Stage-Wise Reward Shaping for Acrobatic Robots: A Constrained Multi-Objective Reinforcement Learning Approach

Dohyeong Kim[1*], Hyeokjin Kwon[2*], Junseok Kim[1], Gunmin Lee[1], and Songhwai Oh[1,2]

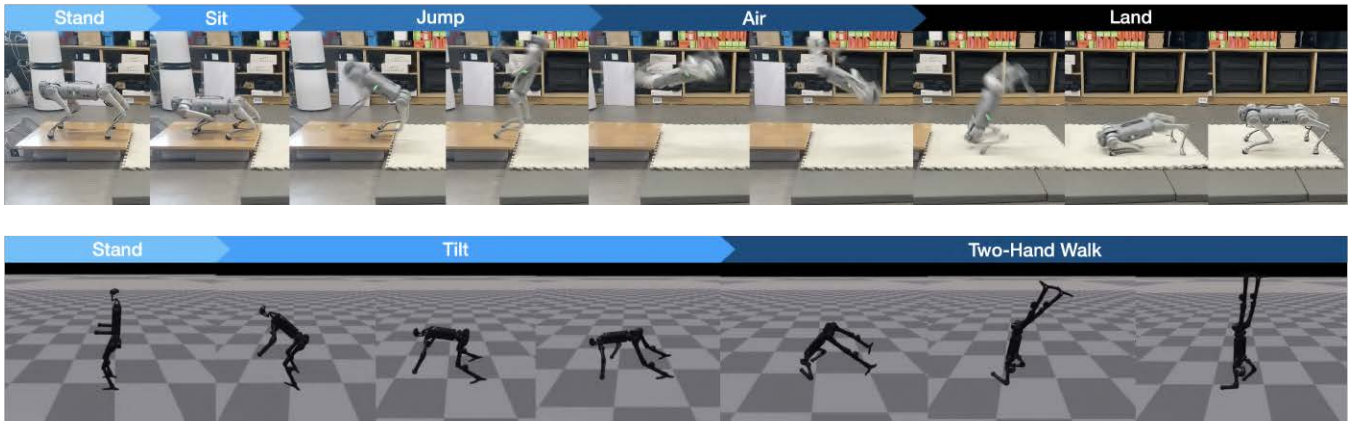


Fig. 1: **Snapshots of robot movements with stage transitions.** The top row shows a quadrupedal robot performing a back-flip in a real-world environment, and the bottom row shows a humanoid performing a two-hand walk in simulation. Each sequence of snapshots is annotated at the top with the current stage of the episode.

***Abstract*— As the complexity of tasks addressed through reinforcement learning (RL) increases, the definition of reward functions also has become highly complicated. We introduce an RL method aimed at simplifying the reward-shaping process through intuitive strategies. Initially, instead of a single reward function composed of various terms, we define multiple reward and cost functions within a constrained multi-objective RL (CMORL) framework. For tasks involving sequential complex movements, we segment the task into distinct stages and define multiple rewards and costs for each stage. Finally, we introduce a practical CMORL algorithm that maximizes objectives based on these rewards while satisfying constraints defined by the costs. The proposed method has been successfully demonstrated across a variety of acrobatic tasks in both simulation and real-world environments. Additionally, it has been shown to successfully perform tasks compared to existing RL and constrained RL algorithms. Our code is available at `https://github.com/rllab-snu/Stage-Wise-CMORL`.**

I. INTRODUCTION

Recently, reinforcement learning (RL) has driven significant progress in real-world robotic applications [1]–[5]. Quadrupedal robots have demonstrated stable locomotion on rough terrain [1], [6] and performed parkour-like stunts [7], [8], while bipedal robots have successfully tackled challenging tasks such as jumping and running [9], [10]. In order to accomplish such legged robot tasks, a reward function should be defined considering multiple factors, such as task performance, safety, and energy efficiency. Consequently, reward functions are generally formalized as a sum of various terms related to performance, safety, and regularization [1], [2], [6]. However, due to the numerous terms, the reward-shaping process, defining each term and its respective weight, can be laborious and challenging. Simplifying this process is essential to apply RL to a broader range of tasks.

In the case of acrobatic tasks involving complex movements, such as rolls and back-flips, the difficulty of designing rewards increases significantly. By taking the back-flip as an example, this task requires a focus on jumping at the beginning of the episode and on landing after the jump. As a result, the proportions of reward terms related to jumping and landing should be adjusted dynamically, complicating the reward-shaping process. Alternatively, imitation RL methods using motion capture data or animatronic data have been developed [11]–[13], wherein the reward is defined to minimize the pose difference between the robot and the collected data. However, these methods are expensive as they require collecting extensive data for each task. Therefore, a method is required that can intuitively design reward functions without relying on additional imitation data.

In this paper, we propose an RL method that defines multiple reward functions in a stage-wise manner by utilizing a constrained multi-objective RL (CMORL) framework [14]. Examples of the stages and representative results are presented in Fig. 1. To simplify the reward-shaping process, our approach does not integrate multiple terms into a single

*Equal contribution.

[1]D. Kim, J. Kim, G. Lee, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 08826, Korea (e-mail: {dohyeong.kim, junseok.kim, gunmin.lee}@rllab.snu.ac.kr, songhwai@snu.ac.kr).

[2]H. Kwon ans S. Oh are with the Interdisciplinary Program in Artificial Intelligence and ASRI, Seoul National University, Seoul 08826, Korea (e-mail: hyeokjin.kwon@rllab.snu.ac.kr).

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01190, [SW Star Lab] Robot Learning: Efficient, Safe, and Socially-Acceptable Machine Learning).

