Fast and Robust Terrain-Adaptive Quadruped Locomotion via GPU-Accelerated Simulation and Sim2Real Transfer

Bryan Kyunghun Kim
University of Southern California
bryankki@usc.edu

Tyler Rotello
University of Southern California
rotello@usc.edu

Changhui Hou
University of Southern California
changhui@usc.edu

Abstract

In the rapidly advancing field of robotics, achieving reliable real-world deployment of behaviors learned in simulation—known as Sim2Real transfer—remains a significant challenge. In this work, we address both the issue of robustness across diverse terrain and the need for faster, more effective Sim2Real transfer. Building on joystick-driven locomotion policies originally trained in flat, controlled environments, we extend capability to more complex terrains such as icy surfaces and uneven ground. Using the Unitree Go1 robot in MuJoCo simulation, we modify environment parameters to replicate real-world challenges and train terrain-specific policy networks that learn the joint-level control necessary for stable linear and angular movement. By leveraging GPU-accelerated simulation engines, we massively parallelize training, enabling fast, cost-effective policy development. Our approach results in robust, terrain-adaptive policies suitable for real-world deployment. As a final step, we demonstrate physical transfer by attempting to deploy the learned policies onto the Unitree Go2 platform.

1 Introduction

Quadrupedal robots hold significant potential for real-world applications such as disaster response, autonomous delivery, and planetary exploration. However, achieving robust deployment in unstructured environments remains a major challenge. Two fundamental obstacles persist: (1) locomotion policies trained in simulation often fail to generalize to the complexities of the real world, and (2) physical training is resource-intensive, both in time and hardware wear.

Simulation-based reinforcement learning (RL) offers a scalable alternative, enabling safe and repeatable training of locomotion policies. Yet, the reality gap—the mismatch between simulated dynamics and real-world physics—limits the success of direct policy transfer. Addressing this challenge requires both improving simulation fidelity and enhancing the robustness of learned policies (Tan et al. [2018]).

One widely adopted approach is domain randomization, which introduces randomized environmental parameters—such as friction, mass, or surface properties—into training. The idea is to expose the agent to "such a wide distribution of training conditions that the real world appears as just another sample from that distribution" (Tobin et al. [2017]). Although initially supported through empirical success, domain randomization has since been formalized with theoretical analysis. For example, it has been shown that "sim-to-real transfer can succeed under mild conditions without any real-world

training samples" and that policies which use memory (i.e., history) can further close the gap (Chen et al. [2021]).

To accelerate training and support large-scale environment variation, we construct a GPU-accelerated simulation and learning pipeline centered on Proximal Policy Optimization (PPO), a reinforcement learning algorithm that offers "a favorable balance between sample complexity, simplicity, and wall-time" (Schulman et al. [2017]). This pipeline allows us to train locomotion policies in minutes using high-throughput rollouts and rapidly evaluate transfer to the real world.

Our methodology comprises three key innovations:

1. **Aggressive domain randomization.** By stochastically varying terrain features—such as friction, slope, and surface topology—across training episodes, we expose the agent to a broad distribution of scenarios. This promotes invariant control primitives rather than overfitting to narrow conditions.
2. **Physics-faithful contact modeling.** We augment domain randomization with precise simulation of contact dynamics in MuJoCo, ensuring that learned policies maintain stable joint-level control even under low-friction or deformable interactions.
3. **Modular sim-to-real workflow.** We establish a checkpointing and evaluation framework that automates transfer to hardware with minimal parameter adjustment, facilitating deployment on the Unitree Go2 platform.

Preliminary results show that our terrain-specific policies, trained across icy, rocky, and low-friction environments, maintain stable gaits and significantly outperform flatland-trained baselines. These findings support the hypothesis that structured domain variability and high-fidelity simulation, coupled with sample-efficient policy optimization, can effectively bridge the gap between simulation and reality for agile legged locomotion.

2 Related Works

2.1 MuJoCo Playground

In this project, DeepMind researchers present a new simulation suite that builds on the MuJoCo physics engine using the JAX framework. The key advantage of MuJoCo Playground (Zakka et al. [2025]) lies in its native support for GPU-accelerated physics, enabling significantly faster training compared to traditional CPU-based setups. The paper benchmarks a set of classic locomotion tasks and demonstrates that GPU acceleration not only reduces training time but also allows for larger batch sizes and more complex agents. The study highlights the inefficiencies of current parallelization strategies and the importance of tailoring training code to maximize GPU utilization. It also enables a fast and simplified sim-to-real deployment process, which allows developers to tweak a few parameters and test it on a physical robot in little time. This work directly supports our decision to use MuJoCo Playground as our primary simulation environment, especially for training energy-efficient gaits and high-dimensional policies for quadruped robots like the Unitree Go1/Go2.

