clipped according to Equation 4, they remain the same even when the constraint boundary is violated; thus, the system is also stable in this case.
2) *During the convergence:* When $\dot{R}_{0t} \approx 0$, $\dot{V}(R_{1t}) \leq 0$, this indicates that ROGER maintains stability as R_{0t} converges.
3) *Positive reward with trade-off:* When $\dot{R}_{0t} \leq 0$ and $R_{0t} \geq 0$, $\dot{V}(R_{1t}) \leq 0$, this indicates that ROGER can trade off the primary reward for stability as long as the main reward remains non-negative.
4) *Other cases:* When $\gamma\dot{R}_{0t} \geq 0$, stability is not strictly guaranteed. However, as long as R_{0t} and R_{1t} do not change abruptly—e.g., with an appropriate learning rate—ROGER remains bounded according to other conditions.

In summary, when the system does not change abruptly (i.e., gentle system and learning dynamics), ROGER achieves partial stability under key conditions: near the constraint boundary (condition 1) and during convergence (condition 2), further supported by the ability to trade off R_{0t} for stability (condition 3).

2) Learning Analysis: This study also proves that, while being stable (i.e., involving gentle system and learning dynamics), adaptive adjustment of reward-weighting gains, as implemented in ROGER, ensures an increase in the primary reward R_{0t}. Specifically, the cumulative change in R_{0t} over time is always greater than or equal to its initial reward R_{0t_0}, i.e.,

$$\int dR_{0t} \geq R_{0t_0}, \forall t,$$

Since the learning rule maximizes R_t, its change satisfies:

$$dR_{0t} \geq \frac{3R_{1t}^2 + 2R_{1t}R_{0t}}{1 - R_{1t}^2}dR_{1t}. \tag{S2}$$

Considering the equivalent differential equation, where $x = R_{1t}$ and $y = R_{0t}$:

$$dy = \frac{3x^2 + 2xy}{1 - x^2}dx,$$

This equation can be written using the standard form $dy/dx - \frac{2x}{1-x^2}y = \frac{3x^2}{1-x^2}$, which can be solved using an integrating factor $\mu(x) = \exp\left(\int \frac{-2x}{1-x^2}dx\right) = 1 - x^2$.

After multiplying by the integrating factor, the equation becomes:

$$\frac{d}{dx}\left((1 - x^2)y\right) = (1 - x^2)\frac{3x^2}{1 - x^2},$$

Integrating both sides then yields:

$$\begin{aligned}(1 - x^2)y &= \int 3x^2 dx,\\ y &= \frac{x^3 + c}{1 - x^2},\end{aligned} \tag{S3}$$

where c is a constant.

Thus, integrating inequality S2 both sides then yields:

$$\begin{aligned}\int dR_{0t} &\geq \frac{R_{1t}^3}{1 - R_{1t}^2} + \frac{c}{1 - R_{1t}^2},\\ &\geq \frac{R_{1t}^3}{1 - R_{1t}^2} + R_{0t_0},\end{aligned}$$

Given that $0 \leq R_{1t} < 1, \forall t$, the inequality becomes:

$$\int dR_{0t} \geq \frac{R_{1t}^3}{1 - R_{1t}^2} + R_{0t_0} \geq R_{0t_0}. \tag{S4}$$

Figure S1 presents the trajectories of R_{0t} and R_{1t} obtained from locomotion learning experiments on the quadruped robot and MuJoCo hopper. The trajectories illustrate the continuous increase in the primary reward described by inequality S4, demonstrating successful learning.

