

Learning Humanoid Control from Simulation to Real to Simulation

Tairan He

CMU-RI-TR-25-38

April 25, 2025

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:

Prof. Guanya Shi, *advisor*
Prof. Changliu Liu, *advisor*
Prof. Kris Kitani
Zhengyi Luo

*Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Robotics.*

Copyright © 2025 Tairan He. All rights reserved.

To all my advisors, collaborators, friends, family, and robots.

Abstract

Humanoid robots possess exceptional potential due to their capacity to perform versatile and human-like whole-body tasks. However, realizing agile and coordinated motions remains challenging, primarily due to the discrepancy in dynamics between simulations and real-world environments. Current methodologies, including system identification (SysID) and domain randomization (DR), often demand extensive parameter tuning efforts or yield overly cautious policies, thus compromising agility.

In this thesis, we introduce **ASAP** (Aligning Simulation and Real Physics), a two-stage framework aimed at effectively addressing dynamics mismatches and enabling agile whole-body movements in humanoid robots. During the first stage, motion-tracking policies are pre-trained in simulation leveraging retargeted human motion data. In the second stage, policies are deployed on a real-world humanoid robot to gather data for training a delta (residual) action model, which compensates for simulation-to-reality dynamics differences. Subsequently, **ASAP** fine-tunes the pre-trained policies by incorporating the delta action model within the simulator, resulting in improved alignment with real-world dynamics.

Our evaluation encompasses three distinct transfer scenarios: IsaacGym to IsaacSim, IsaacGym to Genesis, and IsaacGym to the physical Unitree G1 humanoid robot. Results demonstrate that **ASAP** significantly enhances agility and whole-body coordination compared to baseline approaches such as SysID, DR, and traditional delta dynamics learning techniques. By enabling highly agile and expressive motions previously challenging to achieve, our approach underscores the effectiveness of delta action learning in bridging the sim-to-real gap. This work highlights a promising direction for advancing agile humanoid robotics.

Acknowledgments

I would first like to express my deepest gratitude to my wonderful advisors, Professor Guanya Shi and Professor Changliu Liu, for their inspiring guidance, unwavering support, and patience throughout my journey at Carnegie Mellon University. Their mentorship has profoundly shaped my growth not only as a researcher but also as a person. Without their insightful advice, candid feedback, and continuous encouragement, this thesis would not have reached its current form.

I am also sincerely thankful to my thesis committee members, Professor Kris Kitani and Zhengyi Luo, for their invaluable input, constructive feedback, and generous sharing of expertise. Professor Kitani has provided thoughtful guidance and generously shared his insights, significantly enriching my academic experience. Zhengyi has been a constant and reliable mentor, generously sharing his knowledge, ideas, and experiences, and patiently guiding me through numerous research challenges. I deeply cherish the invaluable lessons and perspectives I have learned from both of them.

Furthermore, I extend my heartfelt appreciation to my friends and family, whose unwavering emotional support, understanding, and encouragement have been indispensable throughout my master’s studies. Their companionship, love, and motivation have carried me through challenging moments, stress, and uncertainty inherent to research, making this achievement possible.

Lastly, I want to thank all the robots that accompanied me through this research journey—especially the Unitree Go1, H1, and G1. These robots shared countless late nights and early mornings with me, experiencing together the highs and lows of experimental trials. I am genuinely grateful for their “accompany,” and I sincerely apologize for the occasional mishaps that left them less intact than I wished. This thesis is dedicated in part to these tireless robotic companions who made this work both possible and memorable.

Contents

1	Introduction	1
2	Background	5
2.1	Learning-based Humanoid Control	5
2.2	Offline and Online System Identification	6
2.3	Residual Learning for Robotics	6
3	Methods	9
3.1	Pre-training: Acquiring Agile Humanoid Motions from Human Data .	9
3.1.1	Data Generation via Retargeting Human Motion	9
3.1.2	Phase-conditioned Motion Tracking via Reinforcement Learning	10
3.2	Post-training: Learning Delta Action Model and Fine-tuning Policies for Real-World Dynamics	13
3.2.1	Real-World Data Collection	13
3.2.2	Delta Action Model Learning	14
3.2.3	Fine-tuning Policy under Corrected Simulation Dynamics . . .	15
3.2.4	Deployment on Real Humanoid Robot	16
4	Experiments	17
4.1	Performance Evaluation of ASAP	17
4.1.1	Comparison of Dynamics Matching Capability	19
4.1.2	Comparison of Policy Fine-Tuning Performance	21
4.1.3	Real-World Evaluations	23
4.2	Extensive Studies and Analyses	25
4.2.1	Key Factors in Training Delta Action Models	25
4.2.2	Different Usage of Delta Action Model	27
4.2.3	Does ASAP Fine-Tuning Outperform Random Action Noise Fine-Tuning?	27
5	Conclusions	33
	Bibliography	35

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

1.1	Overview of ASAP . (a) Motion Tracking Pre-training and Real Trajectory Collection : With the humanoid motions retargeted from human videos, we pre-train multiple motion tracking policies to roll out real-world trajectories. (b) Delta Action Model Training : Based on the real-world rollout data, we train the delta action model by minimizing the discrepancy between simulation state s_t and real-world state s_t^r . (c) Policy Fine-tuning : We freeze the delta action model, incorporate it into the simulator to align the real-world physics and then fine-tune the pre-trained motion tracking policy. (d) Real-World Deployment : Finally, we deploy the fine-tuned policy directly in the real world without delta action model.	2
1.2	The humanoid robot (Unitree G1) demonstrates diverse agile whole-body skills, showcasing the control policies' agility: (a) Cristiano Ronaldo's signature celebration involving a jump with a 180-degree mid-air rotation; (b) LeBron James's "Silencer" celebration involving single-leg balancing; and (c) Kobe Bryant's famous fadeaway jump shot involving single-leg jumping and landing; (d) 1.5m-forward jumping; (e) Leg stretching; (f) 1.3m-side jumping.	4
3.1	Retargeting Human Video Motions to Robot Motions: (a) Human motions are captured from video. (b) Using TRAM [93], 3D human motion is reconstructed in the SMPL parameter format. (c) A reinforcement learning (RL) policy is trained in simulation to track the SMPL motion. (d) The learned SMPL motion is retargeted to the Unitree G1 humanoid robot in simulation. (e) The trained RL policy is deployed on the real robot, executing the final motion in the physical world. This pipeline ensures the retargeted motions remain physically feasible and suitable for real-world deployment.	10

4.1	Replaying IsaacSim State-Action trajectories in IsaacGym. The upper four panels visualize the Unitree G1 humanoid executing a soccer-shooting motion under four distinct open-loop actions. Corresponding metric curves (bottom) quantify tracking performance. Importantly, our delta action model (ASAP) is trained across multiple motions and is not overfitted to this specific example.	18
4.2	Baselines of ASAP. (a) Model-free RL training. (b) System ID from real to sim using real-world data. (c) Learning delta dynamics model using real-world data. (d) Our proposed method, learning delta action model using real-world data.	19
4.3	Visual comparisons of motion imitation results across different difficulty levels (Easy, Medium, Hard) for various tasks including Jump Forward, Side Jump, Single Foot Balance, Squat, Step Backward, Step Forward, and Walk.	21
4.4	Visualization of G1 motion tracking before and after ASAP fine-tuning in IsaacGym, IsaacSim and Genesis. Top: LeBron James’ “Silencer” motion tracking policy fine-tuning for IsaacGym to IsaacSim. Bottom: <i>single foot balance</i> motion tracking policy fine-tuning for IsaacGym to Genesis.	22
4.5	Visualization of LeBron James’ “Silencer” motion on the G1 robot before (upper figure enclosed in blue) and after (bottom figure enclosed in red) ASAP policy finetuning. The left half shows the policy fine-tuning for the in-distribution motions while the right half shows the out-of-distribution ones. After ASAP finetuning, the robot behaves more smoothly and reduces jerky lower-body motions.	24
4.6	We deploy the pretrained policy of a forward jump motion tracking task, challenging the 1.35m-tall Unitree G1 robot for a forward leap over 1m.	25

4.7	Analysis of dataset size, training horizon, and action norm on the performance of π^Δ . (a) Dataset Size: Mean Per Joint Position Error (MPJPE) is evaluated for both in-distribution (green) and out-of-distribution (blue) scenarios. Increasing dataset size leads to enhanced generalization, evidenced by decreasing errors in out-of-distribution evaluations. Closed-loop MPJPE (red bars) also shows improvement with larger datasets. (b) Training Horizon: Open-loop MPJPE (heatmap) improves across evaluation points as training horizons increase, achieving the lowest error at 1.5s. However, closed-loop MPJPE (red bars) shows a sweet spot at a training horizon of 1.0s, beyond which no further improvements are observed. The red dashed line represents the pretrained baseline without π^Δ fine-tuning. (c) Action Norm: The action norm weight significantly influences performance. Both open-loop and closed-loop MPJPE decrease as the weight increases up to 0.1, achieving the lowest error. However, further increases in the action norm weight result in degradation of open-loop performance, highlighting the trade-off between action smoothness and policy flexibility.	29
4.8	MPJPE comparison over timesteps for fine-tuning methods using delta actionmodel. RL Fine-Tuning achieves the lowest error, while Fixed-Point Iteration and Gradient Search perform worse than the baseline (Before DeltaA) showing the highest error.	30
4.9	MPJPE vs. Noise Level for policies fine-tuned with random action noise. Policies with noise levels $\beta \in [0.025, 0.2]$ show improved performance compared to no fine-tuning. Delta action achieves better tracking precision (126 MPJPE) compared to the best action noise (173 MPJPE).	30
4.10	Visualization of IsaacGym-to-IsaacSim π^Δ output magnitude. We compute the average absolute value of each joint over the 4300-episode dataset. Larger red dots indicate higher values. The results suggest that lower-body motors exhibit a larger discrepancy compared to upper-body joints, with the most significant gap observed in the ankle pitch joint of the G1 humanoid.	31

List of Tables

3.1	Reward Terms for Pretraining	12
3.2	Domain Randomizations	13
3.3	Reward Terms for Delta Action Learning	15
4.1	SysID Parameters	19
4.2	Open-loop performance comparison across simulators and motion lengths.	20
4.3	Closed-loop motion imitation evaluation across different simulators. All variants are trained with identical rewards.	20
4.4	Real-world closed-loop performance comparing with and without ASAP finetuning on one in-distribution motion and one out-of-distribution motion.	26

Chapter 1

Introduction

For decades, researchers have envisioned humanoid robots capable of achieving—or even surpassing—human-level agility. Despite substantial progress, prior studies [19, 46, 47, 50, 73, 74, 95, 107] have primarily focused on locomotion tasks, viewing legs predominantly as instruments for mobility. More recent efforts [10, 24, 25, 26, 32] have expanded the focus towards whole-body expressiveness, particularly upper-body movements, yet the agile and coordinated motions characteristic of human behavior remain elusive. A significant hurdle in achieving agile whole-body humanoid skills arises from inherent hardware limitations coupled with the persistent mismatch between simulated dynamics and real-world physics.

