Learning Smooth Humanoid Locomotion through Lipschitz-Constrained Policies

Zixuan Chen*[1] Xialin He*[2] Yen-Jen Wang*[3] Qiayuan Liao[3] Yanjie Ze[4] Zhongyu Li[3]
S. Shankar Sastry[3] Jiajun Wu[4] Koushil Sreenath[3] Saurabh Gupta[2] Xue Bin Peng[1,5]
[1]Simon Fraser University [2]UIUC [3]UC Berkeley [4]Stanford University [5]NVIDIA *Equal Contribution
lipschitz-constrained-policy.github.io

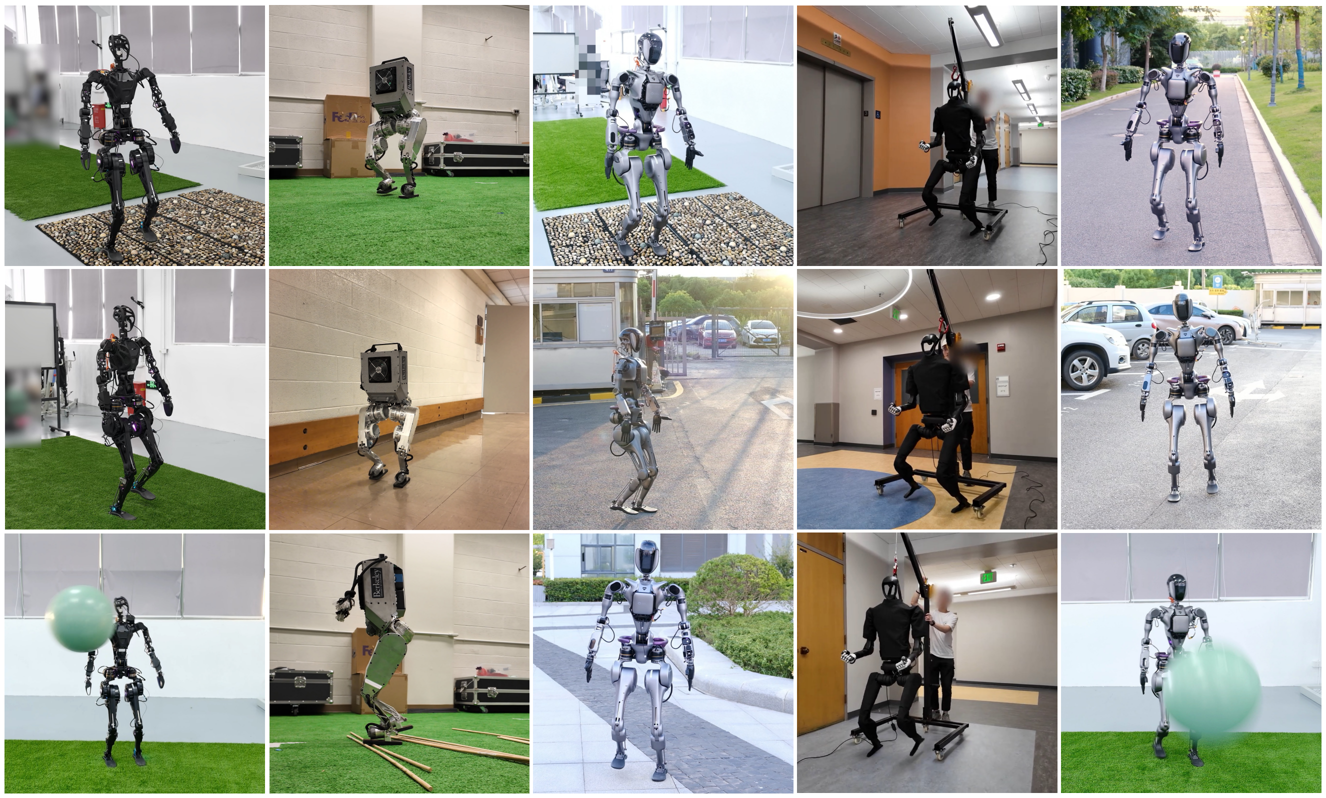

Fig. 1: Lipschitz-constrained policies (LCP) provide a simple and general method for training policies to produce smooth behaviors, which can be directly deployed on a wide range of real-world humanoid robots. Our policies exhibit robust behaviors that can recover from external forces and walk across irregular terrain. For full videos, please visit the project website.

***Abstract*— Reinforcement learning combined with sim-to-real transfer offers a general framework for developing locomotion controllers for legged robots. To facilitate successful deployment in the real world, smoothing techniques, such as low-pass filters and smoothness rewards, are often employed to develop policies with smooth behaviors. However, because these techniques are non-differentiable and usually require tedious tuning of a large set of hyperparameters, they tend to require extensive manual tuning for each robotic platform. To address this challenge and establish a general technique for enforcing smooth behaviors, we propose a simple and effective method that imposes a Lipschitz constraint on a learned policy, which we refer to as Lipschitz-Constrained Policies (LCP). We show that the Lipschitz constraint can be implemented in the form of a gradient penalty, which provides a differentiable objective that can be easily incorporated with automatic differentiation frameworks. We demonstrate that LCP effectively replaces the need for smoothing rewards or low-pass filters and can be easily integrated into training frameworks for many distinct humanoid robots. We extensively evaluate LCP in both simulation and real-world humanoid robots, producing smooth and robust locomotion controllers. All simulation and deployment code, along with complete checkpoints, is available on our project page: https://lipschitz-constrained-policy.github.io.**

I. Introduction

Humanoid research aims to develop intelligent, human-like machines capable of autonomously operating in everyday environments [1]–[4]. One of the most fundamental challenges in this field is achieving reliable mobility. Developing robust locomotion controllers and adapting them to real robots would greatly improve their capabilities.