Technique	Primary Gain (λ_{0t})	Penalty Gain (λ_{it})	Penalty Term ($\mathbf{R_{it}}$)
Penalty [4, 5]	1.0 (fixed)	tuned and fixed	arbitrary, e.g., $\lvert\tilde{x}_i\rvert$
Logarithmic CBF [6]	1.0 (fixed)	tuned and fixed	$[\ln\left(\frac{\tilde{x}_i}{\tau_i-\delta_i}\right)^2]_+$
Quadratic CBF [10]	1.0 (fixed)	tuned and fixed	$[\tilde{x}_i^2-(\tau_i-\delta_i)^2]_+$
PDO [12]	1.0 (fixed)	$[\lambda_{it}+\eta_\lambda(\tilde{R}_{it}-(\tau_i-\delta_i))]_+$	arbitrary, e.g., $\lvert\tilde{x}_i\rvert$
CRPO [16]	0.0 if exist $\tilde{R}_{it}>\tau_i-\delta_i$ 1.0 if $\tilde{R}_{it}\leq\tau_i-\delta_i$	1.0 if $\tilde{R}_{it}>\tau_i-\delta_i$ 0.0 if $\tilde{R}_{it}\leq\tau_i-\delta_i$	arbitrary, e.g., $\lvert\tilde{x}_i\rvert$
OL-AUX [24]	1.0 (fixed)	$\lambda_{it}+\eta_\lambda\nabla\mathbb{E}[R_{0t}]\nabla\mathbb{E}[\tilde{R}_{it}]$	arbitrary, e.g., $\lvert\tilde{x}_i\rvert$
ROGER (ours)	$1-\Delta_t$ (as in Equation 2)	$r_{\lambda_i t}\Delta_t$ (as in Equation 3)	arbitrary, e.g., $\lvert\tilde{x}_i\rvert$

TABLE S1: Comparison of the fundamental concepts behind various state-of-the-art model-free constrained RL techniques.

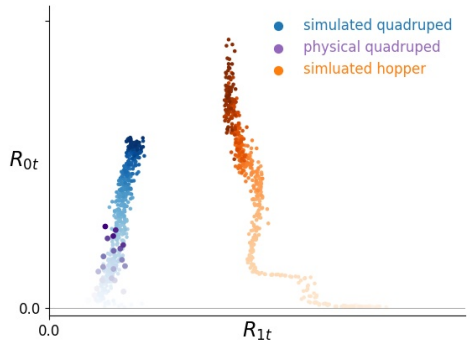


Fig. S1: Trajectories of R_{0t}–R_{1t} obtained from a simulated quadruped robot (blue), a physical quadruped robot (purple), and a simulated MuJoCo hopper (orange). Color intensity corresponds to learning time: bright dots indicate early training, while darker dots indicate later training.

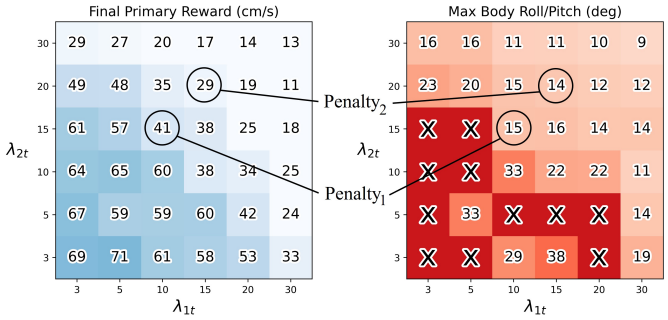


Fig. S2: (left) Final primary reward term and (right) maximum body roll/pitch angles, i.e., 99.9$^{\text{th}}$ percentile, obtained from the locomotion learning of a Unitree B1 quadruped robot with fixed gain penalties. The results were obtained from a grid search performed over different combinations of penalty weighting gains: roll penalty weighting gain (λ_{1t}) and pitch penalty weighting gain (λ_{2t}), with the primary weighting gain (λ_{0t}) fixed at 1.0. The figure presents the average and maximum values obtained from over ten repetitions in each condition. It should be noted that X indicates failure conditions where the robot falls during learning, defined as roll or pitch angles $\geq 45°$, while circles indicate the conditions presented in Figure 5.

C. Hyperparameter Selection

1) Fixed-Gain Penalty: The fixed-gain penalty is a traditional technique still employed in most works. However, as shown in Figure S2, the tuning process is relatively complex. On the one hand, low penalty weighting gains produce higher rewards, but the robot often falls during training. On the other hand, the use of higher penalty weighting gains typically results in lower rewards and suboptimal performance.

2) Control Barrier Function (CBF): The CBF technique is also employed. However, as shown in Figure S3, although the CBF receives higher rewards than the fixed-gain penalty in general, most conditions still experience constraint violations during learning, while the tuning process is as complex as that for the fixed-gain penalty.