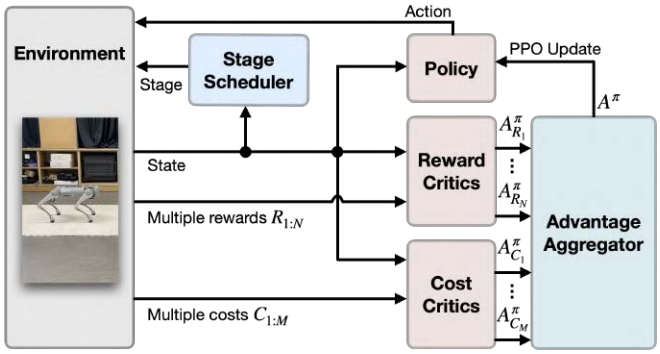


Fig. 2: **Overview of the proposed framework.** An environment provides multiple rewards and costs, and critics compute value estimates for each reward and cost. These estimates are aggregated to calculate the overall advantage, as detailed in Sec. IV-B, which is subsequently used for policy updates. Additionally, a stage scheduler updates the current stage based on a user-defined rule for stage transitions.

reward. Instead, each term is treated as an independent reward or cost function within the CMORL framework, where the cost functions correspond to safety-related terms, such as body collision and joint limit. In addition, to facilitate reward shaping for acrobatic tasks requiring a series of complex movements, we propose segmenting tasks into several stages and defining reward and cost functions for each stage. The overall framework is presented in Fig. 2, and examples of the reward and cost definitions are provided in Table I. In this framework, a new variant of proximal policy optimization (PPO) [15] adapted for CMORL, named *constrained multi-objective PPO (CoMOPPO)*, is introduced for policy updates. Also, to ensure successful real-world deployment, sim-to-real techniques, such as domain randomization [3], [8] and teacher-student distillation [1], are employed.

The proposed method has been applied to a range of acrobatic tasks on quadrupedal and humanoid robots: *two-hand walks, back-flips, side-flips*, and *side-rolls*. For real-world evaluation, the side-roll, back-flip, and two-hand walk tasks were successfully demonstrated using a quadrupedal robot. Moreover, we have compared the proposed method with existing RL [15] and constrained RL [16] algorithms. The results confirmed the effectiveness of the proposed method, as it was the only method that completed the tasks. In conclusion, our contributions are threefold:

- We propose a new reward-shaping process within the CMORL framework, where multiple reward and cost functions are defined stage-wisely.
- We introduce a practical CMORL algorithm that adapts PPO to handle multiple objectives and constraints.
- The proposed method has successfully demonstrated various acrobatic tasks, both in simulation and in real-world environments.

II. RELATED WORK

A. Constrained Multi-Objective Reinforcement Learning

Multi-objective RL (MORL) is divided into single-policy methods [17], [18], which aim to find one Pareto optimal policy, and multi-policy methods [19]–[23], which aim to find a set of Pareto optimal policies. Single-policy methods combine multiple objectives into a single objective using utility functions [24] or preference vectors [17], [18], allowing existing RL algorithms to be applied for policy updates. Multi-policy methods either simultaneously update policies for multiple preferences [20], [22] or train a universal policy that can represent a variety of policies by conditioning on preferences [19], [23]. Among these, LP3 [25] and CoMOGA [14] are CMORL algorithms that extend existing MORL algorithms to consider constraints using Lagrangian [26] and primal [27] approaches, respectively. The proposed algorithm, CoMOPPO, can be viewed as a single-policy method that simplifies the implementation of CoMOGA.

B. Reinforcement Learning for Legged Robots

Advances in simulations, such as Isaac Gym [28], have made it possible to directly deploy RL policies trained in simulations into real-world environments, significantly increasing efficiency and reducing risk. Leveraging these advances, sim-to-real techniques, such as terrain curriculum, have enabled quadrupedal robots to successfully navigate challenging terrains, such as slippery surfaces and steep slopes [1], [2], [4]–[6]. Furthermore, there have been works that enable dynamic parkour-like movements, such as long jumps and two-legged walking, by using novel reward definitions and state representation approaches [3], [7], [8]. To simplify the definition of reward functions, constrained RL algorithms [29] have been employed for legged robots [30], [31]. These methods exclude safety-related terms, such as body collisions and joint limits, from the reward function and instead use them to define explicit constraints. These algorithms have demonstrated the effectiveness of constrained RL through robustness to reward weights. However, the tasks addressed by them are primarily limited to locomotion. We expanded these approaches to the CMORL framework to perform tasks requiring more complex acrobatic movements.

III. BACKGROUND

A. Constrained Multi-Objective Markov Decision Processes

A constrained multi-objective Markov decision process (CMOMDP) is defined as $\langle S, A, P, \rho, \gamma, R_{1:N}, C_{1:M} \rangle$ with a state space S, an action space A, a transition model P, an initial state distribution ρ, a discount factor γ, N reward functions $R_i(s,a,s')|_{i=1}^N$, and M cost functions $C_j(s,a,s')|_{j=1}^M$. A policy is defined as $\pi : S \mapsto \mathcal{P}(A)$, where $\pi(a|s)$ denotes the probability of executing action a in state s. A trajectory is defined as $\tau = \{s_0, a_0, s_1, a_1, ...\}$, where $s_0 \sim \rho$, $a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim P(\cdot|s_t, a_t)\ \forall t$. Action value, state value, and advantage functions for the rewards are defined as $Q^\pi_{R_i}(s,a) := \mathbb{E}_{\tau\sim\pi}[\sum_t \gamma^t R_i(s_t, a_t, s_{t+1})]$, where $s_0 = s$ and $a_0 = a$, $V^\pi_{R_i}(s) := \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi_{R_i}(s,a)]$, and $A^\pi_{R_i}(s,a) := Q^\pi_{R_i}(s,a) - V^\pi_{R_i}(s)$. Similarly, the value and advantage functions for the costs are defined by substituting R_i with C_j. The reward and cost functions are used to construct objectives and constraints in a CMORL problem, respectively.