Existing methodologies for addressing this simulation-to-reality gap broadly fall into three categories: system identification (SysID), domain randomization (DR), and learned dynamics approaches. SysID methods estimate essential physical parameters, such as motor response characteristics, link masses, and terrain interactions [19, 102]. However, SysID methods typically depend on predefined parameter spaces [49], making them prone to inaccuracies when encountering dynamics not captured within the modeled distribution. Moreover, SysID often necessitates ground truth torque measurements [29], which are typically unavailable on common hardware platforms, thus restricting practical use. Alternatively, DR methods attempt to bridge the dynamics mismatch by extensively randomizing simulation parameters during policy training [59, 68, 79, 85, 87]. However, this randomization tends to yield overly conservative policies [25], hindering the attainment of high agility. Lastly, learned

1. Introduction

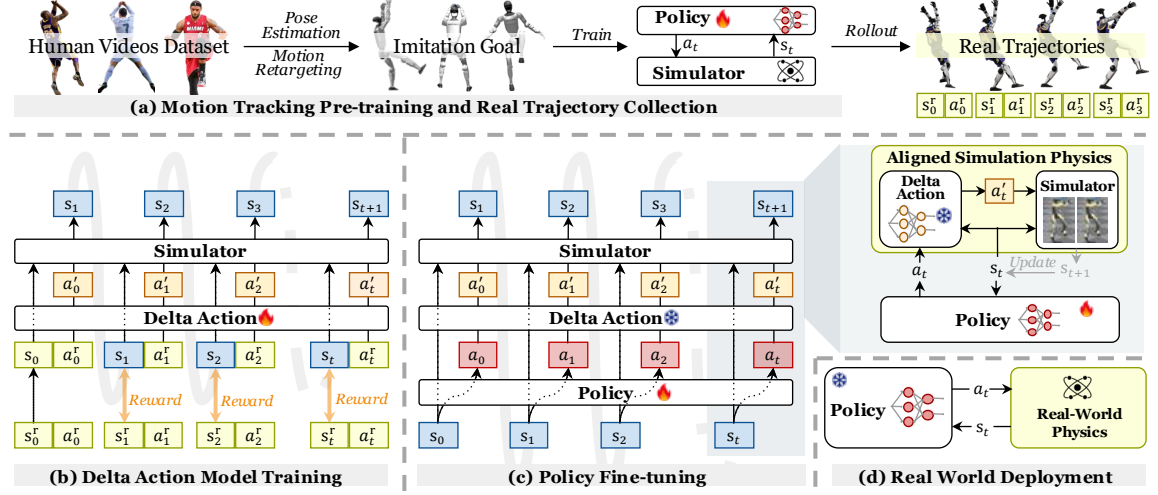


Figure 1.1: Overview of **ASAP**. (a) **Motion Tracking Pre-training and Real Trajectory Collection**: With the humanoid motions retargeted from human videos, we pre-train multiple motion tracking policies to roll out real-world trajectories. (b) **Delta Action Model Training**: Based on the real-world rollout data, we train the delta action model by minimizing the discrepancy between simulation state s_t and real-world state s_t^r . (c) **Policy Fine-tuning**: We freeze the delta action model, incorporate it into the simulator to align the real-world physics and then fine-tune the pre-trained motion tracking policy. (d) **Real-World Deployment**: Finally, we deploy the fine-tuned policy directly in the real world without delta action model.

dynamics approaches have successfully captured real-world dynamics from data in low-dimensional robotic systems, such as drones [81] and ground vehicles [97], yet their effectiveness for humanoid robots remains largely unexplored.

Motivated by these limitations, this thesis proposes **ASAP**, a two-stage reinforcement learning (RL) framework designed explicitly to align simulated dynamics with real-world physics, thereby enabling agile whole-body humanoid behaviors. In the **pre-training stage**, we train base policies in simulation by leveraging human motion capture data. Specifically, human motions are retargeted onto humanoid robot models [25], after which a phase-conditioned policy [67] is trained to closely track these retargeted movements. Nonetheless, directly transferring this policy to physical hardware results in notable performance degradation due to dynamics discrepancies.

To mitigate this challenge, the **post-training stage** employs real-world rollout data collected from the humanoid robot, including proprioceptive states and positional data captured via a motion capture system. This data is subsequently replayed in

simulation, exposing the dynamics mismatch through observed tracking errors. A delta action model, trained via supervised learning on these discrepancies, is then developed to predict residual actions required to compensate for real-world dynamics variations. This learned residual effectively bridges the simulation-to-reality gap. Finally, the pre-trained policy undergoes fine-tuning within the augmented simulation environment, incorporating the delta action model to adapt robustly to real-world physics.

We rigorously validate **ASAP** through extensive experiments, showcasing agile whole-body skills on the Unitree G1 humanoid robot [77]. Experimental results demonstrate that our approach significantly surpasses baseline methods (SysID, DR, and traditional learned dynamics) in terms of motion tracking accuracy, evaluated across both sim-to-sim scenarios (IsaacGym to IsaacSim, IsaacGym to Genesis) and sim-to-real transfers (IsaacGym to Real).

The primary contributions of this thesis are as follows:

1. We introduce **ASAP**, an effective RL-based sim-to-real alignment framework that leverages a delta action model trained with real-world robot data.
2. We demonstrate successful real-world deployment of RL-based whole-body control policies, achieving agile and dynamic humanoid movements previously unattainable.
3. Extensive experimental evaluations across multiple scenarios confirm that **ASAP** significantly reduces dynamics mismatch and tracking errors, facilitating highly agile humanoid robot skills.
4. To support further research and seamless transfer between simulation platforms, we release an open-source multi-simulator training and evaluation codebase.

1. Introduction



Figure 1.2: The humanoid robot (Unitree G1) demonstrates diverse agile whole-body skills, showcasing the control policies' agility: (a) Cristiano Ronaldo's signature celebration involving a jump with a 180-degree mid-air rotation; (b) LeBron James's "Silencer" celebration involving single-leg balancing; and (c) Kobe Bryant's famous fadeaway jump shot involving single-leg jumping and landing; (d) 1.5m-forward jumping; (e) Leg stretching; (f) 1.3m-side jumping.

Chapter 2

Background

2.1 Learning-based Humanoid Control

In recent years, learning-based methods have made significant progress in whole-body control for humanoid robots. Primarily leveraging reinforcement learning algorithms [80] within physics simulators [58, 63, 88], humanoid robots have learned a wide range of skills, including robust locomotion [19, 44, 45, 47, 48, 73, 74, 98, 106], jumping [46], and parkour [50, 107]. More advanced capabilities, such as dancing [10, 32, 105], loco-manipulation [15, 24, 25, 53], and even backflipping [78], have also been demonstrated. Meanwhile, the humanoid character animation community has achieved highly expressive and agile whole-body motions in physics-based simulations [55, 71, 86], including cartwheels [67], backflips [69], sports movements [56, 90, 91, 92, 104], and smooth object interactions [17, 22, 86]. However, transferring these highly dynamic and agile skills to real-world humanoid robots remains challenging due to the dynamics mismatch between simulation and real-world physics. To address this challenge, our work focuses on learning and compensating for this dynamics mismatch, enabling humanoid robots to perform expressive and agile whole-body skills in the real world.

2.2 Offline and Online System Identification

The dynamics mismatch between simulators and real-world physics can be attributed to two primary factors: inaccuracies in the robot model descriptions and the presence of complex real-world dynamics that are difficult for physics-based simulators to capture. Traditional approaches address these issues using system identification (SysID) methods [5, 39], which calibrate the robot model or simulator based on real-world performance. These methods can be broadly categorized into *offline SysID* and *online SysID*, depending on whether system identification occurs at test time. *Offline SysID* methods typically collect real-world data and adjust simulation parameters to train policies in more accurate dynamics. The calibration process may focus on modeling actuator dynamics [29, 85, 99], refining robot dynamics models [2, 18, 21, 30, 36], explicitly identifying critical simulation parameters [9, 13, 96, 102], learning a distribution over simulation parameters [4, 27, 75], or optimizing system parameters to maximize policy performance [64, 76]. *Online SysID* methods, in contrast, aim to learn a representation of the robot’s state or environment properties, enabling real-time adaptation to different conditions. These representations can be learned using optimization-based approaches [43, 70, 101, 103], regression-based methods [14, 19, 31, 40, 41, 42, 60, 61, 62, 72, 89, 100], next-state reconstruction techniques [51, 54, 65, 83, 94], direct reward maximization [47], or by leveraging tracking and prediction errors for online adaptation [16, 28, 57, 66]. Our framework takes a different approach from traditional SysID methods by learning a residual action model that directly compensates for dynamics mismatch through corrective actions, rather than explicitly estimating system parameters.

2.3 Residual Learning for Robotics

Learning a residual component alongside learned or pre-defined base models has been widely used in robotics. Prior work has explored residual policy models that refine the actions of an initial controller [1, 3, 8, 12, 20, 33, 34, 42, 84]. Other approaches leverage residual components to correct inaccuracies in dynamics models [23, 35, 38, 66, 82] or to model residual trajectories resulting from residual actions [11] for achieving precise and agile motions. RGAT [35] uses a residual action model with a learned

forward dynamics to refine the simulator. Our framework builds on this idea by using RL-based residual actions to align the dynamics mismatch between simulation and real-world physics, enabling agile whole-body humanoid skills.

2. Background

Chapter 3

Methods

3.1 Pre-training: Acquiring Agile Humanoid Motions from Human Data

3.1.1 Data Generation via Retargeting Human Motion

To achieve expressive and agile humanoid robot motions, we collect and process human motion video data, converting it into robot-compatible trajectories. The overall data generation pipeline is illustrated in [Figure 3.1](#) and [Figure 1.1](#) (a).

Human Motion Capture and 3D Reconstruction We begin by capturing human videos performing diverse agile movements (examples shown in [Figure 3.1\(a\)](#) and [Figure 4.9](#)). These recorded videos are then processed using TRAM [\[93\]](#), a state-of-the-art method for reconstructing 3D human motions in the SMPL format [\[52\]](#). TRAM extracts global translation, orientation, and articulated body poses, resulting in the dataset $\mathcal{D}_{\text{SMPL}}$, as visualized in [Figure 3.1\(b\)](#).

Physics-based Motion Validation Despite TRAM’s effectiveness, the reconstructed SMPL motions can sometimes contain physically infeasible movements or artifacts due to reconstruction noise [\[25\]](#). To address this, we perform a physics-based validation (“sim-to-data”) step using MaskedMimic [\[86\]](#), a physics-informed

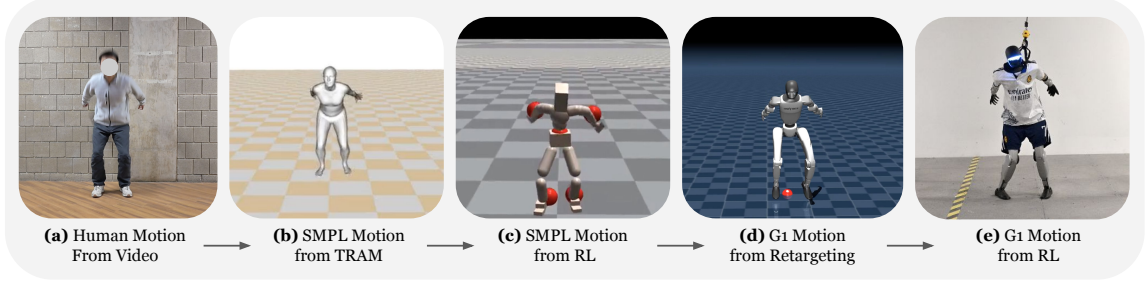


Figure 3.1: Retargeting Human Video Motions to Robot Motions: (a) Human motions are captured from video. (b) Using TRAM [93], 3D human motion is reconstructed in the SMPL parameter format. (c) A reinforcement learning (RL) policy is trained in simulation to track the SMPL motion. (d) The learned SMPL motion is retargeted to the Unitree G1 humanoid robot in simulation. (e) The trained RL policy is deployed on the real robot, executing the final motion in the physical world. This pipeline ensures the retargeted motions remain physically feasible and suitable for real-world deployment.

motion-tracking method. MaskedMimic attempts to track these reconstructed SMPL motions in the IsaacGym simulator [58]. Motions successfully tracked without significant deviations are retained, forming the cleaned dataset $\mathcal{D}_{\text{SMPL}}^{\text{Cleaned}}$, as illustrated in Figure 3.1(c).