Traditional model-based methods, such as Model Predictive Control (MPC), necessitate precise system structure and dynamics modeling, which is labor-intensive and challenging to design. In contrast, model-free reinforcement learning provides a straightforward end-to-end approach to developing robust controllers, significantly alleviating the necessity for meticulous dynamics modeling and system design. However, because model-free RL requires a large number of samples through trial-and-error during training, which cannot be performed in the real world, sim-to-real transfer techniques are utilized to enable the successful deployment of controllers in real-world environments. Combined with sim-to-real techniques, model-free RL-based methods have achieved great success in controlling quadruped robots and humanoid robots [5]–[7].

However, due to the simplified dynamics and actuation models used in simulation, the resulting models tend to be nearly idealized, meaning that the motors can produce the desired torques at any state. As a result, RL-based policies trained in simulation are susceptible to developing jittery behaviors akin to bang-bang control [8]. This results in significant differences between actions in consecutive timesteps, leading to excessively high output torques that real actuators cannot produce. As such, these behaviors often fail to transfer to real robots. Therefore, enforcing smooth behaviors is crucial for successful sim-to-real transfer.

Previous systems use smoothing methods to enforce smooth behaviors from a learned policy, such as smoothness rewards or low-pass filters. Incorporating smoothness rewards during training can be an effective approach to eliciting smoother behaviors. In robot locomotion, researchers typically penalize joint velocities, joint accelerations, and energy consumption [9]. Other approaches attempt to smooth policy behavior by applying low-pass filters [10], [11]. However, smoothness rewards require careful tuning of weights to balance smooth behavior with task completion, and low-pass filters often dampen or limit exploration, causing extra effort when training controllers for a new robot. Additionally, the non-differentiable nature of smoothness rewards and low-pass filters presents another limitation.

In this work, we introduce Lipschitz-Constrained Policies (LCP), a general and differentiable method for encouraging RL policies to develop smooth behaviors. LCP enforces a Lipschitz constraint on the output actions of a policy with respect to the input observations through a differentiable gradient penalty. LCP can be implemented with only a few lines of code and easily incorporated into existing RL frameworks. We demonstrate that this approach can be directly applied to train control policies for a diverse suite of humanoid robots. Our experiments show that LCP can be an alternative to non-differentiable smoothness techniques such as smoothness rewards and low-pass filters. We also demonstrate that LCP can be deployed zero-shot to several real-world robots with different morphologies, indicating the generalization of our method.

II. Related Work

Legged robot locomotion has long been a crucial yet challenging problem in robotics due to legged systems' high dimensionality and instability. Classic model-based control methods have achieved impressive behaviors on legged robots [12]–[14]. In recent years, learning-based methods have shown great potential to automate the controller development process, providing a general approach to building robust controllers for quadrupedal locomotion [15]–[18], bipedal locomotion [11], [19]–[21], and humanoid locomotion [7], [22]–[24].

a) Sim-to-Real Transfer: One of the main challenges in RL-based methods is sim-to-real transfer, where policies are first trained in simulation and then deployed in real-world environments. Substantial effort is often necessary to bridge the domain gap between simulations and the real world, such as developing high-fidelity simulators [25], [26], and incorporating domain randomization techniques during training [18], [22], [27], [28]. Another widely adopted approach is the teacher-student framework, where a privileged teacher policy, with access to full state information, is trained first, followed by the training of an observation-based student policy through distillation [15], [20], [22], [24], [29]–[31]. To further facilitate sim-to-real transfer, our framework also leverages a teacher-student framework [6], [16], [23], which trains a latent representation of the dynamics based on the observation history. These methods have been successful in transferring controllers for both quadruped robots [9], [32], [33], and humanoid robots [7], [23]. Some work also explores utilizing a single policy to control robots with different morphologies zero-shot in real world [34]. However, the policy's performance on real humanoid robots has yet to be validated, and it is not easy to plug into any existing training pipeline.

b) Learning Smooth Behaviors: Due to the simplified dynamics of simulators, policies trained in simulation often exhibit jittery behaviors that cannot be transferred to the real world. Therefore, smooth policy behaviors are critical for successful sim-to-real transfer. Common smoothing techniques include the use of smoothness rewards, such as penalizing sudden changes in actions, degree of freedom (DoF) velocities, DoF accelerations [24], [29], [31], [32], [35], [36], and energy consumption [6], [9]. In addition to smoothness rewards, low-pass filters have also been applied to the output actions of a policy to ensure smoother behaviors [10], [11], [18], [37]. However, smoothness rewards typically require careful manual design and tuning, while low-pass filters often dampen policy exploration, resulting in sub-optimal policies. These techniques are also generally not directly differentiable, requiring sample-based gradient estimators to optimize, such policy gradients.

c) Gradient Penalty: In this work, we propose a simple and differentiable method to train RL policies that produce smooth behaviors by leveraging a gradient penalty. Gradient penalty is a common technique for stabilizing training of generative adversarial network (GAN), which is susceptible

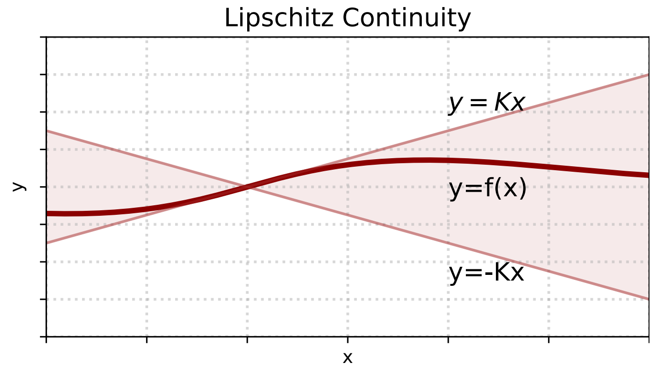

Fig. 2: Lipschitz continuity is a method of quantifying the smoothness functions. A Lipschitz continuous function is a function whose rate-of-change is bounded by a constant K.