B. CMORL Problem Setup

A CMORL problem is defined as follows:

$$\begin{aligned} &\max_\pi J_{R_i}(\pi)\ \forall i \in \{1, ..., N\} \\ &\textbf{s.t.}\ J_{C_j}(\pi) \le d_j/(1-\gamma)\ \forall j \in \{1, ..., M\}, \end{aligned} \tag{1}$$

TABLE I: **Example of reward and cost definitions for the back-flip task.** Rewards and costs are defined in the first five lines and the following five lines, respectively. In the velocity reward, $\mathbf{1}_{\text{turn}}$ is 1 if the robot completes a turn and 0 otherwise. In the balance reward, $\hat{z}_B$ and $\hat{z}_W$ are the z-axis unit vectors of the base and world frame, respectively. In the style reward, q_j^{default} represents the default position of the jth joint. In the contact-related costs, $I_C^{\text{specified}}$ denotes the set of indices of specified links that make contact. During stages other than the jump stage, the foot contact cost outputs the predefined threshold value as the cost is undefined in these phases.

stage	stand	sit	jump	air	land	threshold
reward functions						
base height	$-\lvert p_z - 0.35\rvert$	$-\lvert p_z - 0.2\rvert$	$\mathbf{1}_{p_z \leq 0.5} \cdot p_z$	$\mathbf{1}_{p_z \leq 0.5} \cdot p_z$	$-\lvert p_z - 0.35\rvert$	
base velocity	$-(v_x^2 + v_y^2 + \omega_z^2)$	$-(v_x^2 + v_y^2 + \omega_z^2)$	$-\mathbf{1}_{\text{turn}} \cdot \omega_y$	$-s \cdot \omega_y$	$-(v_x^2 + v_y^2 + \omega_z^2)$	
base balance	$-\angle(\hat{z}_B, \hat{z}_W)$	$-\angle(\hat{z}_B, \hat{z}_W)$	$-\lvert\angle(\hat{y}_B, \hat{z}_W) - \pi/2\rvert$	$-\lvert\angle(\hat{y}_B, \hat{z}_W) - \pi/2\rvert$	$-\angle(\hat{z}_B, \hat{z}_W)$	
energy	$-\sum_j \tau_j^2$	$-\sum_j \tau_j^2$	$-\sum_j \tau_j^2$	$-\sum_j \tau_j^2$	$-\sum_j \tau_j^2$	
style	$-\sum_j (q_j - q_j^{\text{default}})^2$	$-\sum_j (q_j - q_j^{\text{default}})^2$	$-\sum_j (q_j - q_j^{\text{default}})^2$	$-\sum_j (q_j - q_j^{\text{default}})^2$	$-\sum_j (q_j - q_j^{\text{default}})^2$	
cost functions						
foot contact	-	-	$\mathbf{1}_{\lvert I_C^{\text{foot,rear}}\rvert = 0}$	-	-	0.25
body contact	$\mathbf{1}_{\lvert I_C^{\text{body}}\rvert > 0}$	$\mathbf{1}_{\lvert I_C^{\text{body}}\rvert > 0}$	$\mathbf{1}_{\lvert I_C^{\text{body}}\rvert > 0}$	$\mathbf{1}_{\lvert I_C^{\text{body}}\rvert > 0}$	$\mathbf{1}_{\lvert I_C^{\text{body}}\rvert > 0}$	0.025
joint position	$\frac{1}{J}\sum_j \mathbf{1}_{q_j > q_j^{\max} \Vert q_j < q_j^{\min}}$	$\frac{1}{J}\sum_j \mathbf{1}_{q_j > q_j^{\max} \Vert q_j < q_j^{\min}}$	$\frac{1}{J}\sum_j \mathbf{1}_{q_j > q_j^{\max} \Vert q_j < q_j^{\min}}$	$\frac{1}{J}\sum_j \mathbf{1}_{q_j > q_j^{\max} \Vert q_j < q_j^{\min}}$	$\frac{1}{J}\sum_j \mathbf{1}_{q_j > q_j^{\max} \Vert q_j < q_j^{\min}}$	0.025
joint velocity	$\frac{1}{J}\sum_j \mathbf{1}_{\lvert\dot{q}_j\rvert > \dot{q}_j^{\max}}$	$\frac{1}{J}\sum_j \mathbf{1}_{\lvert\dot{q}_j\rvert > \dot{q}_j^{\max}}$	$\frac{1}{J}\sum_j \mathbf{1}_{\lvert\dot{q}_j\rvert > \dot{q}_j^{\max}}$	$\frac{1}{J}\sum_j \mathbf{1}_{\lvert\dot{q}_j\rvert > \dot{q}_j^{\max}}$	$\frac{1}{J}\sum_j \mathbf{1}_{\lvert\dot{q}_j\rvert > \dot{q}_j^{\max}}$	0.025
joint torque	$\frac{1}{J}\sum_j \mathbf{1}_{\lvert\tau_j\rvert > \tau_j^{\max}}$	$\frac{1}{J}\sum_j \mathbf{1}_{\lvert\tau_j\rvert > \tau_j^{\max}}$	$\frac{1}{J}\sum_j \mathbf{1}_{\lvert\tau_j\rvert > \tau_j^{\max}}$	$\frac{1}{J}\sum_j \mathbf{1}_{\lvert\tau_j\rvert > \tau_j^{\max}}$	$\frac{1}{J}\sum_j \mathbf{1}_{\lvert\tau_j\rvert > \tau_j^{\max}}$	0.025