Retargeting Human Motions to Humanoid Robots To create robot-compatible motions, we follow a two-stage shape-and-motion retargeting approach [25]. First, we optimize the SMPL body shape parameter β' to closely resemble the robot’s physical proportions by minimizing joint position discrepancies at rest pose. Next, given the optimized body shape β' , we perform an additional optimization step on the original SMPL motion parameters (translation $\mathbf{p}$ and pose θ) to further minimize body-link positional differences between the human and robot models. This procedure yields a high-quality, robot-compatible trajectory dataset $\mathcal{D}_{\text{Robot}}^{\text{Cleaned}}$, visualized in Figure 3.1(d).

3.1.2 Phase-conditioned Motion Tracking via Reinforcement Learning

We approach motion tracking as a goal-conditioned reinforcement learning (RL) problem, training policies to closely follow trajectories from $\mathcal{D}_{\text{Robot}}^{\text{Cleaned}}$. Inspired by [67],

we represent the robot state at time t as a combination of proprioception s_t^p and a normalized phase variable $\phi \in [0, 1]$, with $\phi = 0$ and $\phi = 1$ corresponding to the motion’s start and end, respectively. The phase variable alone serves as the goal condition s_t^g , simplifying real-world deployment by avoiding reliance on precise odometry.

The robot’s proprioceptive state is defined as:

$$s_t^p \triangleq [\mathbf{q}_{t-4:t}, \dot{\mathbf{q}}_{t-4:t}, \boldsymbol{\omega}_{t-4:t}^{root}, \mathbf{g}_{t-4:t}, \mathbf{a}_{t-5:t-1}], \quad (3.1)$$

which includes the recent history (past 5 timesteps) of joint positions $\mathbf{q}_t \in \mathbb{R}^{23}$, joint velocities $\dot{\mathbf{q}}_t \in \mathbb{R}^{23}$, root angular velocity $\boldsymbol{\omega}_t^{root} \in \mathbb{R}^3$, gravity direction in the root frame $\mathbf{g}_t \in \mathbb{R}^3$, and previous action $\mathbf{a}_{t-1} \in \mathbb{R}^{23}$.

We train the policy π using Proximal Policy Optimization (PPO) [80] to maximize the cumulative discounted reward $\mathbb{E}[\sum_{t=1}^T \gamma^{t-1} r_t]$, with the reward $r_t = \mathcal{R}(s_t^p, s_t^g)$ defined based on tracking accuracy, regularization, and task-specific penalties (details provided in Table 3.1). The policy output $\mathbf{a}_t \in \mathbb{R}^{23}$ represents target joint positions, implemented via a PD controller for actuation.

Key training techniques critical for successful motion-tracking include:

Asymmetric Actor-Critic Architecture Given the inherent partial observability of real-world humanoid tasks, we adopt an asymmetric actor-critic training paradigm. The critic network is provided privileged information such as global reference positions and root linear velocities, enhancing value estimation stability. In contrast, the actor receives only realistic proprioceptive information and the phase variable. This design ensures that the policy remains robust during real-world deployment, where privileged observations (e.g., perfect odometry) are unavailable, thus overcoming known deployment issues in previous work [24, 25].

Progressive Termination Curriculum Tracking complex agile motions, such as jumps and flips, is challenging due to high initial difficulty. We address this by introducing a progressive termination curriculum that starts with a lenient tracking error tolerance (1.5 meters) and gradually tightens to a stricter threshold (0.3 meters) as training progresses. This strategy allows the policy to first master basic balance

and control before progressively improving precise motion tracking capabilities.

Reference State Initialization (RSI) Effective initialization significantly impacts policy training efficiency and robustness. Inspired by the RSI method [67], we randomly select the starting phase variable ϕ at the beginning of each episode. Correspondingly, the robot’s initial state—root pose, velocities, joint angles, and velocities—is derived from the reference trajectory at this sampled phase. RSI facilitates parallel learning of various motion segments, greatly accelerating training and significantly improving tracking performance for agile motions.

Table 3.1: Reward Terms for Pretraining

Term	Weight	Term	Weight
Penalty			
DoF position limits	−10.0	DoF velocity limits	−5.0
Torque limits	−5.0	Termination	−200.0
Regularization			
Torques	-1×10^{-6}	Action rate	−0.5
Feet orientation	−2.0	Feet heading	−0.1
Slippage	−1.0		
Task Reward			
Body position	1.0	VR 3-point	1.6
Body position (feet)	2.1	Body rotation	0.5
Body angular velocity	0.5	Body velocity	0.5
DoF position	0.75	DoF velocity	0.5

Reward Function and Domain Randomization The reward function comprises tracking penalties, regularization terms, and task-specific incentives, detailed explicitly in Table 3.1. Additionally, we employ domain randomization techniques (summarized in Table 3.2) during pre-training to enhance policy robustness and generalization, thus facilitating reliable transfer to diverse environments and real-world hardware.

Table 3.2: Domain Randomizations

Term	Value
Dynamics Randomization	
Friction	$\mathcal{U}(0.2, 1.1)$
P Gain	$\mathcal{U}(0.925, 1.05) \times \text{default}$
Control delay	$\mathcal{U}(20, 40)\text{ms}$
External Perturbation	
Push robot	interval = 10s, $v_{xy} = 0.5 \text{ m/s}$

3.2 Post-training: Learning Delta Action Model and Fine-tuning Policies for Real-World Dynamics

Although policies trained in simulation effectively track reference motions, directly deploying these policies to real-world humanoid robots often results in performance degradation due to the simulation-to-reality dynamics mismatch. To bridge this gap, we propose a two-step post-training procedure, illustrated in [Figure 1.1\(b\)-\(c\)](#). First, we collect real-world data by deploying the pre-trained policy on a physical humanoid robot. Next, we leverage this data to learn a delta (residual) action model that compensates for the dynamics mismatch. Finally, we fine-tune the pre-trained policy using this adjusted simulation environment.

3.2.1 Real-World Data Collection

We deploy the policy trained during pre-training onto the humanoid robot performing agile whole-body tasks, as demonstrated in [Figure 4.6](#). During these rollouts, we collect a dataset of real-world robot trajectories, denoted as:

$$\mathcal{D}^{\text{real}} = \{s_0^{\text{real}}, a_0^{\text{real}}, \dots, s_T^{\text{real}}, a_T^{\text{real}}\}.$$

At each timestep t , we record the robot’s full state using motion-capture data and

onboard sensors:

$$s_t = [p_t^{\text{base}}, v_t^{\text{base}}, \alpha_t^{\text{base}}, \omega_t^{\text{base}}, q_t, \dot{q}_t],$$

where $p_t^{\text{base}} \in \mathbb{R}^3$ denotes the base position, $v_t^{\text{base}} \in \mathbb{R}^3$ the base linear velocity, $\alpha_t^{\text{base}} \in \mathbb{R}^4$ the base orientation (quaternion), $\omega_t^{\text{base}} \in \mathbb{R}^3$ the base angular velocity, and $q_t, \dot{q}_t \in \mathbb{R}^{23}$ are the joint positions and velocities, respectively.

3.2.2 Delta Action Model Learning

Directly replaying real-world actions (a_t^{real}) in simulation typically leads to significant deviations due to the dynamics mismatch. To capture and mitigate this discrepancy, we introduce a delta action model, π_θ^Δ , trained using reinforcement learning (RL). This model learns to predict residual corrections Δa_t to align simulated outcomes with real-world observations.

Specifically, the delta action model is defined as:

$$\Delta a_t = \pi_\theta^\Delta(s_t, a_t^{\text{real}}),$$

where s_t is the current state and a_t^{real} is the recorded real-world action. The corrected action is then applied to the simulator dynamics:

$$s_{t+1} = f^{\text{sim}}(s_t, a_t^{\text{real}} + \Delta a_t),$$

where f^{sim} represents the simulator’s forward dynamics model.

The RL training loop for the delta action model (illustrated in [Figure 1.1\(b\)](#)) consists of the following steps:

1. Initialize the robot’s simulated state at the recorded real-world state s_t^{real} .
2. Execute the corrected action ($a_t^{\text{real}} + \Delta a_t$) in simulation to obtain the next state s_{t+1} .
3. Compute a reward that penalizes the state discrepancy between simulated states s_{t+1} and the corresponding real-world state s_{t+1}^{real} , supplemented by an action magnitude regularization term $\exp(-\|a_t\|) - 1$, as detailed in [Table 3.3](#).
4. Update the delta action model π_θ^Δ using Proximal Policy Optimization (PPO) [\[80\]](#) to maximize cumulative reward and minimize the sim-to-real state mismatch.

Table 3.3: Reward Terms for Delta Action Learning

Term	Weight	Term	Weight
Penalty			
DoF position limits	−10.0	DoF velocity limits	−5.0
Torque limits	−0.1	Termination	−200.0
Regularization			
Action rate	−0.01	Action norm	−0.2
Task Reward			
Body position	1.0	VR 3-point	1.0
Body position (feet)	1.0	Body rotation	0.5
Body angular velocity	0.5	Body velocity	0.5
DoF position	0.5	DoF velocity	0.5

By learning the delta action corrections, the simulator effectively captures previously unmodeled real-world constraints. For example, if simulated motors appear stronger than real hardware, the delta action model learns to reduce the applied torques, accurately reflecting hardware limitations. Thus, this model enables more realistic simulations that closely match observed real-world behaviors.

3.2.3 Fine-tuning Policy under Corrected Simulation Dynamics

After training the delta action model, we reconstruct the simulation dynamics to incorporate these learned residuals:

$$s_{t+1} = f^{\text{ASAP}}(s_t, a_t) = f^{\text{sim}}(s_t, a_t + \pi^\Delta(s_t, a_t)).$$

We then fine-tune the original motion tracking policy (from pre-training) within this adjusted simulator, as shown in Figure 1.1(c). During fine-tuning, the delta action model parameters are kept frozen, and the policy is trained using the same tracking reward defined previously in Table 3.1. This process allows the policy to adapt to real-world dynamics implicitly through the simulator.

3.2.4 Deployment on Real Humanoid Robot

Finally, we deploy the fine-tuned policy (without the delta action model) directly on the real-world robot, as depicted in [Figure 1.1\(d\)](#). Our refined policy demonstrates significantly improved real-world motion tracking quality compared to the initial pre-trained policy. Quantitative analysis and performance improvements achieved through our post-training procedure are presented in detail in [Section 4.1](#).