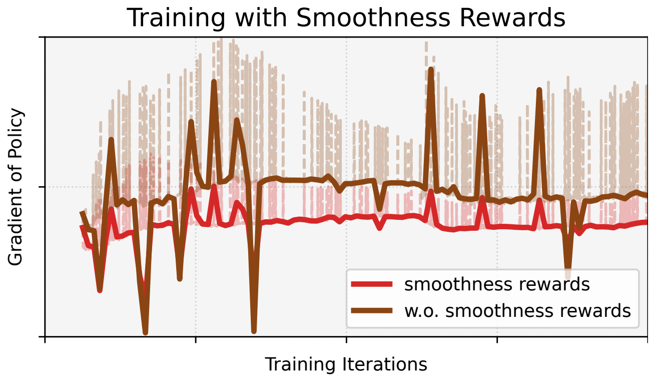

Fig. 3: Gradient of policies trained with and without smoothness rewards. Policies with smoother behaviors also exhibit smaller gradient magnitudes.

to vanishing or exploding gradients. *Arjovsky et al.* [38] proposed the Wasserstein GAN (WGAN) using weight clipping to stabilize training. However, weight clipping still often results in poor model performance and convergence issues. *Gulrajani et al.* [39] introduced the gradient penalty (WGAN-GP) as an alternative to weight clipping, which penalizes the norm of the discriminator's gradient. Since its introduction, the gradient penalty has become a widely used regularization technique for GANs [40], [41]. For motion control, gradient penalty has been an effective technique for improving the stability of adversarial imitation learning. For example, AMP [42], CALM [43], and ASE [44] all apply a gradient penalty to regularize an adversarial discriminator, which then enables a policy to imitate a large variety of challenging motions. While these prior systems demonstrated the effectiveness of gradient penalties as a regularizer for discriminators, in this work, we show that a similar gradient penalty can also be an effective regularizer to encourage policies to produce smooth behaviors, which are then more amenable for real-world transfer.

III. Background

Our method leverages ideas from Lipschitz continuity to train reinforcement learning policies to produce smooth behaviors. This section will review some fundamental concepts for Lipschitz continuity and reinforcement learning to provide a comprehensive background of our proposed method.

A. Lipschitz Continuity

Intuitively, Lipschitz continuity is a property that limits how fast a function can change. This property is a good way of characterizing the smoothness of a function. An intuitive visualization is shown in Fig. 2. Formally, we give the definition of Lipschitz continuity as follows:

Definition III.1 (Lipschitz Continuity)**.** Given two metric spaces (X, d_X) and (Y, d_Y), where d_X denotes the metric on the set X and d_Y is the metric on set Y, a function $f : X \rightarrow Y$ is deemed **Lipschitz continuous** if there exists a real constant K such that, for all $\mathbf{x}_1$ and $\mathbf{x}_2$ in X,

$$d_Y(f(\mathbf{x}_1), f(\mathbf{x}_2)) \leq K d_X(\mathbf{x}_1, \mathbf{x}_2). \quad (1)$$

Any such K is referred to as a **Lipschitz constant** of the function f [45]. A corollary that arises from Lipschitz Continuity is that if the gradient of a function is bounded:

$$\|\nabla_{\mathbf{x}} f(\mathbf{x})\| \leq K, \quad (2)$$

then this function f is Lipschitz continuous. However, it is worth noting that the converse is not true.

B. Reinforcement Learning

In this work, our controllers are trained through reinforcement learning, in which an agent interacts with the environment according to a policy π to maximize an objective function [46]. At each timestep t, the agent observes the state $\mathbf{s}_t$ of the environment, and takes an action $\mathbf{a}_t$ according to the policy $\pi(\mathbf{a}_t \mid \mathbf{s}_t)$. This action then leads to a new state according to the dynamics of the environment $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$. The agent receives a reward $r_t = r(\mathbf{s}_{t+1}, \mathbf{s}_t, \mathbf{a}_t)$ at each step. The agent's goal is to maximize its expected return:

$$J(\pi) = \mathbb{E}_{p(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right], \quad (3)$$

where $p(\tau|\pi)$ represents the likelihood of the trajectory τ, T denotes the time horizon, and γ is the discount factor.

IV. Lipschitz-Constrained Policies

In this section, we introduce Lipschitz-Constrained Policies (LCP), a method for training policies to produce smooth behaviors by incorporating a Lipschitz constraint during training. We begin with a simple experiment to illustrate the motivation behind our method. This is then followed by a detailed description of our proposed method.

A. Motivating Example

We will first illustrate the motivation of LCP with a simple experiment. We know that RL-based policies are prone to producing jittery behaviors, and the most common method for mitigating these behaviors is to incorporate smoothness rewards during training. The smoothness of a function is typically evaluated using the first derivative. Therefore, we compare the ℓ^2-norm of the gradient of policies trained with and without smoothness rewards. Although no specific technique is explicitly applied to regularize the policies' gradient, the gradient trained with smoothness rewards is significantly smaller than that of a policy trained without smoothness rewards, as illustrated in Fig. 3. This fact inspires our proposed method, which explicitly regularizes the gradient of the policy. We show that this simple method leads to smooth behaviors, which can then facilitate successful transfer to the real world.

B. Lipschitz Constraint as a Differentiable Objective

While smoothness rewards can mitigate jittery behaviors, these reward functions can be complex to design, with a large number of hyperparameters that require tuning. Furthermore, these smoothness rewards are non-differentiable since they are implemented as part of the underlying environment. Therefore, they often need to be optimized through sampling-based methods, such as policy gradients. This work proposes a simple and differentiable smoothness objective for policy optimization based on Lipschitz continuity.