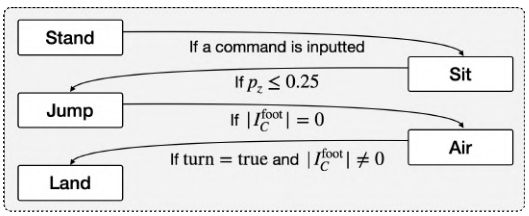


Fig. 3: **Example of stage transitions for the back-flip.** The task start in the stand stage. Upon receiving a back-flip command, the robot attempts to sit. When the base height drops below $0.25\ m$, it transitions to the jump stage. During this stage, the robot attempts to jump, transitioning to the air stage once all feet detach from the ground. After completing the aerial motion, the robot transitions to the landing stage as soon as at least one foot makes contact with the ground.

where $J_{R_i}(\pi) := \mathbb{E}_{\tau\sim\pi}\left[\sum_t \gamma^t R_i(s_t, a_t, s_{t+1})\right]$, and d_j is a threshold of the jth constraint. The target of the CMORL problem finds a *constrained-Pareto (CP)* optimal policy [14]. Given two feasible policies $\pi_1, \pi_2 \in \{\pi | J_{C_j}(\pi) \leq d_j/(1-\gamma)\ \forall j\}$, if $J_{R_i}(\pi_1) \leq J_{R_i}(\pi_2)\ \forall i$, π_1 is *dominated* by π_2. If a policy is not dominated by any other policies, the policy is CP optimal. Given that a CP optimal policy is not unique, a specific policy can be obtained by transforming multiple objectives into a single objective using a *preference vector* $\omega \in \Omega = \{v \in \mathbb{R}^N | \sum_i v_i = 1,\ v_i \geq 0\}$. This transformation can be achieved either by linearly weight-summing the preference vector and objectives [23], [24], [32] or by using other scalarization methods [17], [18]. Subsequently, single-objective RL algorithms can be applied to obtain the policy.

IV. PROPOSED METHOD

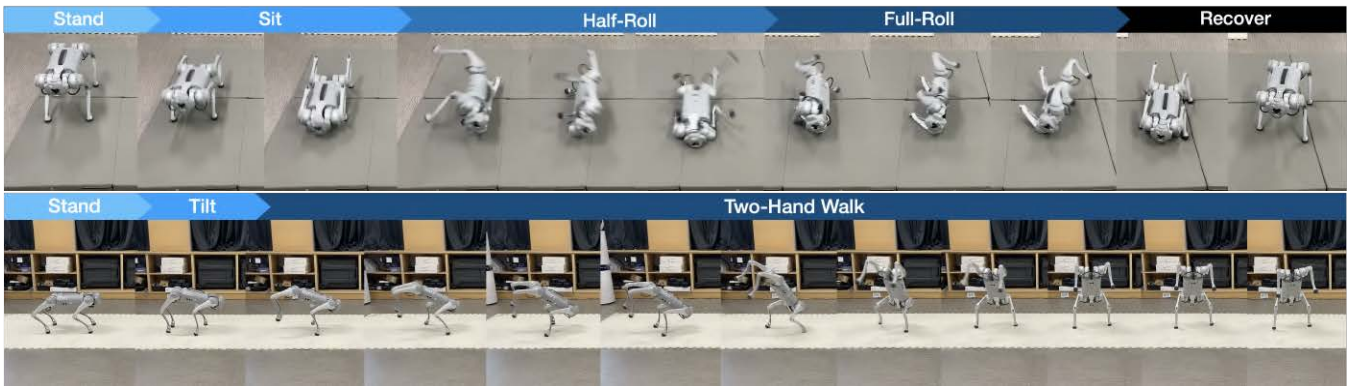
Now, we introduce a new CMORL framework, which enables intuitive reward shaping for acrobatic tasks. The proposed method consists of three main parts: **1)** stage-wise reward shaping, **2)** a policy update rule handling multiple objectives and constraints, and **3)** sim-to-real techniques for deploying policies trained in simulation to real-world. In the rest of the section, the details of each part will be described.

A. Stage-Wise Reward Shaping

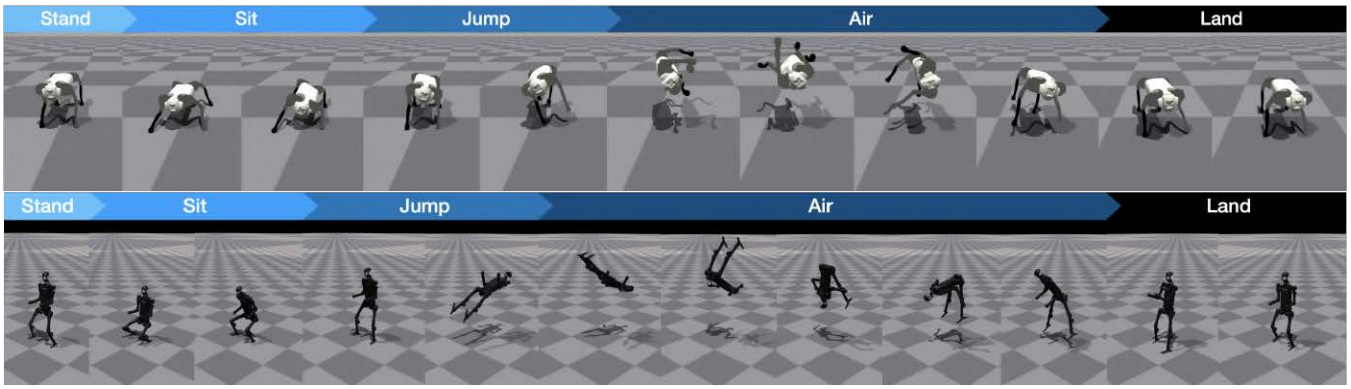
In general, a reward function consists of several terms including task-related terms, regularization terms, and safety-related terms. Instead of integrating all terms into a scalar reward, we use a CMORL framework which maximizes multiple objectives corresponding to each reward term and satisfying constraints corresponding to the safety-related terms. Furthermore, for a complex task requiring sequential movements, it is required to adjust weights of individual reward terms dynamically, which further increases the difficulty of reward shaping. To resolve this issue, we propose to divide a task into a sequence of *stages* and define reward and cost functions in a stage-wise manner. Since segmenting tasks into stages clarifies the required motions for each stage, the reward-shaping process becomes straightforward. An example of the stages and definitions of the reward and cost functions for the back-flip task are provided in Fig. 3 and Table I, respectively. Using this example, we provide an overview of how the stages are segmented and how the reward and cost functions are defined.