2.2 GPU-Accelerated Robotics Training

Kinetix(Matthews et al. [2025]) explores a research direction in which generalist agents are trained across a wide range of tasks using highly parallelizable and JAX-based environments. The paper introduces an open-ended setup where agents are exposed to a massive number of control environments—each with unique dynamics and challenges—to encourage generalization and long-term skill acquisition.

The main takeaway from this paper is that Kinetix utilizes JAX to simulate large batches of diverse environments and becomes an open-ended 2-dimensional space that can be used to train general agents. The benefits of putting a model through a large number of different environments can help improve its robustness and furthermore assist its deployment to the real world. While we are not aiming to train a generalist agent per se, the key insight from Kinetix is its use of parallel task diversity to promote robustness and policy generalization. This idea translates into our plan to implement domain randomization—introducing varied terrains, physical parameters, and goal locations—to

simulate a wide task distribution that prepares the Unitree models for real-world variability. Additionally, the use of JAX across both papers justifies our use of MuJoCo Playground as a compatible backend.

2.3 Importance of Domain Randomization

The work done by Yang and the MIT CSAIL group (Yu et al. [2024]) explores the use of large language models (LLMs) and text-to-image diffusion models to create synthetic visual training data for reinforcement learning. The key innovation is the generation of realistic yet diverse images of parkour-style environments, from which an agent learns a visual policy. The training pipeline involves:

1. **Using ChatGPT** to describe challenging visual scenes or goals.
2. **Generating corresponding images** with LucidSim.
3. **Training a visual control policy** directly on this generated data using a vision transformer.

The work done by this group further emphasizes the importance of domain randomization in pursuit of bridging the gap between simulation and real-world movement. The interesting takeaway from this paper, however, is that LLM-generated image environments resulted in policies that generalize better to unseen conditions compared to standard domain randomization, which is the method both Playground and we use when training quadruped models. This opens the door for a potential enhancement to our current policy training in that the model may better generalize to unseen environments and be able to move around in different terrain.

This paper exemplifies our original stretch goal: using LLMs to generate novel training environments or modify simulation parameters. While our current efforts are rooted in flatland locomotion without vision, this research provides a blueprint for future extensions—such as terrain generation or natural language-conditioned tasks (e.g., "walk to the red cube on the hill"). The insight that LLM-generated content may outperform traditional randomization highlights a new frontier in simulation robustness, which we plan to revisit after deploying base locomotion.

3 Methodology

Our project begins with a clear task definition: enabling a quadrupedal robot to achieve stable and adaptable locomotion across a variety of terrains. This involves training the robot to move not just on flat, controlled surfaces, but also on more challenging conditions such as icy ground, rough terrain, and uneven stairs. To simulate these environments, we used the MuJoCo physics engine and leveraged its XML modeling system to define terrain properties—modifying parameters such as friction coefficients, heightfields, and material reflectance to reflect diverse real-world conditions. These XML files allowed us to procedurally generate diverse and realistic training environments, forming the foundation for domain randomization. We then trained policy networks using Proximal Policy Optimization (PPO) to control the robot's joints in a way that maintains stability, energy efficiency, and forward motion in the desired direction, commonly referred to as a 'joystick policy'. Throughout training, we tuned hyperparameters like number of epochs and rollout length while swapping out evaluation environments to ensure efficient learning and generalization across terrains while allowing us to compare our policies with the baseline ones. To preserve progress and enable transfer, we periodically save model checkpoints. These checkpoints can be evaluated in different simulated environments via rollout and, if successful, directly deployed onto a physical robot platform such as the Unitree Go2, which is the robot our group has access to. This Sim2Real pipeline allows us to develop robust locomotion policies in simulation that are ready for real-world testing with minimal additional tuning.

3.1 Simulation Framework and Stack

This project builds upon the foundations laid by MuJoCo and MuJoCo Playground, leveraging existing technologies to significantly accelerate the training process for quadrupedal locomotion. Our overarching goal is to enhance and extend state-of-the-art Sim2Real methods through careful experimentation and tool selection.