Equation 2 stipulates that any function with bounded gradients is Lipschitz continuous. Therefore, we can formulate a constrained policy optimization problem that enforces Lipschitz continuity through a gradient constraint:

$$\begin{aligned} \max_{\pi} \quad & J(\pi) \\ \text{s.t.} \quad & \max_{\mathbf{s},\mathbf{a}} \left[\|\nabla_{\mathbf{s}} \log \pi(\mathbf{a}|\mathbf{s})\|^2 \right] \leq K^2 \end{aligned} \tag{4}$$

where K is a constant and $J(\pi)$ is the RL objective defined in Equation 3. Since calculating the maximum gradient norm across all states is intractable, we approximate this constraint with an expectation over samples collected from policy rollouts, following the heuristic from *Schulman et al.* [47]:

$$\begin{aligned} \max_{\pi} \quad & J(\pi) \\ \text{s.t.} \quad & \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}} \left[\|\nabla_{\mathbf{s}} \log \pi(\mathbf{a}|\mathbf{s})\|^2 \right] \leq K^2, \end{aligned} \tag{5}$$

where $\mathcal{D}$ is a dataset consisting of state-action pairs $(\mathbf{s}_t, \mathbf{a}_t)$ collected from the policy. Next, to facilitate optimization with gradient-based methods, the constraint can be reformulated into a penalty by introducing a Lagrange multiplier λ:

$$\min_{\lambda \geq 0} \max_{\pi} \quad J(\pi) - \lambda \left(\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}} \left[\|\nabla_{\mathbf{s}} \log \pi(\mathbf{a}|\mathbf{s})\|^2 \right] - K^2 \right). \tag{6}$$

To further simplify the objective, we set λ_{gp} as a manually specified coefficient, and since K is a constant, this leads to a simple differentiable gradient penalty (GP) on the policy:

$$\max_{\pi} \quad J(\pi) - \lambda_{\text{gp}} \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}} \left[\|\nabla_{\mathbf{s}} \log \pi(\mathbf{a}|\mathbf{s})\|^2 \right]. \tag{7}$$

This gradient penalty can be easily implemented in any reinforcement learning framework, requiring only a few lines of code. The gradient penalty provides a simple and differentiable alternative to smoothness rewards or low-pass filters, which are not differentiable with respect to the policy parameters. Our experiments show that LCP provides an effective alternative to non-differentiable smoothing techniques and can be directly used to train robust locomotion controllers for a diverse cast of robots.

V. Training Setup

To evaluate the effectiveness of our method, we apply LCP to train policies for a variety of humanoid robots, where the task is for the robots to walk while following steering commands.

a) Observations: The input observations to the policy $\mathbf{o}_t = [\boldsymbol{\phi}_t, \mathbf{c}_t, \mathbf{s}_t^{\text{robot}}, \mathbf{a}_{t-1}]$ consists of a gait phase variable $\boldsymbol{\phi}_t \in \mathbb{R}^2$ (a periodic clock signal represented by its sine and cosine components), command $\mathbf{c}_t$, measured joint positions and velocities $\mathbf{s}_t^{\text{robot}}$, and the previous output action of the policy $\mathbf{a}_{t-1}$. To enable robust sim-to-real transfer, the policy also takes privileged information $\mathbf{e}_t$ as input, which consists of the base mass, center of mass, motor strengths, and root linear velocity. Observations $\mathbf{o}_t$ are normalized with a running mean and standard deviation before being passed as input to the policy.

b) Commands: The command input to the policy $\mathbf{c}_t = [\text{v}_\text{x}^\text{cmd}, \text{v}_\text{y}^\text{cmd}, \text{v}_\text{yaw}^\text{cmd}]$ consists of the desired linear velocities along x-axis $\text{v}_\text{x}^\text{cmd} \in [0\text{m/s}, 0.8\text{m/s}]$ and y-axis $\text{v}_\text{y}^\text{cmd} \in [-0.4\text{m/s}, 0.4\text{m/s}]$, and the desired yaw velocity $\text{v}_\text{yaw}^\text{cmd} \in [-0.6\text{rad/s}, 0.6\text{rad/s}]$, both are in the robot frame. During training, commands are randomly sampled from their respective ranges every 150 timestep or when the environment is reset.

c) Actions: The policy's output actions specify target joint rotations for all joints in the robot's body, which are then converted to torque commands by PD controllers with manually specified PD gains.

d) Training: All policies are modeled using neural networks and trained using the PPO algorithm [48]. The policies are trained solely in simulation with domain randomization and then deployed directly on the real robots [27]. Sim-to-real transfer is performed using Regularized Online Adaptation (ROA) [6], [32].

VI. Experiments

LCP's effectiveness is evaluated on a set of diverse humanoid robots to show its generalization ability. We conduct an extensive suite of simulation and real-world experiments, comparing LCP to commonly used smoothing techniques from prior systems.

A. Robot Platforms

We evaluate our framework on three real-world robots: the human-sized Fourier GR1T1, Fourier GR1T2, Unitree H1, and the smaller Berkeley Humanoid. We will first provide an overview of each robot's body structure. Then, our experiments show that LCP is a general smoothing technique that can be applied widely to several distinct robots.

a) Fourier GR1T1 & Fourier GR1T2: The Fourier GR1T1 and Fourier GR1T2 have the same mechanical structure. They both comprise 21 joints, with 12 joints in the lower body and 9 in the upper body. Notice that the torque limit for the *ankle roll* joint is minimal; we treat this as a passive joint during training and deploying. This means we control 19 joints of GR1 in total.