1) Stage Transitions: As shown in Fig. 1, to perform a back-flip, the robot should remain in the standby mode until a command is received. Once the command is inputted, the robot sits down slightly, then jumps to rotate in the air, and finally lands. Through this insight, the task can be divided into five stages named *Stand-Sit-Jump-Air-Land*.

2) Reward and Cost Functions: In the stand, sit, and land stages, the robot is required to remain stationary; therefore, the base velocity reward is defined as the negative of the current velocity. Conversely, in the jump and air stages, where the robot should rotate backward, the velocity reward is set to the y-directional angular velocity. Also, to ensure sufficient jump height during these stages, the height reward is set to the base height. In order to prevent tilting during the jump and air stages, the balance reward is set to remain the angle between the y-axis of the base and the z-axis of the world perpendicular. In the other stages, the reward is set to minimize the angle between the z-axis of the base and the world frames to maintain the robot upright. The energy and style rewards are used as regularization to ensure natural motions. The body contact cost prevents the robot from falling over, while costs associated with joint position, velocity, and torque are implemented to limit those values within their respective ranges. The foot contact cost is defined specifically for jumping with the rear legs; the cost is incurred if the rear legs detach before the front legs.



(a) Sim-to-Real Tasks



(b) Simulation Tasks

Fig. 4: **Snapshots of motion sequences generated by trained policies.** The first two rows show the Unitree Go1 robot performing side-roll and two-hand walk tasks in the real-world environment. The remaining two rows illustrate the Go1 robot performing the side-flip task and the H1 robot performing the back-flip task in simulation.

B. CMORL Policy Update

In this section, we introduce a policy update rule, termed *constrained multi-objective PPO (CoMOPPO)*, designed to maximize multiple objectives while satisfying constraints. According to Kim et al. [14], convergence to a CP optimal policy can be achieved by aggregating the advantage functions of rewards and costs through a weighted summation, where the weights satisfy specific conditions, and updating the policy using TRPO [33] with the aggregated advantage. It can be written as follows:

$$\pi_{t+1} = \underset{\pi}{\operatorname{argmax}} \underset{\tau\sim\pi_t}{\mathbb{E}} \left[\frac{\pi(a|s)}{\pi_t(a|s)} A^{\pi_t}(s,a) \right] \\ \mathbf{s.t.}\ \mathbb{E}_{\tau\sim\pi_t} \left[D_{\mathrm{KL}}(\pi_t(\cdot|s)||\pi(\cdot|s)) \right] \leq \epsilon, \tag{2}$$

where $A^{\pi_t}(s,a) := \sum_i \nu_{t,i} A^{\pi_t}_{R_i}(s,a) - \sum_j \lambda_{t,j} A^{\pi_t}_{C_j}(s,a)$, ϵ is a trust region size, and D_{KL} is the KL divergence. For the condition on the weights, ν and λ, please refer to Theorem 4.2 in [14].

In order to properly determine the weights, ν and λ, we use **1)** reward normalization and **2)** standard deviation of advantage functions. First, in the CMORL setting, each reward function operates at a different scale, making it essential to adjust them to a consistent level. To this end, we apply reward normalization for each reward and stage, and train the value functions using the normalized rewards. This approach automatically adjusts the ratio of each objective to a consistent level. Next, it is important to maintain consistency not only in the scale of the rewards but also in the ratio between objectives and constraints. Without a consistent ratio, the policy may be updated to over-maximizing objectives rather than satisfying constraints, potentially destabilizing the training process. To resolve this, we normalize the reward and cost advantages by their respective standard deviations, ensuring the policy to be updated with a consistent ratio of objectives to constraints. As a result, the proposed rule for advantage aggregation as follows:

$$A^{\pi} = \frac{A^{\pi}_R}{\mathrm{Std}[A^{\pi}_R]} - \eta \sum_j \frac{A^{\pi}_{C_j}}{\mathrm{Std}[A^{\pi}_{C_j}]} \mathbf{1}_{(J_{C_j}(\pi) > d_j)}, \tag{3}$$

where Std denotes the standard deviation, $A^{\pi}_R := \sum_i \omega_i \hat{A}^{\pi}_{R_i}$ with a given preference ω, $\hat{A}^{\pi}_{R_i}$ is the advantage function calculated from the normalized rewards, and the hyper-parameter η serves as the ratio of constraints, as done in [16]. With the aggregated advantage functions, the policy can be updated using (2). However, to simplify the implementation, we apply a PPO update, which is formulated as follows [15]:

$$\pi_{t+1} = \underset{\pi}{\operatorname{argmax}} \underset{\tau\sim\pi_t}{\mathbb{E}} \left[\min(r_t A^{\pi_t}, \mathrm{clip}(r_t, 1-\epsilon, 1+\epsilon) A^{\pi_t}) \right],$$

where $r_t := \pi(a|s)/\pi_t(a|s)$, and ϵ is a hyper-parameter.