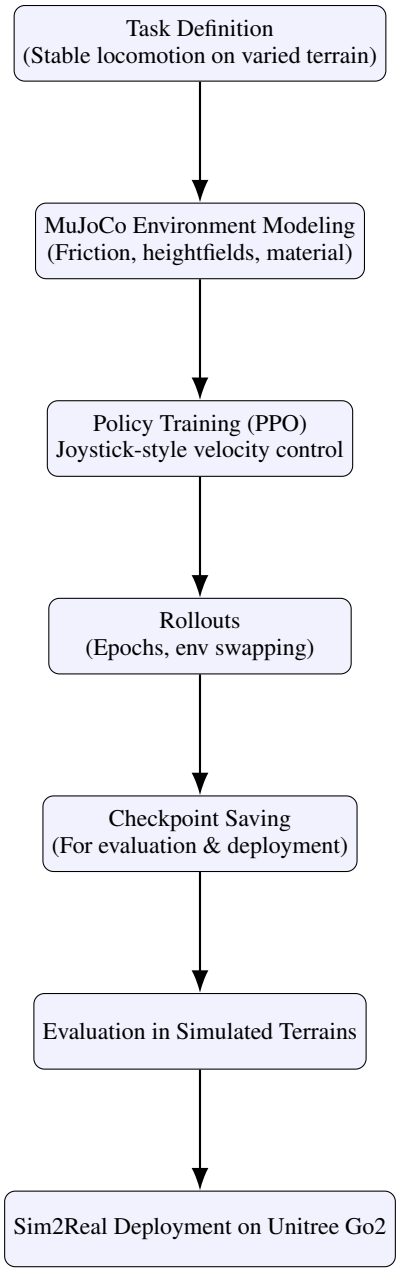

Figure 1: Overview of Project Pipeline for Terrain-Adaptive Quadruped Locomotion

Our initial experiments focused on evaluating the viability and performance of two simulation frameworks: Gymnasium and MuJoCo Playground. Gymnasium provided an accessible entry point for prototyping, offering simple humanoid environments with default MuJoCo binaries on both Windows and macOS/Linux. Its integration with PPO algorithms via Stable-Baselines3 allowed for rapid experimentation. However, when scaling to quadruped control—specifically with the Unitree robots—Gymnasium revealed several limitations. It lacked a prebuilt Unitree model, and its CPU-bound execution hindered training efficiency, even in basic flat-terrain environments.

In contrast, MuJoCo Playground offered substantial improvements in both simulation fidelity and computational performance. By utilizing GPU acceleration, we were able to train control policies on an A100 GPU, achieving a benchmark training score of 30.2 out of 40. Despite substantial performance improvements, we observed that the GPU was not fully utilized, suggesting room for

further software-level optimization. Indeed, re-running the same benchmark task using the original MuJoCo developer code on a consumer-grade RTX 4090 GPU resulted in even faster training times—approximately 7 minutes—underscoring the importance of software optimization in addition to hardware acceleration. MuJoCo Playground effectively addresses the CPU bottlenecks of traditional MuJoCo setups by incorporating GPU-centric simulation and training libraries.

To explore further acceleration opportunities, we evaluated Brax, a differentiable physics engine written in JAX that supports native GPU execution and scalable reinforcement learning (Freeman et al. [2021]). Brax integrates directly with RL algorithms like PPO, allowing for massive parallelization during training. In our tests, Brax reduced training times from several hours or days to just a few minutes, highlighting its potential for rapid iteration and large-scale experimentation. Ultimately, based on these findings, we selected MuJoCo Playground as the primary simulation framework for our work. It provided the best balance of fidelity, extensibility, and performance for terrain-adaptive quadruped training, while allowing us to incorporate additional performance optimizations inspired by Brax's parallel training paradigm.

3.2 Training

For this project, we adopted a state-based Proximal Policy Optimization (PPO) framework to train joystick-style locomotion policies, wherein the robot learns to follow a user-defined velocity vector across diverse terrains. Our implementation built directly upon the existing PPO infrastructure provided by MuJoCo Playground, which includes carefully tuned defaults suited for legged locomotion. The policy network architecture consisted of a multilayer perceptron (MLP) with three hidden layers. We employed standard PPO hyperparameters, including a learning rate of 3e-4, a discount rate greater than 0.9, and an entropy coefficient to encourage exploration during early training.