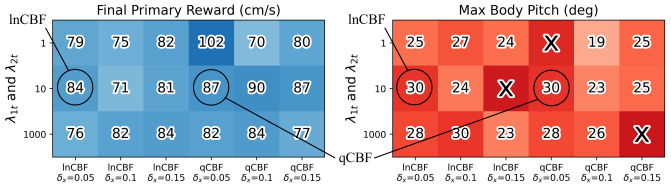


Fig. S3: (left) Final primary reward term and (right) maximum body roll/pitch angles, i.e., 99.9$^{\text{th}}$ percentile, obtained from the locomotion learning of a Unitree B1 quadruped robot with CBFs. The results were obtained from a grid search performed over different combinations of penalty weighting gains, CBF types, and parameters, with the primary weighting gain (λ_{0t}) fixed at 1.0. The figure presents the average and maximum values obtained from over ten repetitions in each condition. It should be noted that X indicates failure conditions where the robot falls during learning, defined as roll or pitch angles $\geq$ 45°, while circles indicate the conditions presented in Figure 5.

3) Primal-Dual Optimization (PDO): Figure S4 shows that PDO can prevent the robot from falling during training in some cases; however, it fails to meet the 10° constraint during learning in all cases and still requires careful hyperparameter tuning.

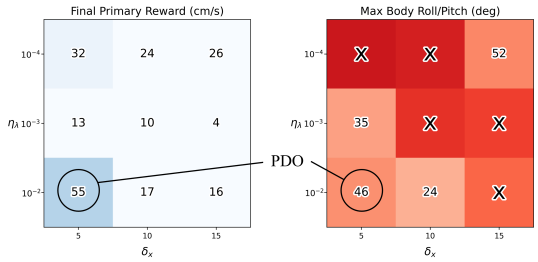


Fig. S4: (left) Final primary reward term and (right) maximum body roll/pitch angles, i.e., 99.9$^{\text{th}}$ percentile, obtained from locomotion learning of a Unitree B1 quadruped robot with PDO. The results were obtained from a grid search performed over different tolerance values. The figure presents the average and maximum values obtained from over ten repetitions in each condition. It should be noted that X indicates failure conditions where the robot falls during learning, defined as roll or pitch angles $\geq$ 45°, while circles indicate the conditions presented in Figure 5.

4) Online Learning with Auxiliary Loss (OL-AUX): Figure S5 shows that, although OL-AUX can prevent the robot from falling during training in some cases, it fails to meet the 10° constraint and still requires careful hyperparameter tuning.

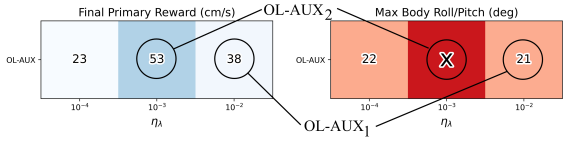


Fig. S5: (left) Final primary reward term and (right) maximum body roll/pitch angles, i.e., 99.9$^{\text{th}}$ percentile, obtained from locomotion learning of a Unitree B1 quadruped robot with OL-AUX. The figure presents the average and maximum values obtained from over ten repetitions in each condition. It should be noted that X indicates failure conditions where the robot falls during learning, defined as roll or pitch angles $\geq$ 45°, while circles indicate the conditions presented in Figure 5.

5) Constraint-Rectified Parameter Optimization (CRPO): Figure S6 shows that, although CRPO can prevent the robot from falling during training in this case, it still requires careful selection of hyperparameter values to meet the constraints.

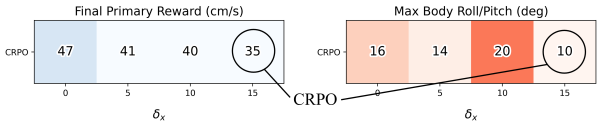


Fig. S6: (left) Final primary reward term and (right) maximum body roll/pitch angles, i.e., 99.9$^{\text{th}}$ percentile, obtained from locomotion learning of a Unitree B1 quadruped robot with CRPO. The results were obtained from a grid search performed over different tolerance values. The figure presents the average and maximum values obtained from over ten repetitions in each condition. It should be noted that a circle indicates the condition presented in Figure 5.