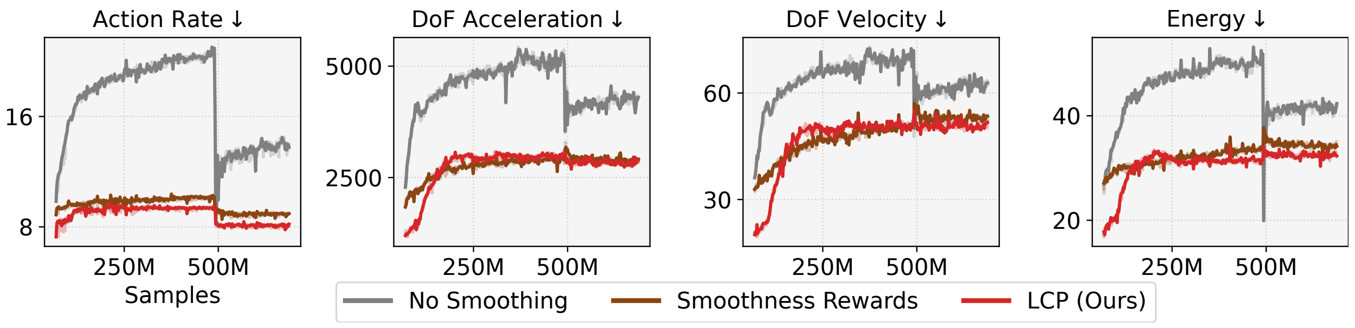

Fig. 4: Smoothness metrics recorded over the course of training. LCP produces smooth behaviors that are comparable to policies that are trained with explicit smoothness rewards.

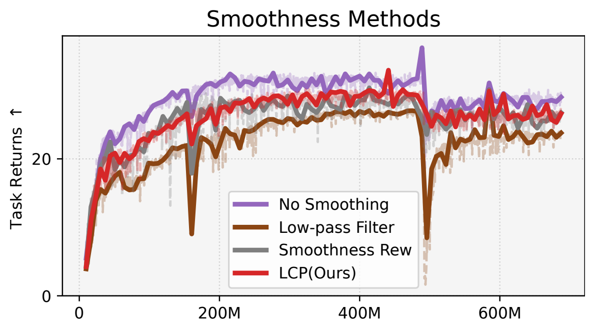

Fig. 5: Task returns of different smoothing methods. LCP provides an effective alternative to other techniques.

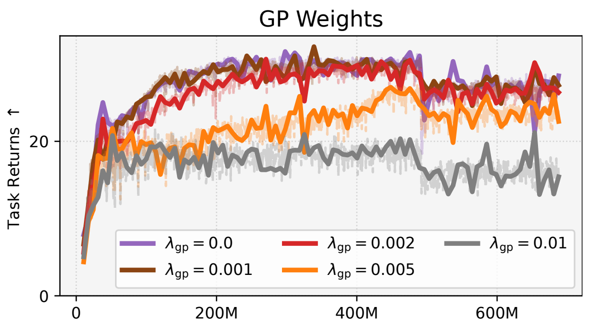

Fig. 6: Task returns of LCP with different λ_{gp}. Excessively large λ_{gp} may hinder policy learning.

b) Unitree H1: The Unitree H1 has 19 joints, with 10 joints in the lower body, 9 in the upper body, and 1 ankle joint per leg. All joints are actively controlled.

c) Berkeley Humanoid: Berkeley Humanoid is a small robot with a height of 0.85m [49]. It has 12 degrees of freedom, with 6 joints in each leg and 2 joints in each ankle.

B. Results

To evaluate the effectiveness of LCP, we compare our method to the following baselines:

- **No smoothing:** No smoothing techniques are applied during training. This baseline demonstrates the necessity of smoothing techniques for sim-to-real transfer;
- **Smoothness rewards:** Smoothness rewards are the most commonly used smoothing method, where additional reward terms are incorporated into the reward function to encourage smooth behaviors. These reward functions are not directly differentiable.
- **Low-pass Filters:** Low-pass filters are commonly used for action smoothing, where a filter is applied to the policy's output actions before the action is applied to the environment. Low-pass filters are also not easily differentiable for policy training.

To evaluate the effectiveness of various smoothing techniques, we record a suite of smoothness metrics, including mean DoF velocities (rad/s), mean energy (N·rad/s), action rate (rad/s), robot base acceleration (m/s^2), action jitter (rad/s^3), and DoF position jitter (rad/s^3). Action rate is the first derivative of output actions over time. The jitter metrics represent the third derivative of their respective quantities [50]. We also record the mean task return, which is calculated using the linear and angular velocity tracking rewards.

a) Is LCP effective for producing smooth behaviors?: We train policies with LCP using a GP coefficient of $\lambda_{\text{gp}} = 0.002$. We track various smoothness metrics throughout the training process, including energy consumption, degrees-of-freedom (DoF) velocities, DoF accelerations, and action rates. We then compare these metrics against policies trained with and without smoothness rewards. The results are recorded in Fig. 4. While LCP is not trained with reward functions that directly minimize these smoothness metrics, LCP nonetheless produces smooth behaviors that are similar to policies trained with smoothness rewards. This demonstrates that LCP can be an effective substitute for traditional smoothness rewards.

b) How does LCP affect task performance?: In TABLE I(a) and Fig. 5, we compare the task performance of LCP with policies trained with other smoothing methods. LCP achieves similar task performance compared to policies trained solely with smoothness rewards. Policies trained with low-pass filters tend to exhibit lower task returns, which may be due to the damping introduced by low-pass filters that can in turn impair exploration. Policies trained without smoothing techniques tend to achieve the highest task returns but exhibit highly jittery behaviors unsuitable for real-world deployment.

c) What is the effect of the GP coefficient λ_{gp}?: TABLE I(b) shows the performance of LCP with different GP coefficients λ_{gp}. Incorporating a gradient penalty leads to significantly smoother behaviors. However, with small

TABLE I: **Ablation Studies.** All policies are trained with three random seeds and tested in 1000 environments for 500 steps, corresponding to 10 seconds clock time.