C. Sim-to-Real Techniques

In order to deploy policies trained in simulation to real-world environments, we use two widely-used sim-to-real

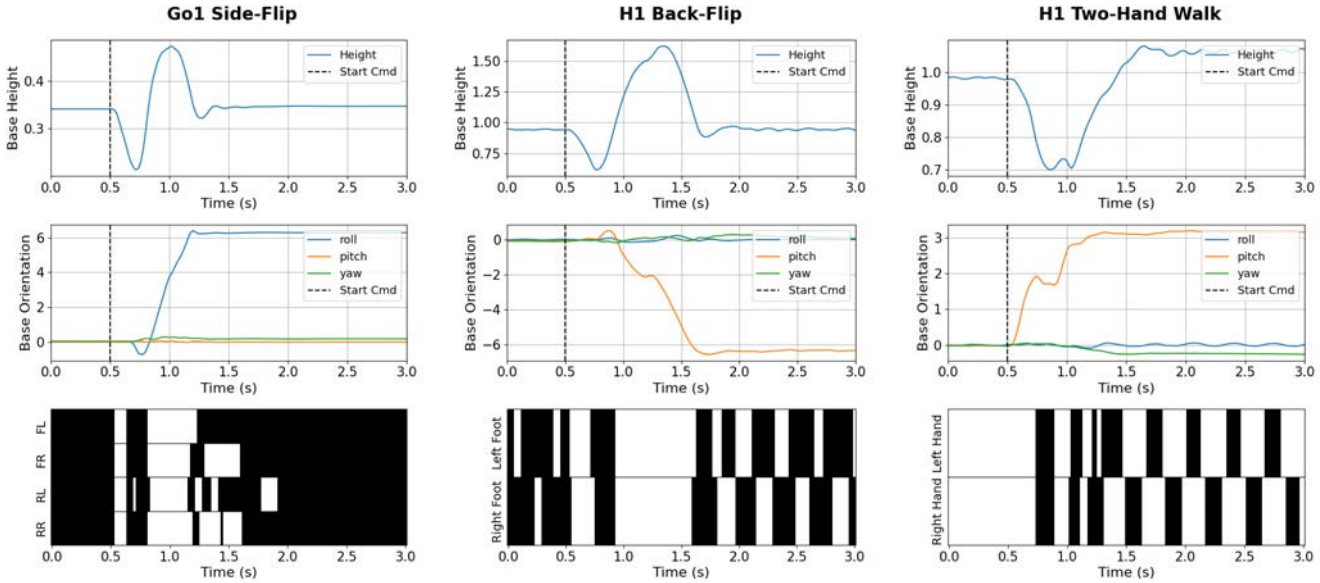


Fig. 5: **Simulation experiment results.** The first two rows show the changes in height and body orientation, while the last indicates whether the specified parts of the robot are in contact with the ground (black) or not (white) in each task.

techniques: **1)** domain randomization and **2)** teacher-student learning [1], [30], [34]. For domain randomization, we follow an RL approach proposed in [2], which involves randomizing motor strength and offset, gravity, friction, restitution, noise in joint positions and velocities, and base orientation. Details on the range of randomization are provided in the appendix of [2]. Next, teacher-student learning [1] is used to distill a teacher policy, which uses states including privileged information, into a student policy that relies solely on sensor data, such as joint position and base orientation. The process is similar to the method proposed in [1], but the action execution differs slightly. In our approach, the actions of the teacher and student policies are alternately provided to the environment at fixed intervals. This helps in collecting a dataset for teacher-student learning that closely aligns with the student policy.

V. EXPERIMENTS

This section describes the tasks and their corresponding results for both simulation and real-world environments. Details on the motions generated by the trained policies, along with the corresponding reward and cost functions for each task, can be found in the attached video.

A. Environmental Setup

We use the Isaac Gym simulator [28] due to its effectiveness in sim-to-real transfer and its flexibility in creating a wide range of tasks across various robotic platforms. In simulation experiments, we employ two types of robots from Unitree Robotics: Go1, a quadrupedal robot, and H1, a humanoid [35]. The quadrupedal robot comprises 17 body links and 12 motors, while the humanoid robot features 20 body links and 19 motors. In real-world experiments, we deploy the quadruped robot, Go1, through sim-to-real techniques as discussed in Section IV-C.

The state representation includes body orientation, joint positions and velocities, commands, as well as the previous action. For the teacher policy, privileged information—such as linear and angular velocities, height, and foot contact, which are difficult to access in the real-world but available in simulation—is additionally used.

B. Task Details

We have designed four acrobatic tasks: *back-flip, side-flip, side-roll*, and *two-hand walk*. The back-flip and two-hand walk tasks are implemented on both types of robots, while the side-roll and side-flip tasks are designated for the quadruped. Snapshots of each task are shown in Fig. 1 and Fig. 4, and the following are descriptions of each task.

1) Back-Flip: The robot jumps backward into the air, rotates 360 degrees without touching the ground, and lands on its feet in its initial pose. The stage transition, along with the reward and cost functions, is discussed in Section IV-A.

2) Side-Flip: This task is similar to the back-flip, but the robot jumps to the right side instead of backward. The stage transitions remain identical to those of the back-flip.

3) Side-Roll: The robot performs a full roll along its right side, returning to its original pose upon completion. This task is segmented into five stages: *Stand*, where the robot remains upright; *Sit*, where the robot lowers itself to prepare for the roll; *Half-roll*, where the robot lies on its back; *Full-roll*, where the robot completes the roll; and *Recover*, where the robot returns to its default pose and orientation.