6) Reward-Oriented Gains Via Embodied Regulation (ROGER): Given that the key hyperparameter of ROGER is the constraint threshold (τ_i), Figure S7 presents the results after varying τ_1 (roll constraint threshold) and τ_2 (pitch constraint threshold) between 5° and 17° (i.e., 0.1, 0.2, and 0.3 radians). The figure shows that ROGER is less sensitive to the hyperparameter choices than other state-of-the-art methods since none of the tested cases cause the robot to fall. The maximum body pitch and roll angles appear to be environment-dependent, with the maximum roll/pitch being around 10° and the final values of the primary reward equal to those of the relaxed fixed-gain penalty shown in the bottom left of Figure S2.

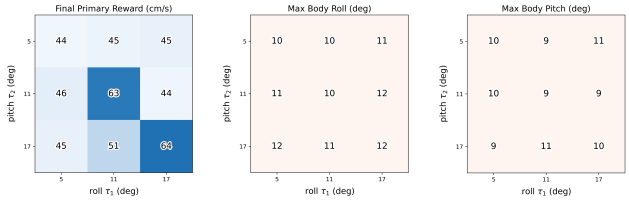


Fig. S7: (left) Final primary reward term, (middle) maximum body roll angle, and (right) maximum body pitch angles, i.e., 99.9$^{\text{th}}$ percentile, obtained from locomotion learning of a Unitree B1 quadruped robot with ROGER. The results were obtained from a grid search performed over different combinations of constraint thresholds. The figure presents the average and maximum values obtained from over ten repetitions in each condition.

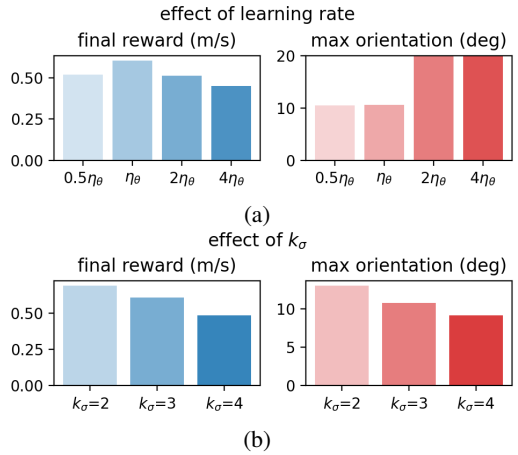


Fig. S8: (left) Final primary reward term and (right) maximum body orientation (i.e., both roll and pitch angles) obtained from locomotion learning of a Unitree B1 quadruped robot with ROGER, when (a) the learning rate is varied between $0.5\eta_\theta$ and $4\eta_\theta$ and (b) the parameter k_σ used for computing statistical estimates of penalties is varied between 2 and 4. The figures present the average and maximum values obtained from over ten repetitions in each condition.

D. MuJoCo Continuous Locomotion Learning

An additional experiment is included here to present the generalization of ROGER, using multiple MuJoCo continuous locomotion learning: Ant, Half Cheetah, Hopper, and Walker2D, as shown in Figure S9.

In Figure S10, four testing conditions were investigated: (1) the baseline reward function with fixed weighting gains obtained directly from the OpenAI Gymnasium, i.e., default; (2) ROGER with all three reward channels, i.e., ROGER; (3) ROGER with the primary reward and torque penalty channels, i.e., ROGER$_\tau$; and (4) ROGER with the primary reward and orientation penalty channels, i.e., ROGER$_\beta$.

Figure S11 presents the results with excessively low constraint thresholds (τ_i). The result shows that selecting constraint thresholds that are too low causes the robot to