Method	Action Jitter ↓	DoF Pos Jitter ↓	DoF Velocity ↓	Energy ↓	Base Acc ↓	Task Return ↑
(a) Ablation on Smooth Methods						
LCP (ours)	$\mathbf{3.21 \pm 0.11}$	$\mathbf{0.17 \pm 0.01}$	$\mathbf{10.65 \pm 0.37}$	$\mathbf{24.57 \pm 1.17}$	$\mathbf{0.06 \pm 0.002}$	26.03 ± 1.51
Smoothness Reward	5.74 ± 0.08	0.19 ± 0.002	11.35 ± 0.51	25.92 ± 0.84	$\mathbf{0.06 \pm 0.002}$	26.56 ± 0.26
Low-pass Filter	7.86 ± 3.00	0.23 ± 0.04	11.72 ± 0.14	32.83 ± 5.50	$\mathbf{0.06 \pm 0.002}$	24.98 ± 1.29
No Smoothness	42.19 ± 4.72	0.41 ± 0.08	12.92 ± 0.99	42.68 ± 10.27	0.09 ± 0.01	$\mathbf{28.87 \pm 0.85}$
(b) Ablation on GP Weights (λ_{gp})						
LCP w. $\lambda_{gp} = 0.0$	42.19 ± 4.72	0.41 ± 0.08	12.92 ± 0.99	42.68 ± 10.27	0.09 ± 0.01	$\mathbf{28.87 \pm 0.85}$
LCP w. $\lambda_{gp} = 0.001$	3.69 ± 0.31	0.21 ± 0.05	11.44 ± 1.18	27.09 ± 4.44	0.06 ± 0.01	26.32 ± 1.20
LCP w. $\lambda_{gp} = 0.002$ (ours)	3.21 ± 0.11	0.17 ± 0.01	10.65 ± 0.37	24.57 ± 1.17	0.06 ± 0.002	26.03 ± 1.51
LCP w. $\lambda_{gp} = 0.005$	2.10 ± 0.05	0.15 ± 0.01	10.44 ± 0.70	26.24 ± 3.50	0.05 ± 0.002	23.92 ± 2.05
LCP w. $\lambda_{gp} = 0.01$	$\mathbf{0.17 \pm 0.01}$	$\mathbf{0.07 \pm 0.00}$	$\mathbf{2.75 \pm 0.12}$	$\mathbf{5.89 \pm 0.28}$	$\mathbf{0.007 \pm 0.00}$	16.11 ± 2.76
(c) Ablation on GP Inputs						
LCP w. GP on whole obs (ours)	$\mathbf{3.21 \pm 0.11}$	$\mathbf{0.17 \pm 0.01}$	$\mathbf{10.65 \pm 0.37}$	$\mathbf{24.57 \pm 1.17}$	$\mathbf{0.06 \pm 0.002}$	$\mathbf{26.03 \pm 1.51}$
LCP w. GP on current obs	7.16 ± 0.60	0.35 ± 0.03	13.70 ± 1.50	35.18 ± 4.84	0.09 ± 0.005	25.44 ± 3.73

TABLE II: Sim-to-sim perfomance when transferring policies trained in IsaacGym to Mujoco. All policies are trained with three random seeds and tested for 3 trials with 500 steps, corresponding to 10 seconds per trial.

	Action Jitter ↓	DoF Pos Jitter ↓	DoF Velocity ↓	Energy ↓	Base Acc ↓	Task Return ↑
Fourier GR1	1.47 ± 0.43	0.34 ± 0.07	9.54 ± 1.53	36.38 ± 2.97	0.08 ± 0.004	24.33 ± 1.25
Unitree H1	0.44 ± 0.03	0.10 ± 0.007	9.12 ± 0.38	76.22 ± 5.81	0.04 ± 0.005	21.74 ± 1.40
Berkeley Humanoid	1.77 ± 0.32	0.12 ± 0.01	7.92 ± 0.21	19.99 ± 0.36	0.06 ± 0.00	26.50 ± 0.57

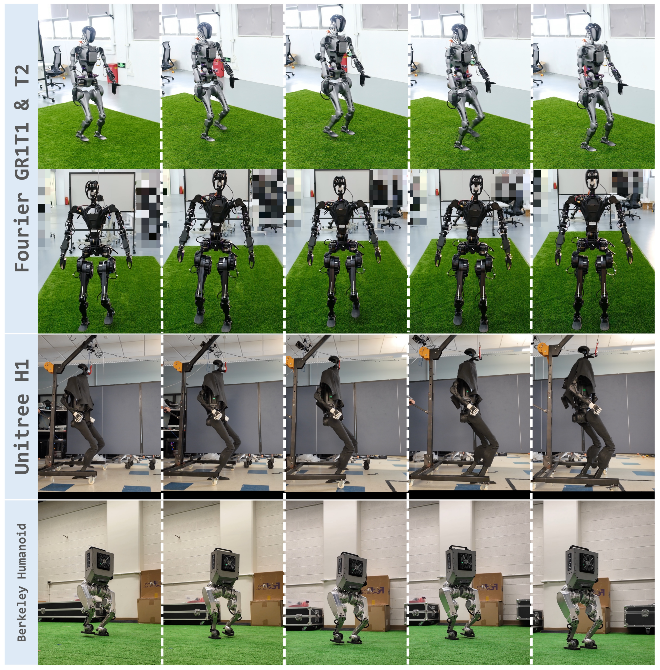

Fig. 7: Real-world deployment. LCP is able to train effective locomotion policies on a wide range of robots, which can be directly transferred to the real world.