Chapter 4

Experiments

4.1 Performance Evaluation of **ASAP**

In this section, we present extensive experimental results on three policy transfers: IsaacGym [58] to IsaacSim [63], IsaacGym to Genesis [6], and IsaacGym to real-world Unitree G1 humanoid robot. Our experiments aim to address the following key questions:

- **Q1:** Can **ASAP** outperform other baseline methods to compensate for the dynamics mismatch?
- **Q2:** Can **ASAP** finetune policy to outperform SysID and Delta Dynamics methods?
- **Q3:** Does **ASAP** work for sim-to-real transfer?

Experiments Setup. To address these questions, we evaluate **ASAP** on motion tracking tasks in both simulation (Section 4.1.1 and Section 4.1.2) and real-world settings (Section 4.1.3).

In the simulation, we use the retargeted motion dataset from the videos we shoot, denoted as $\mathcal{D}_{\text{Robot}}^{\text{Cleaned}}$, which contains diverse human motion sequences. We select 43 motions categorized into three difficulty levels: easy, medium, and hard (as partially visualized in Figure 4.3), based on motion complexity and the required agility. **ASAP** is evaluated through simulation-to-simulation transfer by training policies in IsaacGym and using two other simulators, IsaacSim and Genesis, as a

4. Experiments

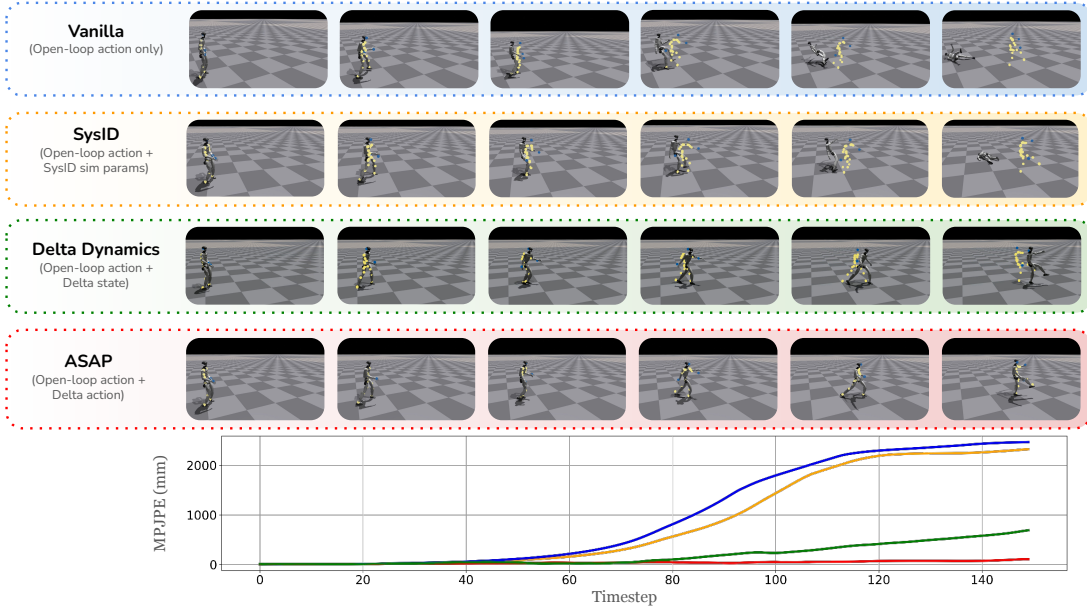


Figure 4.1: Replaying IsaacSim State-Action trajectories in IsaacGym. The upper four panels visualize the Unitree G1 humanoid executing a soccer-shooting motion under four distinct open-loop actions. Corresponding metric curves (bottom) quantify tracking performance. Importantly, our delta action model (ASAP) is trained across multiple motions and is not overfitted to this specific example.

proxy of “real-world” environments. This setup allows for a systematic evaluation of **ASAP**’s generalization and transferability. The success of the transfer is assessed by metrics described in subsequent sections.

For real-world evaluation, we deploy **ASAP** on Unitree G1 robot with fixed wrists to track motion sequences that has obvious sim-to-real gap. These sequences are chosen to capture a broad range of motor capabilities and demonstrate the sim-to-real capability for agile whole-body control.

Baselines. We have the following baselines:

Oracle: This baseline is trained and evaluated entirely within IsaacGym. It assumes perfect alignment between the training and testing environments, serving as an upper bound for performance in simulation.

Vanilla (Figure 4.2 a): The RL policy is trained in IsaacGym and evaluated in IsaacSim, Genesis, or the real world.

SysID (Figure 4.2 b): We identify the following representative parameters in our

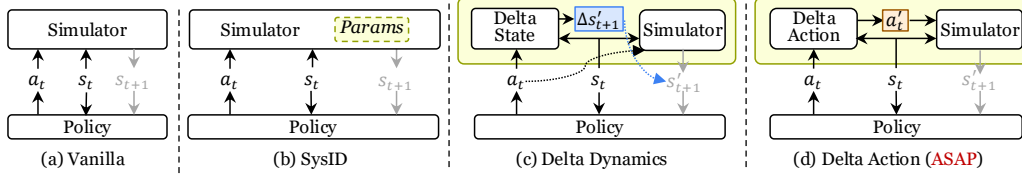


Figure 4.2: Baselines of **ASAP**. (a) Model-free RL training. (b) System ID from real to sim using real-world data. (c) Learning delta dynamics model using real-world data. (d) Our proposed method, learning delta action model using real-world data.

Table 4.1: SysID Parameters

Parameter	Range	Parameter	Range
c_x	$[-0.02, 0.0, 0.02]$	c_y	$[-0.02, 0.0, 0.02]$
c_z	$[-0.02, 0.0, 0.02]$	k_m	$[0.95, 1.0, 1.05]$
k_p^i	$[0.95, 1.0, 1.05]$	k_d^i	$[0.95, 1.0, 1.05]$

simulated model that best align the ones in the real world: base center of mass (CoM) shift (c_x, c_y, c_z) , base link mass offset ratio k_m and low-level PD gain ratios (k_p^i, k_d^i) where $i = 1, 2, \dots, 23$. Specifically, we search the best parameters among certain discrete ranges by replaying the recorded trajectories in real with different simulation parameters summarized in Table 4.1. We then finetune the pre-trained policy in IsaacGym with the best SysID parameters.

DeltaDynamics (Figure 4.2 c): We train a residual dynamics model $f_\theta^\Delta(s_t, a_t)$ to capture the discrepancy between simulated and real-world physics.

Metrics. We report success rate, deeming imitation unsuccessful when, at any point during imitation, the average difference in body distance is on average further than 0.5m. We evaluate policy’s ability to imitate the reference motion by comparing the tracking error of the global body position $E_{g\text{-mpjpe}}$ (mm), the root-relative mean per-joint (MPJPE) E_{mpjpe} (mm), acceleration error E_{acc} (mm/frame²), and root velocity E_{vel} (mm/frame). The mean values of the metrics are computed across all motion sequences used.

4.1.1 Comparison of Dynamics Matching Capability

To address **Q1** (*Can **ASAP** outperform other baseline methods to compensate for the dynamics mismatch?*), we establish sim-to-sim transfer benchmarks to assess the ef-

4. Experiments

Table 4.2: Open-loop performance comparison across simulators and motion lengths.

Simulator & Length		IsaacSim				Genesis			
Length	Method	$E_{g\text{-mpjpe}}$	E_{mpjpe}	E_{acc}	E_{vel}	$E_{g\text{-mpjpe}}$	E_{mpjpe}	E_{acc}	E_{vel}
0.25s	OpenLoop	19.5	15.1	6.44	5.80	19.8	15.3	6.53	5.88
	SysID	19.4	15.0	6.43	5.74	19.3	15.0	6.42	5.73
	DeltaDynamics	24.4	13.6	9.43	7.85	20.0	12.4	8.42	6.89
	ASAP	19.9	15.6	6.48	5.86	19.0	14.9	6.19	5.59
0.5s	OpenLoop	33.3	23.2	6.80	6.84	33.1	23.0	6.78	6.82
	SysID	32.1	22.2	6.57	6.56	32.2	22.3	6.57	6.57
	DeltaDynamics	36.5	16.4	8.89	7.98	27.8	14.0	7.63	6.74
	ASAP	26.8	19.2	5.09	5.36	25.9	18.4	4.93	5.19
1.0s	OpenLoop	80.8	43.5	10.6	11.1	82.5	44.5	10.8	11.4
	SysID	77.6	41.5	10.2	10.7	76.5	41.6	10.0	10.5
	DeltaDynamics	68.1	21.5	9.61	9.14	50.2	17.2	8.19	7.62
	ASAP	37.9	22.9	4.38	5.26	36.9	22.6	4.23	5.10

Table 4.3: Closed-loop motion imitation evaluation across different simulators. All variants are trained with identical rewards.