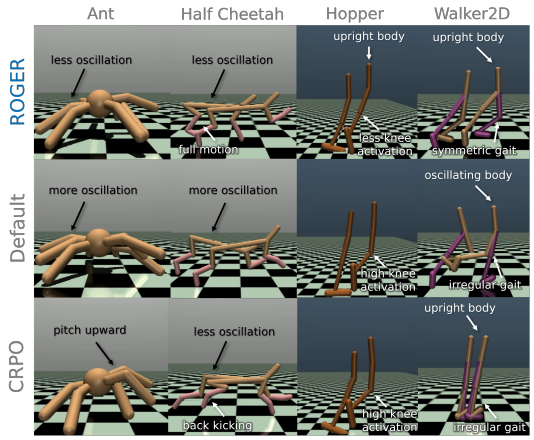


Fig. S9: Snapshots of different MuJoCo continuous locomotion learning tasks: Ant, Half Cheetah, Hopper, and Walker2D, trained with ROGER, CRPO [16], and the default reward function provided by OpenAI Gymnasium. High body oscillation indicates a greater orientation penalty and less stable locomotion, while higher joint activation indicates a higher action penalty, more torque usage, and a less energy-efficient policy. A video of this experiment is available at https://youtu.be/SpL4awVgDZM.

prioritize reducing the constraint penalties before improving the primary reward.

The overall experimental setup is presented in Section III-B, showing the comparison performed between the adaptive reward with ROGER, CRPO [16], and the default reward function provided by OpenAI Gymnasium (i.e., a properly tuned reward function). It should be noted that the humanoid is omitted because it was unable to learn proper locomotion—specifically, a walking distance of more than twice the body length, using the same set of hyperparameters listed below.

Hyperparameter	Value
learning rate	3e-4
entropy coefficient	1e-2
value coefficient	0.5
PPO clip rage	0.2
GAE discounted rate (γ_{GAE})	0.99
GAE bootstrapping (λ_{GAE})	0.95
trajectory length	2048
batch size	2048
training epoch	10
maximum gradient norm	0.5

TABLE S2: Hyperparameters for MuJoCo locomotion learning tasks.

Under the implementation of ROGER, multiple reward channels were employed in all locomotion tasks, with additional channels added as penalties/constraints according to the termination criteria. For two-dimensional locomotion tasks, including Half Cheetah, Hopper, and Walker2D, the channels consisted of the primary reward (healthy forward speed reward plus remaining terms), torque usage penalty, and

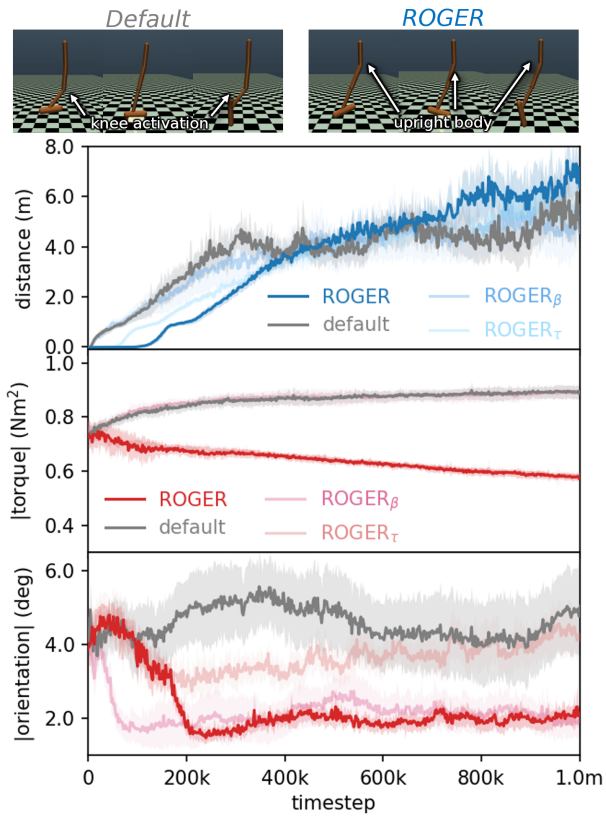


Fig. S10: Snapshots, hopping distance, joint torque usage, and absolute orientation deviation, obtained from a MuJoCo hopper trained using the default reward function from OpenAI Gymnasium and three different ROGER variants: ROGER with the primary reward, torque penalty, and orientation penalty; ROGER with the primary reward and torque penalty, i.e., ROGER$_\tau$; and ROGER with the primary reward and orientation penalty, i.e., ROGER$_\beta$. A video of the experiment is available at https://youtu.be/SpL4awVgDZM.

pitch deviation penalty. For three-dimensional tasks, i.e., Ant, the channels comprised the primary reward (healthy forward speed reward plus remaining terms), torque usage penalty, and pitch deviation penalty. Intuitively, for all tasks, the torque usage constraint threshold was set to 1 Nm (i.e., the maximum limit); the orientation deviation constraint threshold was 45° for Walker2D to allow for forward-backward hip swing, 45° for Ant to maintain the body, and 10° for Half Cheetah and Hopper to encourage upright body segments; and the height deviation constraint threshold for Ant was 0.2 m from the starting height.