coefficients (e.g., $\lambda_{gp} = 0.001$), the policy can develop jittery behaviors that are dangerous to deploy in the real world. However, excessively larger coefficients (e.g., $\lambda_{gp} = 0.01$) lead to a substantial decline in task return due to overly smooth and sluggish behaviors. As shown in Fig. 6, large values of λ_{gp} may also lead to slower learning speeds. Our experiments suggest that $\lambda_{gp} = 0.002$ strikes an effective balance between policy smoothness and task performance. As with other smoothing techniques, some care is required to tune the GP coefficient for a better performance.

d) Which components of the observation should GP be applied to?: Since the policies are trained using the ROA for sim-to-real transfer, the policy's input consists of the current observation and a history of past observations. TABLE I(c) compares the performance of LCP when the gradient penalty is applied to the whole input observation or only to the current observation. We find that applying GP on the whole observation achieves the best performance. Regularizing the policy only with respect to the current observation can still lead to non-smooth behaviors due to changes in the history.

e) Sim-to-Sim Transfer: Before deploying models we test our models in a different simulator Mujoco [25]. As shown in TABLE II, we observe a slight decrease in task return compared to IsaacGym for full-sized robots such as Fourier GR1 and Unitree H1, suggesting that the domain gap is more significant for larger robots. The overall results show that LCP performs well in sim-to-sim transfer, providing confidence for subsequent real-world deployments.

C. Real World Deployment

We deploy LCP models trained with the same reward functions and $\lambda_{gp} = 0.002$ on four distinct robots. As shown in Fig. 1, LCP effectively enables different robots to walk in the real world. Fig. 7 shows snapshots of the robots' behaviors over the course of one gait cycle.

Terrains To evaluate the robustness of the learned policies, we apply the policies in the real world to walk on three types of terrain: smooth, soft, and rough plane. We measure the jitter metrics to evaluate LCP's performance, as shown in TABLE III. The behaviors of our policies remain smooth

TABLE III: Performance during real-world deployment. Performance for each method is calculated across 3 models from different training runs. Each model is executed for 10 seconds. Standard deviation is recorded for each test.

Robot	Action Jitter ↓	DoF Pos Jitter ↓	DoF Velocity ↓
(a) Smooth Plane			
Fourier GR1	1.12 ± 0.16	0.28 ± 0.13	10.82 ± 1.58
Unitree H1	1.11 ± 0.07	0.14 ± 0.01	10.95 ± 0.53
Berkeley Humanoid	1.56 ± 0.10	0.10 ± 0.01	4.99 ± 0.60
(b) Soft Plane			
Fourier GR1	1.18 ± 0.17	0.24 ± 0.09	10.45 ± 1.42
Unitree H1	1.18 ± 0.09	0.15 ± 0.01	11.80 ± 0.57
Berkeley Humanoid	1.66 ± 0.03	0.12 ± 0.01	6.78 ± 1.57
(c) Rough Plane			
Fourier GR1	1.18 ± 0.22	0.26 ± 0.11	11.61 ± 1.64
Unitree H1	1.20 ± 0.09	0.14 ± 0.01	11.68 ± 0.84
Berkeley Humanoid	1.63 ± 0.11	0.11 ± 0.01	5.02 ± 0.48

in the presence of variations in the terrain and is able to effective traverse the different surfaces.

External Forces To further test the robustness of our policies, we apply external forces to the robot in the real world Fig. 1. The recovery behaviors of our models are shown in the supplementary video. The LCP models can robustly recover from unexpected external perturbations.

VII. Conclusion

In this work, we present Lipschitz Constrained Policies (LCP), a simple and general method for training controllers to produce smooth behaviors amenable to sim-to-real transfer. LCP approximates a Lipschitz constraint on the policy, implemented in the form of a differentiable gradient penalty applied during training. Through extensive simulation and real-world experiments, we show the effectiveness of LCP in training locomotion controllers for a wide range of real humanoid robots. While LCP has demonstrated its effectiveness in real-world locomotion experiments, our results are still limited to basic walking behaviors. Evaluating LCP on more dynamic skills, such as running and jumping, would help further validate this method's generality.

Acknowledgement

We extend our gratitude to Fourier Intelligence for their hardware support, Jiaze Cai, Yiyang Shao, Junfeng Long, Haoru Xue, and Renzhi Tao for their assistance with real-world experiments, and Mintae Kim for his valuable advice on the mathematical formulas. This work was supported in part by an NSERC Discovery Grant (DGECR-2023-00280).

References

[1] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang, "Open-television: Teleoperation with immersive active visual feedback," *arXiv preprint arXiv:2407.01512*, 2024.

[2] Y. Ze, Z. Chen, W. Wang, T. Chen, X. He, Y. Yuan, X. B. Peng, and J. Wu, "Generalizable humanoid manipulation with improved 3d diffusion policies," *arXiv preprint arXiv:2410.10803*, 2024.

[3] I. Radosavovic, S. Kamat, T. Darrell, and J. Malik, "Learning humanoid locomotion over challenging terrain," 2024. [Online]. Available: https://arxiv.org/abs/2410.03654

[4] F. Liu, Z. Gu, Y. Cai, Z. Zhou, S. Zhao, H. Jung, S. Ha, Y. Chen, D. Xu, and Y. Zhao, "Opt2skill: Imitating dynamically-feasible whole-body trajectories for versatile humanoid loco-manipulation," *arXiv preprint arXiv:2409.20514*, 2024.

[5] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.

[6] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: learning a unified policy for manipulation and locomotion," in *Conference on Robot Learning*. PMLR, 2023, pp. 138–149.

[7] X. Cheng, Y. Ji, J. Chen, R. Yang, G. Yang, and X. Wang, "Expressive whole-body control for humanoid robots," *arXiv preprint arXiv:2402.16796*, 2024.