Test Environment		IsaacSim					Genesis				
Level	Method	Succ $\uparrow$	$E_{g\text{-mpjpe}}$ $\downarrow$	E_{mpjpe} $\downarrow$	E_{acc} $\downarrow$	E_{vel} $\downarrow$	Succ $\uparrow$	$E_{g\text{-mpjpe}}$ $\downarrow$	E_{mpjpe} $\downarrow$	E_{acc} $\downarrow$	E_{vel} $\downarrow$
Easy	Oracle (IsaacGym $\rightarrow$ IsaacGym)	100% $_{\pm 0.000\%}$	97.5 ± 0.605	43.2 ± 0.112	2.56 ± 0.024	4.48 ± 0.023	100% $_{\pm 0.000\%}$	97.5 ± 0.605	43.2 ± 0.112	2.56 ± 0.024	4.48 ± 0.023
	Vanilla (IsaacGym $\rightarrow$ TestEnv)	100% $_{\pm 0.000\%}$	107 ± 0.578	45.4 ± 0.169	2.83 ± 0.012	4.59 ± 0.021	100% $_{\pm 0.000\%}$	140 ± 1.85	70.1 ± 0.626	2.68 ± 0.042	4.65 ± 0.046
	SysID	100% $_{\pm 0.000\%}$	105 ± 1.35	47.8 ± 0.970	3.09 ± 0.011	4.98 ± 0.020	100% $_{\pm 0.000\%}$	127 ± 0.233	79.9 ± 0.330	2.99 ± 0.035	4.95 ± 0.012
	DeltaDynamics	100% $_{\pm 0.000\%}$	127 ± 2.97	56.7 ± 0.390	3.50 ± 0.028	5.56 ± 0.031	83.3% $_{\pm 0.000\%}$	168 ± 7.62	87.0 ± 1.51	3.08 ± 0.18	5.39 ± 0.34
	ASAP	100% $_{\pm 0.000\%}$	106 ± 0.498	44.3 ± 0.103	2.74 ± 0.025	4.46 ± 0.020	100% $_{\pm 0.000\%}$	125 ± 4.75	73.5 ± 0.570	2.10 ± 0.083	4.11 ± 0.133
Medium	Oracle (IsaacGym $\rightarrow$ IsaacGym)	100% $_{\pm 0.000\%}$	111 ± 0.635	48.8 ± 0.133	2.63 ± 0.017	4.82 ± 0.019	100% $_{\pm 0.000\%}$	111 ± 0.635	48.8 ± 0.133	2.63 ± 0.017	4.82 ± 0.019
	Vanilla (IsaacGym $\rightarrow$ TestEnv)	100% $_{\pm 0.000\%}$	114 ± 0.720	49.2 ± 0.104	2.92 ± 0.021	5.07 ± 0.016	94.3% $_{\pm 7.00\%}$	169 ± 5.76	72.0 ± 0.692	3.26 ± 0.076	5.86 ± 0.101
	SysID	100% $_{\pm 0.000\%}$	115 ± 1.256	49.1 ± 0.560	3.43 ± 0.021	5.01 ± 0.017	100% $_{\pm 0.000\%}$	138 ± 2.70	75.4 ± 1.18	3.14 ± 0.042	5.50 ± 0.058
	DeltaDynamics	83.3% $_{\pm 0.000\%}$	151 ± 2.62	68.0 ± 0.364	2.90 ± 0.047	5.90 ± 0.107	83.3% $_{\pm 0.000\%}$	190 ± 1.46	89.4 ± 0.50	3.44 ± 0.16	7.49 ± 0.11
	ASAP	100% $_{\pm 0.000\%}$	112 ± 1.648	49.3 ± 0.574	2.53 ± 0.019	4.45 ± 0.026	100% $_{\pm 0.000\%}$	126 ± 1.63	71.2 ± 0.163	2.81 ± 0.037	5.13 ± 0.066
Hard	Oracle (IsaacGym $\rightarrow$ IsaacGym)	100% $_{\pm 0.000\%}$	116 ± 0.711	52.5 ± 0.298	3.40 ± 0.027	6.16 ± 0.028	100% $_{\pm 0.000\%}$	116 ± 0.711	52.5 ± 0.298	3.40 ± 0.027	6.16 ± 0.028
	Vanilla (IsaacGym $\rightarrow$ TestEnv)	100% $_{\pm 0.000\%}$	148 ± 0.845	51.6 ± 0.137	4.41 ± 0.055	6.88 ± 0.064	82.9% $_{\pm 5.70\%}$	175 ± 9.77	80.7 ± 1.69	3.87 ± 0.175	7.19 ± 0.199
	SysID	100% $_{\pm 0.000\%}$	165 ± 3.83	58.4 ± 0.229	4.87 ± 0.197	7.13 ± 0.131	100% $_{\pm 0.000\%}$	186 ± 3.84	93.0 ± 1.49	4.98 ± 0.245	8.98 ± 0.119
	DeltaDynamics	66.7% $_{\pm 0.000\%}$	137 ± 2.59	60.2 ± 0.477	4.20 ± 0.041	7.10 ± 0.024	60.0% $_{\pm 5.70\%}$	190 ± 14.0	89.6 ± 9.34	4.29 ± 1.16	8.70 ± 2.33
	ASAP	100% $_{\pm 0.000\%}$	129 ± 1.57	56.5 ± 1.15	3.72 ± 0.036	6.52 ± 0.042	100% $_{\pm 0.000\%}$	129 ± 2.31	77.0 ± 1.07	2.69 ± 0.040	5.65 ± 0.073

fectiveness of different methods in bridging the dynamics gap. IsaacGym serves as the *training environment*, while IsaacSim and Genesis function as *testing environments*. The primary objective is to evaluate the generalization capability of each approach when exposed to new dynamics conditions. *Open-loop* evaluation measures how accurately a method can reproduce testing-environment trajectories in the training environment. This is achieved by rolling out the same trajectory executed in the testing environment and assessing tracking discrepancies using key metrics such as MPJPE. An ideal method should minimize the discrepancies between training and testing trajectories when replaying testing-environment actions, thereby demonstrating an improved capacity for compensating dynamics mismatch. Quantitative results in Table 4.2 demonstrate that ASAP consistently outperforms the OpenLoop baseline across all replayed motion lengths, achieving lower $E_{g\text{-mpjpe}}$ and E_{mpjpe} values, which

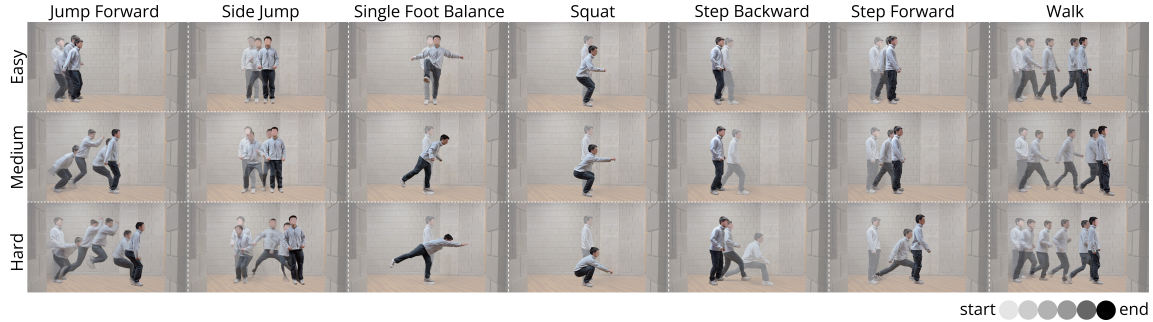


Figure 4.3: Visual comparisons of motion imitation results across different difficulty levels (Easy, Medium, Hard) for various tasks including Jump Forward, Side Jump, Single Foot Balance, Squat, Step Backward, Step Forward, and Walk.

indicate improved alignment with testing-env trajectories. While SysID helps address short-horizon dynamics gaps, it struggles with long-horizon scenarios due to cumulative error buildup. DeltaDynamics improves upon both SysID and OpenLoop for long horizons but suffers from overfitting, as evidenced by cascading errors magnified over time, as shown in Figure 4.1. ASAP, however, demonstrates superior generalization by learning residual policies that effectively bridge the dynamics gap. Comparable trends are observed in the Genesis simulator, where ASAP achieves notable improvements across all metrics relative to the baseline. These results emphasize the efficacy of learning delta action model to reduce the physics gap and improve open-loop replay performance.

4.1.2 Comparison of Policy Fine-Tuning Performance

To address **Q2** (*Can ASAP finetune policy to outperform SysID and Delta Dynamics methods?*), we evaluate the effectiveness of different methods in fine-tuning RL policies for improved testing-environment performance. We fine-tune RL policies in modified training environments and subsequently deploy them in the testing environments, quantifying motion-tracking errors in testing environments. As shown in Table 4.3, ASAP consistently outperforms baselines such as Vanilla, SysID, and DeltaDynamics across all difficulty levels (Easy, Medium, and Hard) in both simulators (IsaacSim and Genesis). For the Easy level, our method achieves the lowest $E_{g\text{-mpjpe}}$ and E_{mpjpe} in IsaacSim ($E_{g\text{-mpjpe}} = 106$ and $E_{\text{mpjpe}} = 44.3$) and Genesis ($E_{g\text{-mpjpe}} = 125$ and $E_{\text{mpjpe}} = 73.5$), with minimal acceleration (E_{acc}) and velocity (E_{vel}) errors. In more

4. Experiments

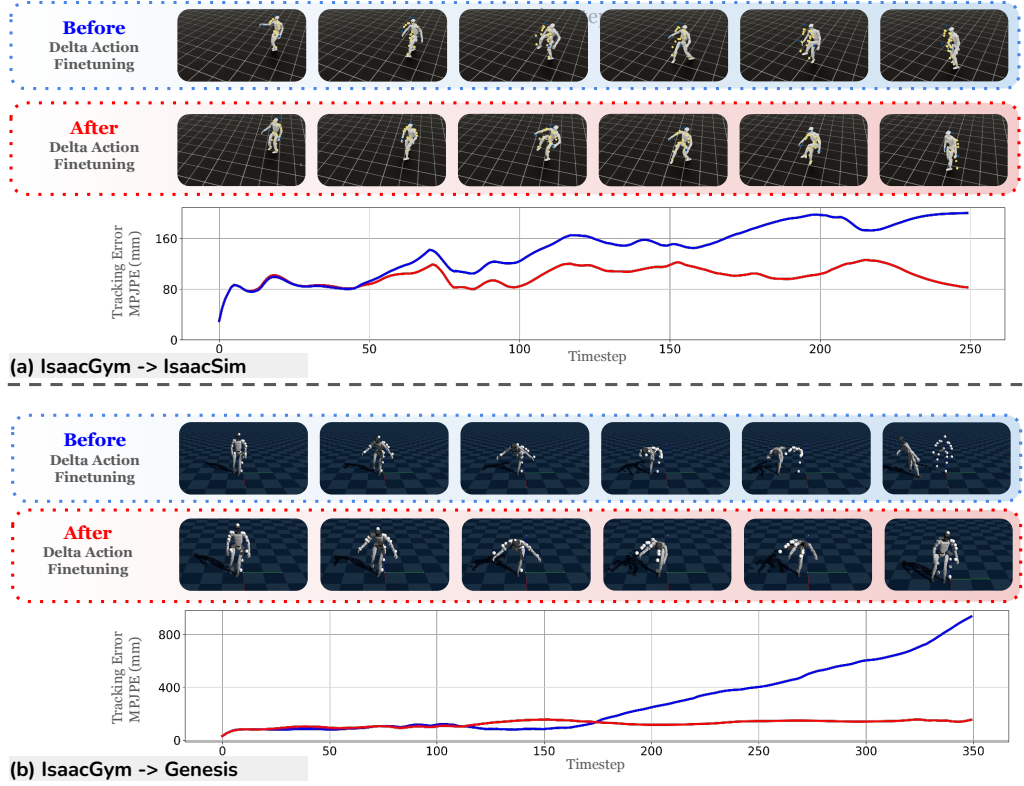


Figure 4.4: Visualization of G1 motion tracking before and after **ASAP** fine-tuning in IsaacGym, IsaacSim and Genesis. Top: LeBron James’ “Silencer” motion tracking policy fine-tuning for IsaacGym to IsaacSim. Bottom: *single foot balance* motion tracking policy fine-tuning for IsaacGym to Genesis.

challenging tasks, such as the Hard level, our method continues to excel, significantly reducing motion-tracking errors. For instance, in Genesis, it achieves $E_{g\text{-mpjpe}} = 129$ and $E_{\text{mpjpe}} = 77.0$, outperforming SysID and DeltaDynamics by substantial margins. Additionally, our method consistently maintains a 100% success rate across both simulators, unlike DeltaDynamics, which experiences lower success rates in harder environments. To further illustrate the advantages of **ASAP**, we provide per-step visualizations in Figure 4.4, comparing **ASAP** with RL policies deployed without fine-tuning. These visualizations demonstrate that **ASAP** successfully adapts to new dynamics and maintains stable tracking performance, whereas baseline methods accumulate errors over time, leading to degraded tracking capability. These results highlight the robustness and adaptability of our approach in addressing the sim-to-real gap while preventing overfitting and exploitation. The findings validate that **ASAP**

is an effective paradigm for improving closed-loop performance and ensuring reliable deployment in complex real-world scenarios.

4.1.3 Real-World Evaluations

To answer **Q3** (*Does ASAP work for sim-to-real transfer?*). We validate **ASAP** on real-world Unitree G1 robot.