Each policy was trained using observations from the robot's proprioceptive state, including joint angles, velocities, body orientation, and the target velocity vector. The action space consisted of continuous joint commands for all actuated degrees of freedom. Training was conducted using rollout lengths and batch sizes consistent with MuJoCo Playground defaults, typically involving a large number of parallel environments to accelerate sample collection, fully utilizing the capabilities of the JAX framework. Training and evaluation environments were set depending on which terrain we wanted the model to adapt to. For instance, for a model that is looking to move on icy ground, that icy ground would be its main training environment rather than the default controlled environment. Checkpoints were saved at regular intervals, allowing us to evaluate terrain-specific policies and compare performance across different environments such as icy floors, uneven terrain, and stairs. This model served as the basis for our Sim2Real deployment efforts on the physical Unitree Go2 robot.

The reward function used to train our quadruped locomotion policy is carefully shaped to encourage goal-directed movement while penalizing inefficient or unstable behavior. In environments in which there are a lot of varying factors, stability and upright posture are pivotal in keeping the robot in a position where it could successfully move. The core objective is to maximize accurate tracking of the commanded linear and angular velocities, which is reflected in the relatively high positive weights assigned to linear velocity tracking(1.0) and angular velocity tracking(0.5). This incentivizes the policy to move in the direction and rotational speed specified by the joystick input.

Simultaneously, the policy is penalized for behaviors that deviate from physically realistic or energy-efficient locomotion. Negative rewards for torques, action-rate, and energy act as regularization terms, discouraging excessive or erratic control inputs that could lead to unstable motion or high energy usage—an important consideration for real-world deployment on hardware-constrained robots. The 'orientation' parameter is heavily penalized (-5.0), which places strong emphasis on maintaining a level and upright body posture. This reflects a design choice to enforce stability on uneven or sloped terrains. Other terms such as feet clearance, feet slip, and feet air time contribute to the realism of gait patterns by penalizing dragging or insufficient clearance during stepping motions.

Additionally, the stand-still and termination penalties further discourage the robot from freezing or falling, guiding the policy toward sustained, uninterrupted locomotion. However, due to stability being the priority, these penalties were tweaked to test if stopping to find balance would help the robot's movement in the long run. Overall, the reward function is designed to strike a balance between tracking performance, energetic efficiency, and gait realism, enabling the trained policy to

generalize effectively across a variety of terrains and prepare for eventual Sim2Real deployment on the Go2.

4 Sim2Real Deployment on Unitree Go2

This section outlines our efforts to deploy a trained locomotion policy onto the physical Unitree Go2 robot. While our simulation framework was originally built for the Go1, we adapted it to the Go2 by integrating its official XML model into MuJoCo and defining custom task environments—flat, icy, rough, and stairs—mirroring the Go1 setup. Due to the structural and control similarities between the two robots, we were able to reuse the same PPO architecture and hyperparameters, although we had to adjust reward parameters and state functions. Policy training remained efficient and consistent with our terrain-adaptive approach. However, physical deployment presented unexpected challenges. Our initial attempt to run the MuJoCo Playground policy via Unitree's SDK2 and ROS 2 drivers showed successful network communication, but failed to execute any bundled demo programs. Cross-testing in the lab with a different workstation and another Go2 unit revealed that the issue was specific to our robot—not the code or environment—suggesting a firmware or hardware restriction.

Further investigation confirmed that the issue stemmed from hardware edition differences. The lab's Go2 was an Edu model, which allows custom code execution even after firmware updates, whereas our Go2 was a non-Edu (Pro/Lite) version locked by Secure Boot, preventing unsigned binaries from running. To overcome this limitation, we performed a community-driven jailbreak, gaining root access and unlocking the low-level motor API needed for RL deployment. While this enabled command execution, a critical sensor limitation remained: the API did not provide global linear velocity, a key signal for our velocity-tracking policy. We experimented with four workarounds—zeroing velocity, integrating IMU data, removing the velocity term from training, and injecting a prerecorded velocity curve from simulation. None yielded stable locomotion. Although the robot managed brief standing and tentative steps, these attempts fell short of reliable performance, highlighting the need for either velocity estimation on-board or retraining with alternative sensing inputs to achieve successful Sim2Real transfer on non-Edu Go2 units.