4) Two-Hand Walk: The robot performs walking using its hands or front legs, while maintaining balance. This task is divided into three stages: *Stand*, where the robot maintains its default pose; *Tilt*, where the robot lowers its front legs or places its hands on the ground to prepare for standing; and *Walk*, where the robot walks using only its two hands.

C. Results

As illustrated in Fig. 1 and Fig. 4, the robots were able to successfully execute the tasks in both simulation and real-world. In simulation, the robot precisely performed the required maneuvers, returning to its original pose without losing stability during the side-roll and flip tasks. As shown

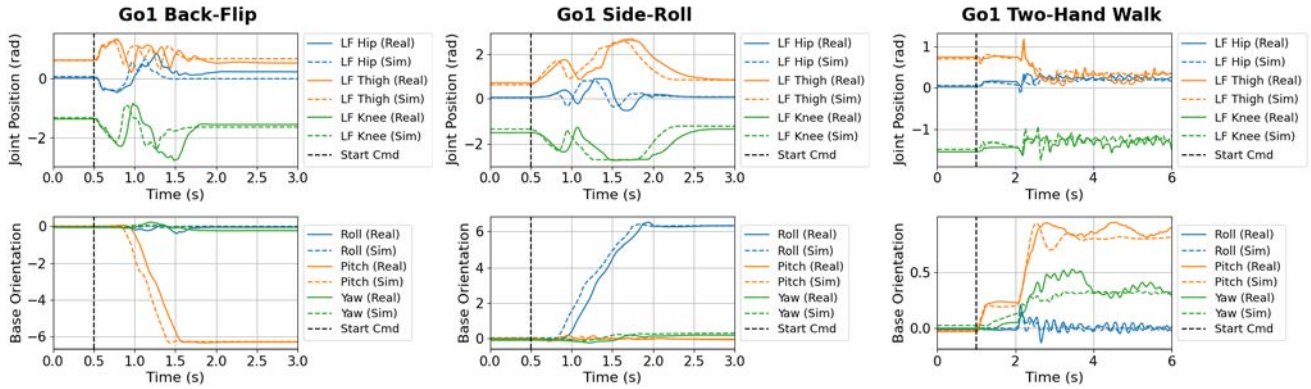


Fig. 6: **Sim-to-real experimental results.** The graph shows the position changes of three joints—hip, thigh, and knee—in the left front leg, along with the body orientation over time for each task. The solid line represents the real-world data, while the dotted line indicates the simulation results.

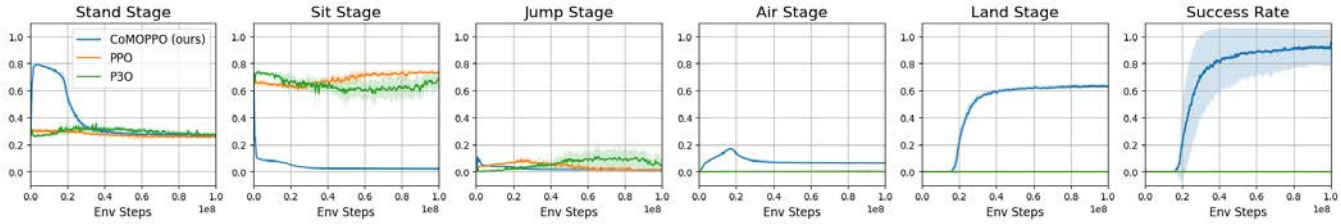


Fig. 7: **Comparison with RL and constrained RL algorithms.** The first five graphs show the proportion of time the robot remains in each stage during an episode, while the last shows the success rate. All results are obtained by training algorithms with five random seeds. Bold lines represent the mean, and shaded areas indicate the standard deviation of the results.

in Fig. 5, it is clearly demonstrated that during the Go1 side-flip task, the robot first lowered its height and rotated to the right side, as indicated by the roll angle. In the H1 back-flip task, the change in the pitch angle and the height indicates that the robot flipped backward. For the H1 two-hand task, the agent exhibited stable balance, consistently alternating its support between the two hands.

Similarly, in the sim-to-real experiments, the robot was able to replicate the learned behaviors with high precision. To provide a more detailed comparison between the simulation and real-world performance, we analyzed the changes in the joint positions of the left front leg and body orientation throughout the tasks. As shown in Fig. 6, the robot successfully jumped backward in both simulation and real-world. In the side-roll task, the change in base orientation followed a similar pattern to the side-flip task, as the robot moved sideways in both tasks. In the two-hand walk, the robot was tilted forward, as shown by the pitch angle, while oscillations in the joint positions imply that the robot was repeatedly lifting and placing its feet to stabilize its balance.

The robot's real-world behavior closely matched that observed in the simulation, displaying similar patterns in joint position and orientation changes. This strong resemblance confirms the success of sim-to-real process, demonstrating that the trained policies were effectively transferred from the simulation to the real environment. The actual movements of the robot during these tasks can be seen in the attached video.

D. Ablation Study

To demonstrate the efficacy of CMORL, we compared the proposed method with existing RL and constrained RL algorithms in the Go1 back-flip task. In the case of RL, we employed PPO [15], where a single reward function is defined by weight-summing the multiple reward and cost functions, and the weights are obtained from the average ratios of objectives and constraints calculated during the training of CoMOPPO. For constrained RL, we utilized penalized PPO (P3O) [16], where a single reward function is defined by summing the multiple rewards with the same weights used in PPO, and the constraints are the same as in CoMOPPO.