[8] J. Lasalle, "The 'bang-bang' principle," *IFAC Proceedings Volumes*, vol. 1, no. 1, pp. 503–507, 1960, 1st International IFAC Congress on Automatic and Remote Control, Moscow, USSR, 1960. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S147466701770095X

[9] Z. Fu, A. Kumar, J. Malik, and D. Pathak, "Minimizing energy consumption leads to the emergence of gaits in legged robots," in *Conference on Robot Learning (CoRL)*, 2021.

[10] Y. Ji, Z. Li, Y. Sun, X. B. Peng, S. Levine, G. Berseth, and K. Sreenath, "Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1479–1486.

[11] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2811–2817.

[12] H. Miura and I. Shimoyama, "Dynamic walk of a biped," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 60–74, 1984.

[13] K. Sreenath, H.-W. Park, I. Poulakakis, and J. W. Grizzle, "A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on mabel," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1170–1193, 2011.

[14] H. Geyer, A. Seyfarth, and R. Blickhan, "Positive force feedback in bouncing gaits?" *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 270, no. 1529, pp. 2173–2183, 2003.

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

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

[17] H. Lai, W. Zhang, X. He, C. Yu, Z. Tian, Y. Yu, and J. Wang, "Sim-to-real transfer for quadrupedal locomotion via terrain transformer," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5141–5147.

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

[19] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Robust and versatile bipedal jumping control through reinforcement learning," *arXiv preprint arXiv:2302.09450*, 2023.

[20] A. Kumar, Z. Li, J. Zeng, D. Pathak, K. Sreenath, and J. Malik, "Adapting rapid motor adaptation for bipedal robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1161–1168.

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

[22] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi9579, 2024.

[23] X. Gu, Y.-J. Wang, X. Zhu, C. Shi, Y. Guo, Y. Liu, and J. Chen, "Advancing humanoid locomotion: Mastering challenging terrains with denoising world model learning," *arXiv preprint arXiv:2408.14472*, 2024.

[24] T. He, Z. Luo, X. He, W. Xiao, C. Zhang, W. Zhang, K. Kitani, C. Liu, and G. Shi, "Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning," *arXiv preprint arXiv:2406.08858*, 2024.

[25] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[26] 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," *arXiv preprint arXiv:2108.10470*, 2021.

[27] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1–8.

[28] I. Radosavovic, B. Zhang, B. Shi, J. Rajasegaran, S. Kamat, T. Darrell, K. Sreenath, and J. Malik, "Humanoid locomotion as next token prediction," *arXiv preprint arXiv:2402.19469*, 2024.

[29] X. Gu, Y.-J. Wang, and J. Chen, "Humanoid-gym: Reinforcement learning for humanoid robot with zero-shot sim2real transfer," *arXiv preprint arXiv:2404.05695*, 2024.

[30] Z. Zhuang, S. Yao, and H. Zhao, "Humanoid parkour learning," *arXiv preprint arXiv:2406.10759*, 2024.

[31] Z. Fu, Q. Zhao, Q. Wu, G. Wetzstein, and C. Finn, "Humanplus: Humanoid shadowing and imitation from humans," in *arXiv*, 2024.

[32] M. Liu, Z. Chen, X. Cheng, Y. Ji, R. Yang, and X. Wang, "Visual whole-body control for legged loco-manipulation," *arXiv preprint arXiv:2403.16967*, 2024.

[33] X. Cheng, A. Kumar, and D. Pathak, "Legs as manipulator: Pushing quadrupedal agility beyond locomotion," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5106–5112.

[34] N. Bohlinger, G. Czechmanowski, M. Krupka, P. Kicki, K. Walas, J. Peters, and D. Tateo, "One policy to run them all: an end-to-end learning approach to multi-embodiment locomotion," *arXiv preprint arXiv:2409.06366*, 2024.

[35] T. He, Z. Luo, W. Xiao, C. Zhang, K. Kitani, C. Liu, and G. Shi, "Learning human-to-humanoid real-time whole-body teleoperation," *arXiv preprint arXiv:2403.04436*, 2024.

[36] C. Zhang, W. Xiao, T. He, and G. Shi, "Wococo: Learning whole-body humanoid control with sequential contacts," *arXiv preprint arXiv:2406.06005*, 2024.

[37] G. Feng, H. Zhang, Z. Li, X. B. Peng, B. Basireddy, L. Yue, Z. Song, L. Yang, Y. Liu, K. Sreenath, *et al.*, "Genloco: Generalized locomotion controllers for quadrupedal robots," in *Conference on Robot Learning*. PMLR, 2023, pp. 1893–1903.

[38] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017. [Online]. Available: https://arxiv.org/abs/1701.07875

[39] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," 2017. [Online]. Available: https://arxiv.org/abs/1704.00028

[40] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," 2018. [Online]. Available: https://arxiv.org/abs/1710.10196

[41] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," 2019. [Online]. Available: https://arxiv.org/abs/1809.11096

[42] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics*, vol. 40, no. 4, p. 1–20, July 2021. [Online]. Available: http://dx.doi.org/10.1145/3450626.3459670

[43] C. Tessler, Y. Kasten, Y. Guo, S. Mannor, G. Chechik, and X. B. Peng, "Calm: Conditional adversarial latent models for directable virtual characters," in *ACM SIGGRAPH 2023 Conference Proceedings*, ser. SIGGRAPH '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3588432.3591541

[44] 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*, vol. 41, no. 4, p. 1–17, July 2022. [Online]. Available: http://dx.doi.org/10.1145/3528223.3530110

[45] M. O'Searcoid, *Metric spaces*. Springer Science & Business Media, 2006.

[46] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[47] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015. [Online]. Available: http://arxiv.org/abs/1502.05477

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

[49] Q. Liao, B. Zhang, X. Huang, X. Huang, Z. Li, and K. Sreenath, "Berkeley humanoid: A research platform for learning-based control," *arXiv preprint arXiv:2407.21781*, 2024.

[50] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.