5 Results

The table below summarizes the final episode reward values for our trained terrain-specific policy and the baseline flatland policy (trained without randomization) across three different environments: flatland, icy, and rough+icy.

Table 1: Final Episode Rewards Across Environments

Policy	**Flatland**	**Icy Terrain**	**Rough + Icy Terrain**
Terrain-Robust Policy	25.217	27.521	15.339
Baseline Flatland Policy (No Randomizer)	27.245	23.849	28.826

5.1 Supplementary Videos

The following videos demonstrate our policy performance and deployment efforts:

- **Terrain-Robust Policy on Icy Terrain:** https://youtu.be/G1Js6WhUQus
- **Baseline Policy on Icy Terrain:** https://youtu.be/2Bjnxv_U17I
- **Go2 Deployment:** https://youtu.be/TBclp5C1vkw

In the Go2 deployment video, we observe the model successfully standing and taking a step forward. Although it stumbles, it maintains orientation and avoids being penalized for falling, showcasing partial transfer of balance behavior learned in simulation.

6 Analysis

At first glance, the baseline policy achieves higher reward in the most complex environment (rough + icy), and even slightly outperforms our terrain-trained policy on flatland. However, these raw reward values are not fully indicative of successful behavior. During rollout, we observed that the baseline model—while achieving superficially high reward—failed to exhibit meaningful locomotion on non-flat terrains. In both the icy and rough + icy conditions, it struggled to maintain balance, often standing still or falling shortly after episode start. This likely indicates reward hacking, where the policy optimizes for standing upright or minimizing penalties without achieving actual task success (i.e., forward motion).

In contrast, our terrain-trained policy successfully adapted to each environment, demonstrating stable upright posture and directed locomotion. Its slightly lower reward on flatland is expected, as it was optimized for more challenging conditions, and not explicitly tuned for performance on smooth surfaces. On the most difficult terrain (rough + icy), although the reward was lower than the baseline, the policy was able to maintain balance and maneuver, suggesting that the reward function alone does not fully reflect behavioral quality in challenging environments.

Our results demonstrate that policies trained directly on icy and rough terrains were significantly more effective at achieving stable locomotion than baseline policies trained exclusively on flat ground. When evaluated on their respective terrains, the domain-specific policies enabled the robot to maintain balance and forward progress, whereas flatland-trained policies often failed to produce meaningful movement on more challenging surfaces. Notably, both types of policies achieved similar average rewards during training, suggesting that the baseline policies may have engaged in reward hacking—optimizing for terrain-specific reward signals (e.g., standing still to avoid penalties) without learning generalizable locomotion strategies. This reinforces the importance of training policies in environments that reflect the complexity of their deployment setting.

An additional strength of our approach lies in its training efficiency. Using an A100 GPU, we were able to train each policy, trained over 100 million epochs, in as little as 6 to 13 minutes, regardless of terrain complexity. This rapid turnaround greatly accelerates the Sim2Real pipeline, enabling faster iteration, testing, and deployment of policies across a variety of conditions. Such short training cycles are especially valuable when adapting to new robot embodiments or environmental conditions, as they reduce the time and computational cost typically associated with reinforcement learning in robotics.

6.1 Policy Transferability

One of the key challenges encountered in this project was the issue of policy transferability across robot embodiments. Our training pipeline was originally developed and tested using the Unitree Go1 model in simulation. However, due to hardware availability, physical deployment was carried out on the Unitree Go2 robot. While these two platforms share a broadly similar quadruped morphology and actuation scheme, they differ in several critical aspects, including joint range limits, inertial properties, and low-level control responses. As a result, policies trained exclusively on the Go1 model did not transfer seamlessly to the Go2. In particular, we observed that the Go1-trained policy often failed to produce forward motion on the Go2 without further tuning, frequently defaulting to standing still or losing balance.

This outcome underscores a broader and well-known challenge in robotics: reinforcement learning policies are highly sensitive to the physical embodiment of the robot, and even minor variations in kinematics or dynamics can degrade performance substantially. In our case, we had to retrain the policy from scratch on the Go2 model and adjust key reward parameters—particularly those tied to linear velocity tracking—to achieve somewhat functional behavior. This retraining requirement diminishes the practicality of applying pre-trained policies across platforms and highlights the need for more robust transfer mechanisms.