As illustrated in Fig. 7, CoMOPPO was the only algorithm that successfully complete the task. In contrast to CoMOPPO, which transitions quickly to the *Air* stage after brief *Sit* and *Jump* stages, the other two algorithms remained primarily in the *Sit* stage, indicating that they were unable to properly execute the jump motion.

VI. CONCLUSIONS

In this work, we have proposed an RL method that defines reward and cost functions in a stage-wise manner within the CMORL framework [14]. Additionally, we have developed a practical CMORL algorithm by expanding PPO [15] to handle multiple objectives and constraints. The proposed method has successfully demonstrated acrobatic tasks in both simulation and real-world settings. Moreover, by comparing the proposed method with existing RL and constrained RL algorithms, we have shown the necessity of the CMORL framework. While in this work, tasks are manually segmented into stages, future research could investigate more efficient techniques for automatically dividing complex tasks into stages.

REFERENCES

[1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020.

[2] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Proceedings of Conference on Robot Learning*, 2023.

[3] C. Zhang, N. Rudin, D. Hoeller, and M. Hutter, "Learning agile locomotion on risky terrains," *arXiv preprint arXiv:2311.10484*, 2023.

[4] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid motor adaptation for legged robots," in *Robotics: Science and Systems*, 2021.

[5] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Legged robots that keep on learning: Fine-tuning locomotion policies in the real world," in *Proceedings of International Conference on Robotics and Automation*, 2022.

[6] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4630–4637, 2022.

[7] K. Caluwaerts, A. Iscen, J. C. Kew, W. Yu, T. Zhang, D. Freeman, K.-H. Lee, L. Lee, S. Saliceti, V. Zhuang, *et al.*, "Barkour: Benchmarking animal-level agility with quadruped robots," *arXiv preprint arXiv:2305.14654*, 2023.

[8] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," in *Proceedings of International Conference on Robotics and Automation*, 2024.

[9] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," in *Robotics: Science and Systems*, 2021.

[10] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Robust and versatile bipedal jumping control through reinforcement learning," in *Robotics: Science and Systems*, 2023.

[11] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "DeepMimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[12] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "ASE: Large-scale reusable adversarial skill embeddings for physically simulated characters," *ACM Transactions On Graphics (TOG)*, vol. 41, no. 4, pp. 1–17, 2022.

[13] R. Grandia, E. Knoop, M. A. Hopkins, G. Wiedebach, J. Bishop, S. Pickles, D. Müller, and M. Bächer, "Design and control of a bipedal robotic character," in *Robotics: Science and Systems*, 2024.

[14] D. Kim, M. Hong, J. Park, and S. Oh, "Scale-invariant gradient aggregation for constrained multi-objective reinforcement learning," *arXiv preprint arXiv:2403.00282*, 2024.

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

[16] L. Zhang, L. Shen, L. Yang, S.-Y. Chen, B. Yuan, X. Wang, and D. Tao, "Penalized proximal policy optimization for safe reinforcement learning," in *Proceedings of International Joint Conference on Artificial Intelligence*, 2022.

[17] P. Kyriakis and J. Deshmukh, "Pareto policy adaptation," in *Proceedings of International Conference on Learning Representations*, 2022.

[18] K. Van Moffaert, M. M. Drugan, and A. Nowé, "Scalarized multi-objective reinforcement learning: Novel design techniques," in *Proceedings of IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2013.

[19] T. Basaklar, S. Gumussoy, and U. Ogras, "PD-MORL: Preference-driven multi-objective reinforcement learning algorithm," in *Proceedings of International Conference on Learning Representations*, 2023.

[20] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik, "Prediction-guided multi-objective reinforcement learning for continuous robot control," in *Proceedings of International Conference on Machine Learning*, 2020.

[21] X.-Q. Cai, P. Zhang, L. Zhao, J. Bian, M. Sugiyama, and A. Llorens, "Distributional pareto-optimal multi-objective reinforcement learning," in *Advances in Neural Information Processing Systems*, 2023.

[22] A. Abdolmaleki, S. Huang, L. Hasenclever, M. Neunert, F. Song, M. Zambelli, M. Martins, N. Heess, R. Hadsell, and M. Riedmiller, "A distributional view on multi-objective policy optimization," in *Proceedings of International Conference on Machine Learning*, 2020.

[23] R. Yang, X. Sun, and K. Narasimhan, "A generalized algorithm for multi-objective reinforcement learning and policy adaptation," *Advances in Neural Information Processing Systems*, 2019.

[24] C. F. Hayes, R. Rădulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, *et al.*, "A practical guide to multi-objective reinforcement learning and planning," *Autonomous Agents and Multi-Agent Systems*, vol. 36, no. 1, p. 26, 2022.

[25] S. Huang, A. Abdolmaleki, G. Vezzani, P. Brakel, D. J. Mankowitz, M. Neunert, S. Bohez, Y. Tassa, N. Heess, M. Riedmiller, *et al.*, "A constrained multi-objective reinforcement learning framework," in *Proceedings of Conference on Robot Learning*, 2022.

[26] A. Stooke, J. Achiam, and P. Abbeel, "Responsive safety in reinforcement learning by pid lagrangian methods," in *Proceedings of International Conference on Machine Learning*, 2020.

[27] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of International Conference on Machine Learning*, 2017.

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

[29] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[30] 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.

[31] 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.

[32] H. Lu, D. Herman, and Y. Yu, "Multi-objective reinforcement learning: Convexity, stationarity and Pareto optimality," in *Proceedings of International Conference on Learning Representations*, 2023.

[33] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of International Conference on Machine Learning*, 2015.

[34] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Proceedings of Conference on robot learning*, 2023.

[35] U. Robotics, "unitree_ros," https://github.com/unitreerobotics/unitree_ros, 2024, gitHub repository.