The results in Tables S4–S3 show that ROGER outperformed the default reward functions provided by OpenAI Gymnasium and CRPO [16] in most cases, with the improvement of up to 100% in performance (i.e., walking distance) and lower constraint violations with up to 60% less torque usage and orientation deviation. In all tasks, constrained behaviors were observed progressively during learning.

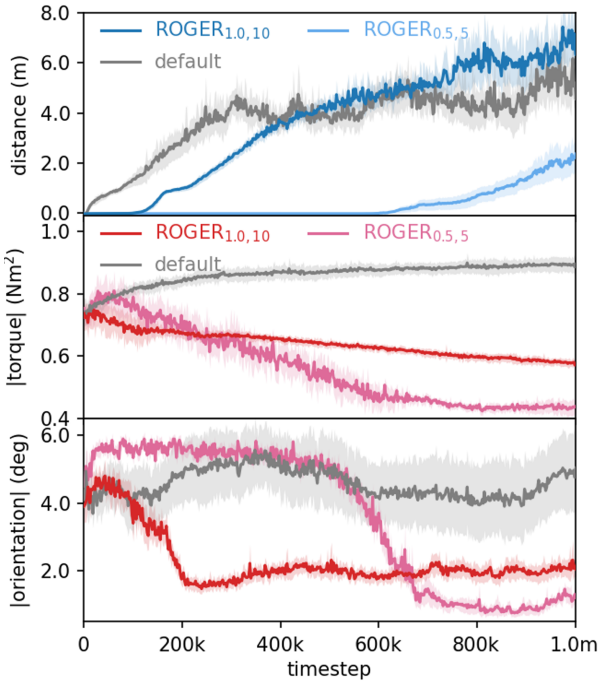


Fig. S11: (Top) Hopping distance, (middle) absolute joint torque, and (bottom) absolute orientation, from the MuJoCo hopper trained with default fixed weighting gains, ROGER with $\tau_1 = 1.0$ Nm and $\tau_2 \approx 10°$, and ROGER with $\tau_1 = 0.5$ Nm and $\tau_2 \approx 5°$. It should be noted that, due to the highly dynamic nature of the hopper, a stable hopping gait cannot be achieved without the robot falling. A video of the experiment is available at https://youtu.be/SpL4awVgDZM.

Robot/Environment	Default	CRPO	ROGER
Ant	**throughout learning / last 10 episodes**		
Torque (%)	10^{-10} / 0	0.82 / 1.18	10^{-6} / 0
Height (%)	10^{-5} / 0	10^{-7} / 0	10^{-5} / 0
Roll (%)	10^{-4} / 0	0.02 / 0.03	10^{-5} / 10^{-12}
Pitch (%)	10^{-8} / 10^{-13}	10^{-8} / 0	10^{-7} / 0
Half Cheetah	**throughout learning / last 10 episodes**		
Torque (%)	10^{-12} / 0	0.88 / 1.97	10^{-5} / 0
Pitch (%)	0.08 / 0.01	10^{-5} / 10^{-10}	4.7 / 0.01
Hopper	**throughout learning / last 10 episodes**		
Torque (%)	0.01 / 10^{-6}	0.01 / 0.01	10^{-7} / 0
Pitch (%)	0.14 / 0.21	0.01 / 10^{-5}	10^{-11} / 0
Walker	**throughout learning / last 10 episodes**		
Torque (%)	10^{-7} / 10^{-10}	0.01 / 0.06	10^{-8} / 0
Pitch (%)	0.07 / 0.08	8.07 / 6.76	10^{-5} / 10^{-13}

TABLE S3: Percentages of constraint violations in four continuous locomotion tasks obtained with default reward functions provided by OpenAI Gymnasium, representing a properly tuned reward function, CRPO [16], and ROGER after 1m training timesteps. A video of the experiment is available at https://youtu.be/SpL4awVgDZM.