Real-World Data. In the real-world experiments, we prioritize both motion safety and representativeness by selecting five motion-tracking tasks, including (i) *kick*, (ii) *jump forward*, (iii) *step forward and back*, (iv) *single foot balance* and (v) *single foot jump*. However, collecting over 400 real-world motion clips—the minimum required to train the full 23-DoF delta action model in simulation, as discussed in [Section 3.2.2](#)—poses significant challenges. Our experiments involve highly dynamic motions that cause rapid overheating of joint motors, leading to hardware failures (two Unitree G1 robots broke during data collection). Given these constraints, we adopt a more sample-efficient approach by focusing exclusively on learning a 4-DoF ankle delta action model rather than the full-body 23-DoF model. This decision is motivated by two key factors: (1) the limited availability of real-world data makes training the full 23-DoF delta action model infeasible, and (2) the Unitree G1 robot [77] features a mechanical linkage design in the ankle, which introduces a significant sim-to-real gap that is difficult to bridge with conventional modeling techniques [37]. Under this setting, the original 23 DoF delta action model reduces to 4 DoF delta action model, which needs much less data to be trainable. In practice, we collect 100 motion clips, which prove sufficient to train an effective 4-DoF delta action model for real-world scenarios.

We execute the tracking policy 30 times for each task. In addition to these motion-tracking tasks, we also collect 10 minutes of locomotion data. The locomotion policy will be addressed in the next section, which is also utilized to bridge different tracking policies.

Policy Transition. In the real world, we cannot easily reset the robot as in simulators, and therefore we train a robust locomotion policy for the policy transition between different motion-tracking tasks. Our locomotion command contains (v, ω, Π) , where v and ω indicate the linear and angular velocities while Π indicates the command

4. Experiments

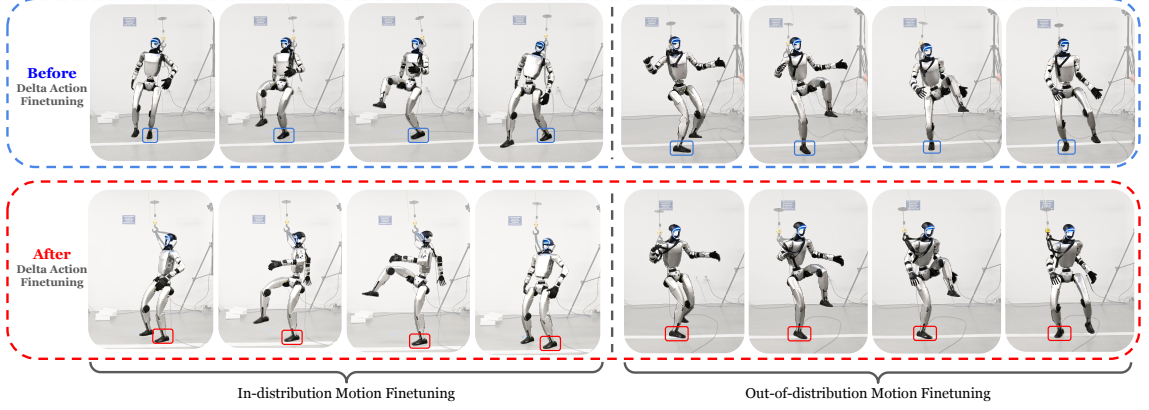


Figure 4.5: Visualization of LeBron James’ “Silencer” motion on the G1 robot before (upper figure enclosed in blue) and after (bottom figure enclosed in red) ASAP policy finetuning. The left half shows the policy finetuning for the in-distribution motions while the right half shows the out-of-distribution ones. After ASAP finetuning, the robot behaves more smoothly and reduces jerky lower-body motions.

to walk or stand still. After each motion-tracking task is done, our locomotion policy will take over to keep the robot balance until the next motion-tracking task begins. In this way, the robot is able to execute multiple tasks without manually resetting.

Real-World Results. The sim-to-real gap is more pronounced than simulator-to-simulator discrepancies due to factors such as noisy sensor input, inaccuracies in robot modeling, and actuator differences. To evaluate the effectiveness of ASAP in addressing these gaps, we compare the closed-loop performance of ASAP with the *Vanilla* baseline on two representative motion tracking tasks (kicking and “Silencer”) in which observe obvious sim-to-real gaps. To show the generalizability of the learned delta action model for out-of-distribution motions, we also fine-tune the policy for LeBron James’ “Silencer” motion as shown in Figure 1.2 and Figure 4.5. The experiment data is summarized in Table 4.4. It shows that ASAP outperforms the baseline on both in-distribution and out-of-distribution humanoid motion tracking tasks, achieving a considerable reduction of the tracking errors across all key metrics ($E_{g\text{-mpjpe}}$, E_{mpjpe} , E_{acc} and E_{vel}). These findings highlight the effectiveness of ASAP in improving sim-to-real transfer for agile humanoid motion tracking.

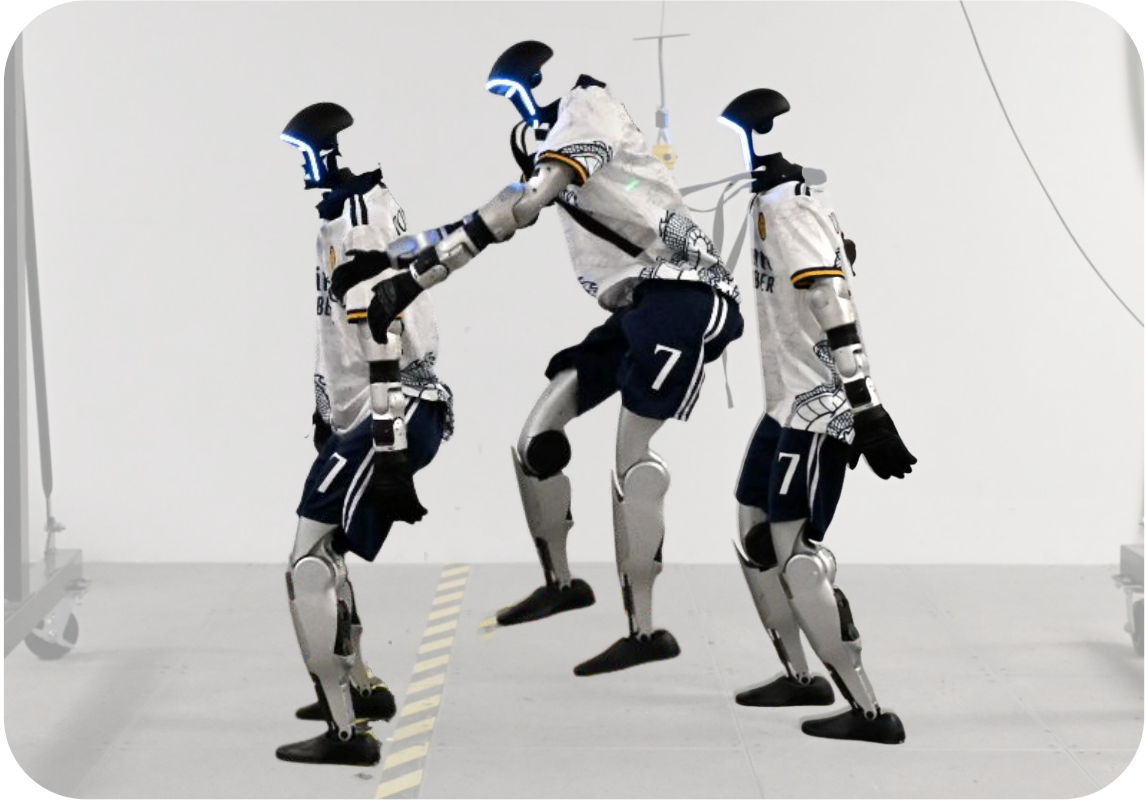


Figure 4.6: We deploy the pretrained policy of a forward jump motion tracking task, challenging the 1.35m-tall Unitree G1 robot for a forward leap over 1m.

4.2 Extensive Studies and Analyses

In this section, we aim to thoroughly analyze **ASAP** by addressing three central research questions:

- **Q4**: How to best train the delta action model of **ASAP**?
- **Q5**: How to best use the delta action model of **ASAP**?
- **Q6**: Why and how does **ASAP** work?

4.2.1 Key Factors in Training Delta Action Models

To Answer **Q4** (*How to best train the delta action model of **ASAP***), we conduct a systematic study on key factors influencing the performance of the delta action model. Specifically, we investigate the impact of dataset size, training horizon, and action

4. Experiments

Table 4.4: Real-world closed-loop performance comparing with and without **ASAP** finetuning on one in-distribution motion and one out-of-distribution motion.

Motion	Real-World-Kick				Real-World-LeBron (OOD)			
Method	$E_{g\text{-mpjpe}}$	E_{mpjpe}	E_{acc}	E_{vel}	$E_{g\text{-mpjpe}}$	E_{mpjpe}	E_{acc}	E_{vel}
Vanilla	61.2	43.5	2.96	2.91	159	55.3	3.43	6.43
ASAP	50.2	40.1	2.46	2.70	112	47.5	2.84	5.94

norm weight, evaluating their effects on both open-loop and closed-loop performance. Our analysis uncovers the essential principles for effectively training a high-performing delta action model.

Dataset Size We analyze the impact of dataset size on the training and generalization of π^Δ . Simulation data is collected in Isaac Sim, and π^Δ is trained in Isaac Gym. Open-loop performance is assessed on both in-distribution (training) and out-of-distribution (unseen) trajectories, while closed-loop performance is evaluated using the fine-tuned policy in Isaac Sim. As shown in Figure 4.7 (a), increasing the dataset size improves π^Δ 's generalization, evidenced by reduced errors in out-of-distribution evaluations. However, the improvement in closed-loop performance saturates, with a marginal decrease of only 0.65% when scaling from 4300 to 43000 samples, suggesting limited additional benefit from larger datasets.

Training Horizon The rollout horizon plays a crucial role in learning π^Δ . As shown in Figure 4.7 (b), longer training horizons generally improve open-loop performance, with a horizon of 1.5s achieving the lowest errors across evaluation points at 0.25s, 0.5s, and 1.0s. However, this trend does not consistently extend to closed-loop performance. The best closed-loop results are observed at a training horizon of 1.0s, indicating that excessively long horizons do not provide additional benefits for fine-tuned policy.

Action Norm Weight Training π^Δ incorporates an action norm reward to balance dynamics alignment and minimal correction. As illustrated in Figure 4.7 (c), both open-loop and closed-loop errors decrease as the action norm weight increases, reaching the lowest error at a weight of 0.1. However, further increasing the action norm weight causes open-loop errors to rise, likely due to the minimal action norm reward dominates in the delta action RL training. This highlights the importance of carefully

tuning the action norm weight to achieve optimal performance.

4.2.2 Different Usage of Delta Action Model

To answer **Q5** (How to best use the delta action model of **ASAP**?), we compare multiple strategies: fixed-point iteration, gradient-based optimization, and reinforcement learning (RL). Given a learned delta policy π^Δ such that:

$$f^{\text{sim}}(s, a + \pi^\Delta(s, a)) \approx f^{\text{real}}(s, a),$$

and a nominal policy $\hat{\pi}(s)$ that performs well in simulation, the goal is to fine-tune $\hat{\pi}(s)$ for real-world deployment.