To address this limitation, future work could explore the use of embodiment-conditioned policies, where the policy receives robot-specific parameters—such as link lengths, joint limits, or mass distributions—as part of its observation space. This conditioning could allow the same policy architecture to generalize its behavior across a range of morphologies. Another promising direction is morphological domain randomization, where the agent is trained across a population of simulated

robots with varying body structures. Such an approach encourages the emergence of locomotion strategies that are less reliant on specific dynamics and more adaptable to unseen hardware. Additionally, meta-learning or modular control policies may provide viable paths forward by promoting parameter sharing and flexible control primitives that adapt to embodiment changes with minimal fine-tuning.

Ultimately, overcoming embodiment sensitivity is critical for developing truly transferable locomotion policies, and remains an open and active area of research within the Sim2Real community.

7 Challenges

The primary limitation encountered in this project emerged during the physical deployment phase onto the Unitree Go2 robot. While our simulation pipeline was successfully adapted from the Go1 model to the Go2 within MuJoCo, the transition to real-world testing was hindered by compatibility issues with the Unitree Go2 SDK. Despite ensuring both the control machine and robot were on the same network segment, we were unable to establish reliable communication with the robot through the SDK. This effectively blocked any attempt at executing trained policies on the physical hardware.

This limitation was exacerbated by the lack of official documentation and developer support for the Go2 SDK. The Go2's closed ecosystem and minimal guidance created a significant barrier to Sim2Real transfer. This underscores a broader challenge in robotic research, where hardware access and manufacturer support can significantly impact reproducibility and deployment success.

While our simulation results demonstrate strong policy performance across varying terrains, the inability to validate these policies on physical hardware remains an open gap in our experimental pipeline. Future work will require either improved support from the hardware vendor or the development of a custom low-level communication layer capable of bypassing SDK limitations.

8 Future Work

A critical area of future work is resolving the velocity feedback limitation that currently blocks Sim2Real deployment. While we successfully trained and evaluated policies in simulation, our physical Unitree Go2 robot lacks accessible global velocity estimates through its low-level API—a key input for our velocity-tracking reward function. Two paths forward appear viable: integrating an external velocity sensor (e.g., optical flow or time-of-flight camera), or exploring the unlocked firmware to locate an undocumented internal velocity signal. Either solution would enable us to re-tune the reward function and reliably execute policies on physical hardware, completing the Sim2Real loop.

A second area of future work involves increasing the complexity of the training environments to better reflect real-world challenges. Introducing more difficult tasks—such as climbing irregular stairs, traversing slopes, or adapting to dynamic terrain changes—will require stronger generalization and more strategic exploration. To address this, we plan to implement curriculum learning, progressively increasing environment difficulty to guide the policy toward mastering complex behaviors in a structured and sample-efficient manner.

Finally, we envision extending our system's capability by mapping natural language commands to locomotion behaviors. For example, commands such as "Follow," "Turn around," or "Stop" could be grounded in sensor-based control objectives like tracking a human, turning 180 degrees, or halting motion. This would move the system toward more intuitive, interactive deployment in real-world human-robot collaboration scenarios and open the door to high-level decision-making layered atop low-level locomotion policies.

References

Xiaoyu Chen, Jiachen Hu, Chi Jin, Lihong Li, and Liwei Wang. Understanding domain randomization for sim-to-real transfer. *arXiv preprint arXiv:2110.03239*, 2021. URL https://arxiv.org/abs/2110.03239.

C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem. Brax – a differentiable physics engine for large scale rigid body simulation. *arXiv*, 2021. doi: 10.48550/arXiv.2106.13281. URL https://doi.org/10.48550/arXiv.2106.13281.

M. Matthews, M. Beukman, C. Lu, and J. Foerster. Kinetix: Investigating the training of general agents through open-ended physics-based control tasks. *arXiv preprint arXiv:2410.23208*, 2025. URL https://arxiv.org/abs/2410.23208.

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

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018. URL https://arxiv.org/abs/1804.10332.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017. URL https://arxiv.org/abs/1703.06907.

A. Yu, G. Yang, R. Choi, Y. Ravan, J. Leonard, and P. Isola. Learning visual parkour from generated images. *arXiv preprint arXiv:2411.00083*, 2024. URL https://arxiv.org/abs/2411.00083.

K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, C. Sferrazza, Y. Tassa, and P. Abbeel. Mujoco playground. *arXiv preprint arXiv:2502.08844*, 2025. URL https://arxiv.org/abs/2502.08844.