Robot/Environment	Default	ROGER	Change (p-value)
Ant	**Default**: less oscillation and front leg activation **ROGER**: more oscillation and front leg activation		
Distance (m), reward	106.02	117.45	+10% (p≈0.47)
Torque (Nm), $\tau = 1.00$	0.25	0.39	+58% (p≪0.01)
Height (m), $\tau = 0.20$	0.00	0.00	+0% (p≈0.50)
Roll (°), $\tau = 45.00$	12.37	10.00	−19% (p≈0.09)
Pitch (°), $\tau = 45.00$	8.27	5.35	−35% (p≪0.01)
Half Cheetah	**Default**: body oscillation, lean forward **ROGER**: maintained body around 0°		
Distance (m), reward	46.10	92.25	+100% (p≪0.01)
Torque (Nm), $\tau = 1.00$	0.76	0.55	−27% (p≪0.01)
Pitch (°), $\tau = 10.00$	6.13	5.53	−10% (p≪0.01)
Hopper	**Default**: higher knee activation **ROGER**: minor knee activation		
Distance (m), reward	5.23	6.57	+25% (p≪0.01)
Torque (Nm), $\tau = 1.00$	0.78	0.33	−57% (p≪0.01)
Pitch (°), $\tau = 10.00$	5.10	2.06	−59% (p≪0.01)
Walker2D	**Default**: poor leg coordination with body oscillation **ROGER**: exhibit walking gait with upright body		
Distance (m), reward	5.99	11.65	+94% (p≪0.01)
Torque (Nm), $\tau = 1.00$	0.91	0.40	−56% (p≪0.01)
Pitch (°), $\tau = 45.00$	21.00	8.31	−59% (p≪0.01)

TABLE S4: Average performance obtained from four continuous locomotion tasks trained with default reward functions provided by OpenAI Gymnasium, representing a properly tuned reward function, and ROGER after 1m training timesteps. A video of this experiment is available at https://youtu.be/SpL4awVgDZM.

Robot/Environment	CRPO	ROGER	Change (p-value)
Ant	**CRPO**: body pitches upward **ROGER**: more oscillation and front leg activation		
Distance (m), reward	57.50	117.45	+104% (p≈0.10)
Torque (Nm), $\tau = 1.00$	0.93	0.39	−58% (p≪0.01)
Height (m), $\tau = 0.20$	0.00	0.00	+0% (p≈0.16)
Roll (°), $\tau = 45.00$	16.04	10.00	−37% (p≈0.03)
Pitch (°), $\tau = 45.00$	10.88	5.35	−50% (p≪0.01)
Half Cheetah	**CRPO**: back leg kicks 45° backward (back kicking) **ROGER**: back leg swings fully beneath the body		
Distance (m), reward	82.48	92.25	+12% (p≈0.78)
Torque (Nm), $\tau = 1.00$	0.98	0.55	−44% (p≪0.01)
Pitch (°), $\tau = 10.00$	3.66	5.53	+51% (p≪0.01)
Hopper	**CRPO**: higher knee activation **ROGER**: minor knee activation		
Distance (m), reward	6.42	6.57	+2% (p≈0.50)
Torque (Nm), $\tau = 1.00$	0.80	0.33	−58% (p≪0.01)
Pitch (°), $\tau = 10.00$	4.38	2.06	−53% (p≪0.01)
Walker2D	**CRPO**: small step, overly prioritize stability **ROGER**: exhibit walking gait with upright body		
Distance (m), reward	1.91	11.65	+510% (p≪0.01)
Torque (Nm), $\tau = 1.00$	0.94	0.40	−57% (p≪0.01)
Pitch (°), $\tau = 45.00$	7.62	8.31	+9% (p≈0.32)

TABLE S5: Average performance obtained from four continuous locomotion tasks trained with CRPO [16] and ROGER after 1m training timesteps. A video of the experiment is available at https://youtu.be/SpL4awVgDZM.