A simple approach is one-step dynamics matching, which leads to the relationship:

$$\pi(s) = \hat{\pi}(s) - \pi^\Delta(s, \pi(s)).$$

We consider two RL-free methods: fixed-point iteration and gradient-based optimization. Fixed-point iteration refines $\hat{\pi}(s)$ iteratively, while gradient-based optimization minimizes a loss function to achieve a better estimate. These methods are compared against RL fine-tuning, which adapts $\hat{\pi}(s)$ using reinforcement learning in simulation.

Our experiments in [Figure 4.8](#) show that RL fine-tuning achieves the lowest tracking error during deployment, outperforming training-free methods. Both RL-free approaches are myopic and suffer from out-of-distribution issues, limiting their real-world applicability.

4.2.3 Does **ASAP** Fine-Tuning Outperform Random Action Noise Fine-Tuning?

To answer **Q6** (*How does **ASAP** work?*), we validate **ASAP** finetuning is better than injecting random-action-noise-based finetuning. And we visualize the average magnitude of the delta action model for each joint.

Random torque noise [7] is a widely used domain randomization technique for legged robots. To determine whether delta action facilitates fine-tuning of pre-trained policies toward real-world dynamics rather than merely enhancing robustness through

4. Experiments

random action noise, we analyze its impact. Specifically, we assess the effect of applying random action noise during policy fine-tuning in Isaac Gym by modifying the environment dynamics as $s_{t+1} = f^{\text{sim}}(s_t, a_t + \beta\delta_a)$, where $\delta_a \sim \mathcal{U}[0, 1]$, and deploy it in Genesis. We conduct an ablation study to examine the influence of the noise magnitude, β , varying from 0.025 to 0.4. As shown in Figure 4.9, within the constrained range of $\beta \in [0.025, 0.2]$, policies fine-tuned with action noise outperform those without fine-tuning in terms of global tracking error (MPJPE). However, the performance of the action noise approach (MPJPE of 150) does not match the precision achieved by ASAP (MPJPE of 126). Furthermore, we visualize the average output of π^Δ learned from IsaacSim data in Figure 4.10, which reveals non-uniform discrepancies across joints. For example, in the G1 humanoid robot under our experimental setup, lower-body motors exhibit a larger dynamics gap compared to upper-body joints. Within the lower-body, the ankle and knee joints show the most pronounced discrepancies. Additionally, asymmetries between the left and right body motors further highlight the complexity. Such structured discrepancies cannot be effectively captured by merely adding uniform action noise. These findings, along with the results in Figure 4.1, demonstrate that delta action not only enhances policy robustness but also enables effective adaptation to real-world dynamics, outperforming naive randomization strategies.

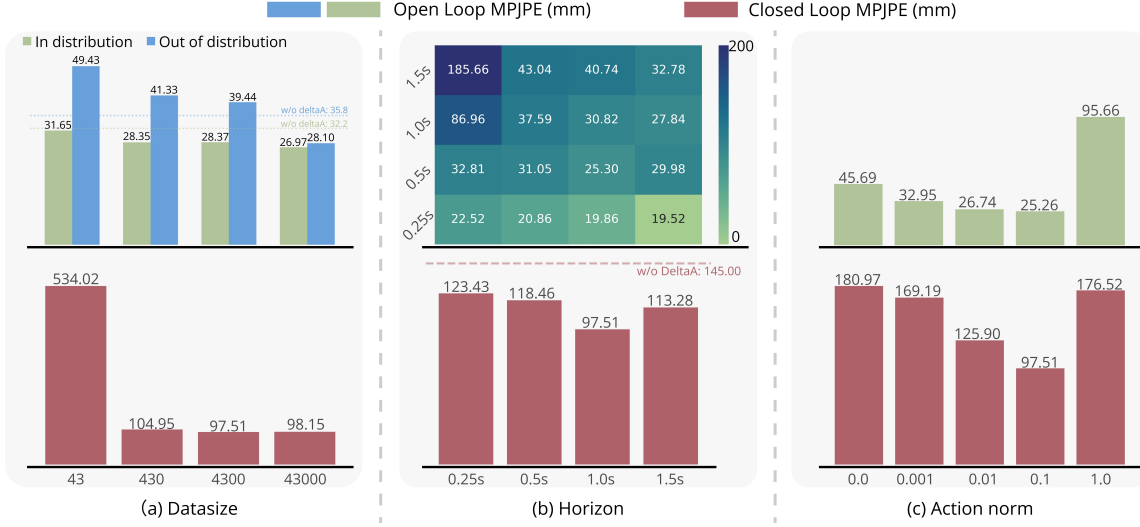


Figure 4.7: Analysis of dataset size, training horizon, and action norm on the performance of π^Δ . (a) **Dataset Size**: Mean Per Joint Position Error (MPJPE) is evaluated for both in-distribution (green) and out-of-distribution (blue) scenarios. Increasing dataset size leads to enhanced generalization, evidenced by decreasing errors in out-of-distribution evaluations. Closed-loop MPJPE (red bars) also shows improvement with larger datasets. (b) **Training Horizon**: Open-loop MPJPE (heatmap) improves across evaluation points as training horizons increase, achieving the lowest error at 1.5s. However, closed-loop MPJPE (red bars) shows a sweet spot at a training horizon of 1.0s, beyond which no further improvements are observed. The red dashed line represents the pretrained baseline without π^Δ fine-tuning. (c) **Action Norm**: The action norm weight significantly influences performance. Both open-loop and closed-loop MPJPE decrease as the weight increases up to 0.1, achieving the lowest error. However, further increases in the action norm weight result in degradation of open-loop performance, highlighting the trade-off between action smoothness and policy flexibility.

4. Experiments

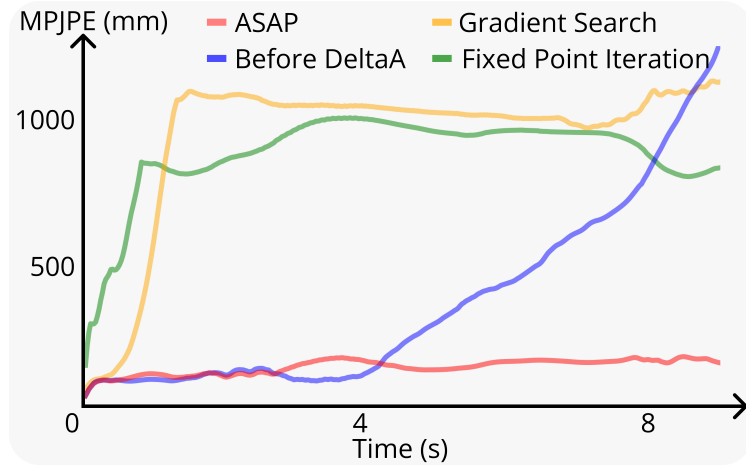


Figure 4.8: MPJPE comparison over timesteps for fine-tuning methods using delta actionmodel. RL Fine-Tuning achieves the lowest error, while Fixed-Point Iteration and Gradient Search perform worse than the baseline (Before DeltaA) showing the highest error.

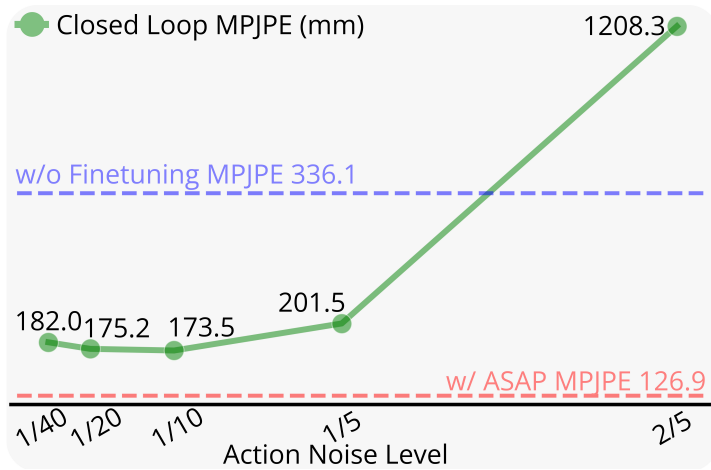


Figure 4.9: MPJPE vs. Noise Level for policies fine-tuned with random action noise. Policies with noise levels $\beta \in [0.025, 0.2]$ show improved performance compared to no fine-tuning. Delta action achieves better tracking precision (126 MPJPE) compared to the best action noise (173 MPJPE).

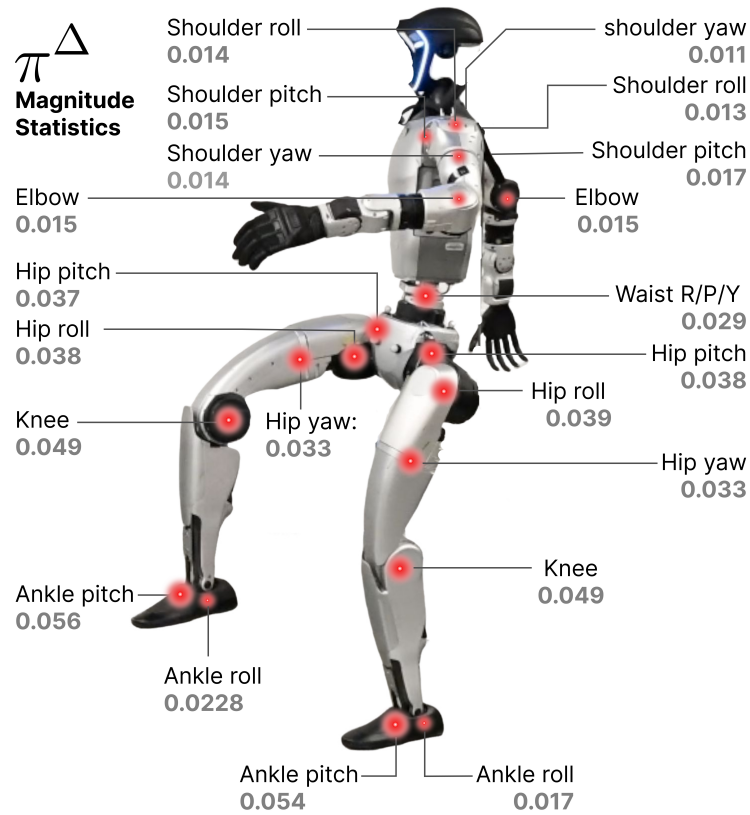


Figure 4.10: Visualization of IsaacGym-to-IsaacSim π^Δ output magnitude. We compute the average absolute value of each joint over the 4300-episode dataset. Larger red dots indicate higher values. The results suggest that lower-body motors exhibit a larger discrepancy compared to upper-body joints, with the most significant gap observed in the ankle pitch joint of the G1 humanoid.

4. *Experiments*

Chapter 5

Conclusions

While **ASAP** demonstrates promising results in bridging the sim-to-real gap for agile humanoid control, our framework has several real-world limitations that highlights critical challenges in scaling agile humanoid control to real-world:

- **Hardware Constraints:** Agile whole-body motions exert significant stress on robots, leading to motor overheating and hardware failure during data collection. Two Unitree G1 robots were broken to some extent during our experiments. This bottleneck limits the scale and diversity of real-world motion sequences that can be safely collected.
- **Dependence on Motion Capture Systems:** Our pipeline requires MoCap setup to record real-world trajectories. This introduces practical deployment barriers in unstructured environments where MoCap setups are unavailable.
- **Data-Hungry Delta Action Training:** While reducing the delta action model to 4 DoF ankle joints improved sample efficiency, training the full 23 DoF model remains impractical for real-world deployment due to the large demand of required motion clips (e.g., > 400 episodes in simulation for the 23 DoF delta action training).

Future directions could focus on developing damage-aware policy to mitigate hardware risks, leveraging MoCap-free alignment to eliminate the reliance on MoCap, and exploring adaptation techniques for delta action models to achieve sample-efficient few-shot alignment.

5. *Conclusions*

Bibliography

- [1] Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Residual reinforcement learning from demonstrations. *arXiv preprint arXiv:2106.08050*, 2021. 2.3
- [2] Chae H An, Christopher G Atkeson, and John M Hollerbach. Estimation of inertial parameters of rigid body links of manipulators. In *1985 24th IEEE Conference on Decision and Control*, pages 990–995. IEEE, 1985. 2.2
- [3] Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement–residual rl for precise assembly. *arXiv preprint arXiv:2407.16677*, 2024. 2.3
- [4] Rika Antonova, Jingyun Yang, Priya Sundaresan, Dieter Fox, Fabio Ramos, and Jeannette Bohg. A bayesian treatment of real-to-sim for deformable object manipulation. *IEEE Robotics and Automation Letters*, 7(3):5819–5826, 2022. 2.2
- [5] Karl Johan Åström and Peter Eykhoff. System identification—a survey. *Automatica*, 7(2):123–162, 1971. 2.2
- [6] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond, December 2024. URL <https://github.com/Genesis-Embodied-AI/Genesis>. 4.1
- [7] Luigi Campanaro, Siddhant Gangapurwala, Wolfgang Merkt, and Ioannis Havoutis. Learning and deploying robust locomotion policies with minimal dynamics randomization. In *6th Annual Learning for Dynamics & Control Conference*, pages 578–590. PMLR, 2024. 4.2.3
- [8] Joao Carvalho, Dorothea Koert, Marek Daniv, and Jan Peters. Residual robot learning for object-centric probabilistic movement primitives. *arXiv preprint arXiv:2203.03918*, 2022. 2.3
- [9] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

2.2

- [10] Xuxin Cheng, Yandong Ji, Junming Chen, Ruihan Yang, Ge Yang, and Xiaolong Wang. Expressive whole-body control for humanoid robots. *arXiv preprint arXiv:2402.16796*, 2024. [1](#), [2.1](#)
- [11] Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects. *The International Journal of Robotics Research*, 43(4): 389–404, 2024. [2.3](#)
- [12] Todor Davchev, Kevin Sebastian Luck, Michael Burke, Franziska Meier, Stefan Schaal, and Subramanian Ramamoorthy. Residual learning from demonstration: Adapting dmps for contact-rich manipulation. *IEEE Robotics and Automation Letters*, 7(2):4488–4495, 2022. [2.3](#)
- [13] Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1290–1296. IEEE, 2021. [2.2](#)
- [14] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, pages 138–149. PMLR, 2023. [2.2](#)
- [15] Zipeng Fu, Qingqing Zhao, Qi Wu, Gordon Wetzstein, and Chelsea Finn. Humanplus: Humanoid shadowing and imitation from humans. *arXiv preprint arXiv:2406.10454*, 2024. [2.1](#)
- [16] Feng Gao, Chao Yu, Yu Wang, and Yi Wu. Neural internal model control: Learning a robust control policy via predictive error feedback. *arXiv preprint arXiv:2411.13079*, 2024. [2.2](#)
- [17] Jiawei Gao, Ziqin Wang, Zeqi Xiao, Jingbo Wang, Tai Wang, Jinkun Cao, Xiaolin Hu, Si Liu, Jifeng Dai, and Jiangmiao Pang. Coohoi: Learning cooperative human-object interaction with manipulated object dynamics. *arXiv preprint arXiv:2406.14558*, 2024. [2.1](#)
- [18] Maxime Gautier, Pierre-Olivier Vandanjon, and Alexandre Janot. Dynamic identification of a 6 dof robot without joint position data. In *2011 IEEE International Conference on Robotics and Automation*, pages 234–239. IEEE, 2011. [2.2](#)
- [19] Xinyang Gu, Yen-Jen Wang, Xiang Zhu, Chengming Shi, Yanjiang Guo, Yichen Liu, and Jianyu Chen. Advancing humanoid locomotion: Mastering challenging terrains with denoising world model learning. *arXiv preprint arXiv:2408.14472*, 2024. [1](#), [2.1](#), [2.2](#)
- [20] Siddhant Halder, Jyothish Pari, Anant Rai, and Lerrel Pinto. Teach a robot

- to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023. [2.3](#)
- [21] Yong Han, Jianhua Wu, Chao Liu, and Zhenhua Xiong. An iterative approach for accurate dynamic model identification of industrial robots. *IEEE Transactions on Robotics*, 36(5):1577–1594, 2020. [2.2](#)
- [22] Mohamed Hassan, Yunrong Guo, Tingwu Wang, Michael Black, Sanja Fidler, and Xue Bin Peng. Synthesizing physical character-scene interactions. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–9, 2023. [2.1](#)
- [23] Guanqi He, Yogita Choudhary, and Guanya Shi. Self-supervised meta-learning for all-layer dnn-based adaptive control with stability guarantees. *arXiv preprint arXiv:2410.07575*, 2024. [2.3](#)
- [24] Tairan He, Zhengyi Luo, Xialin He, Wenli Xiao, Chong Zhang, Weinan Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning. *arXiv preprint arXiv:2406.08858*, 2024. [1](#), [2.1](#), [3.1.2](#)
- [25] Tairan He, Zhengyi Luo, Wenli Xiao, Chong Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Learning human-to-humanoid real-time whole-body teleoperation. *arXiv preprint arXiv:2403.04436*, 2024. [1](#), [2.1](#), [3.1.1](#), [3.1.1](#), [3.1.2](#)
- [26] Tairan He, Wenli Xiao, Toru Lin, Zhengyi Luo, Zhenjia Xu, Zhenyu Jiang, Jan Kautz, Changliu Liu, Guanya Shi, Xiaolong Wang, et al. Hover: Versatile neural whole-body controller for humanoid robots. *arXiv preprint arXiv:2410.21229*, 2024. [1](#)
- [27] Eric Heiden, Christopher E Denniston, David Millard, Fabio Ramos, and Gaurav S Sukhatme. Probabilistic inference of simulation parameters via parallel differentiable simulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3638–3645. IEEE, 2022. [2.2](#)
- [28] Kevin Huang, Rwik Rana, Alexander Spitzer, Guanya Shi, and Byron Boots. Datt: Deep adaptive trajectory tracking for quadrotor control. *arXiv preprint arXiv:2310.09053*, 2023. [2.2](#)
- [29] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019. [1](#), [2.2](#)
- [30] Alexandre Janot, Pierre-Olivier Vandanjon, and Maxime Gautier. A generic instrumental variable approach for industrial robot identification. *IEEE Transactions on Control Systems Technology*, 22(1):132–145, 2013. [2.2](#)
- [31] Gwanghyeon Ji, Juhyeok Mun, Hyeongjun Kim, and Jemin Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust

- legged locomotion. *IEEE Robotics and Automation Letters*, 7(2):4630–4637, 2022. 2.2
- [32] Mazeyu Ji, Xuanbin Peng, Fangchen Liu, Jialong Li, Ge Yang, Xuxin Cheng, and Xiaolong Wang. Exbody2: Advanced expressive humanoid whole-body control. *arXiv preprint arXiv:2412.13196*, 2024. 1, 2.1
- [33] Yunfan Jiang, Chen Wang, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. *arXiv preprint arXiv:2405.10315*, 2024. 2.3
- [34] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pages 6023–6029. IEEE, 2019. 2.3
- [35] Haresh Karnan, Siddharth Desai, Josiah P Hanna, Garrett Warnell, and Peter Stone. Reinforced grounded action transformation for sim-to-real transfer. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4397–4402. IEEE, 2020. 2.3
- [36] Pradeep K Khosla and Takeo Kanade. Parameter identification of robot dynamics. In *1985 24th IEEE conference on decision and control*, pages 1754–1760. IEEE, 1985. 2.2
- [37] Donghyun Kim, Steven Jens Jorgensen, Jaemin Lee, Junhyeok Ahn, Jianwen Luo, and Luis Sentis. Dynamic locomotion for passive-ankle biped robots and humanoids using whole-body locomotion control. *Int. J. Robotics Res.*, 39(8), 2020. doi: 10.1177/0278364920918014. URL <https://doi.org/10.1177/0278364920918014>. 4.1.3
- [38] Alina Kloss, Stefan Schaal, and Jeannette Bohg. Combining learned and analytical models for predicting action effects from sensory data. *The International Journal of Robotics Research*, 41(8):778–797, 2022. 2.3
- [39] F Kozin and HG Natke. System identification techniques. *Structural safety*, 3(3-4):269–316, 1986. 2.2
- [40] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021. 2.2
- [41] Ashish Kumar, Zhongyu Li, Jun Zeng, Deepak Pathak, Koushil Sreenath, and Jitendra Malik. Adapting rapid motor adaptation for bipedal robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1161–1168. IEEE, 2022. 2.2
- [42] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco

- Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020. [2.2](#), [2.3](#)
- [43] Kuang-Huei Lee, Ofir Nachum, Tingnan Zhang, Sergio Guadarrama, Jie Tan, and Wenhao Yu. Pi-ars: Accelerating evolution-learned visual-locomotion with predictive information representations. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1447–1454. IEEE, 2022. [2.2](#)
- [44] Tianyu Li, Hartmut Geyer, Christopher G Atkeson, and Akshara Rai. Using deep reinforcement learning to learn high-level policies on the atrias biped. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 263–269. IEEE, 2019. [2.1](#)
- [45] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2811–2817. IEEE, 2021. [2.1](#)
- [46] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Robust and versatile bipedal jumping control through reinforcement learning. *arXiv preprint arXiv:2302.09450*, 2023. [1](#), [2.1](#)
- [47] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *The International Journal of Robotics Research*, page 02783649241285161, 2024. [1](#), [2.1](#), [2.2](#)
- [48] Qiayuan Liao, Bike Zhang, Xuanyu Huang, Xiaoyu Huang, Zhongyu Li, and Koushil Sreenath. Berkeley humanoid: A research platform for learning-based control. *arXiv preprint arXiv:2407.21781*, 2024. [2.1](#)
- [49] Lennart Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer. [1](#)
- [50] Junfeng Long, Junli Ren, Moji Shi, Zirui Wang, Tao Huang, Ping Luo, and Jiangmiao Pang. Learning humanoid locomotion with perceptive internal model. *arXiv preprint arXiv:2411.14386*, 2024. [1](#), [2.1](#)
- [51] Junfeng Long, Zirui Wang, Quanyi Li, Liu Cao, Jiawei Gao, and Jiangmiao Pang. Hybrid internal model: Learning agile legged locomotion with simulated robot response. In *The Twelfth International Conference on Learning Representations*, 2024. [2.2](#)
- [52] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 851–866. 2023. [3.1.1](#)

- [53] Chenhao Lu, Xuxin Cheng, Jialong Li, Shiqi Yang, Mazeyu Ji, Chengjing Yuan, Ge Yang, Sha Yi, and Xiaolong Wang. Mobile-television: Predictive motion priors for humanoid whole-body control. *arXiv preprint arXiv:2412.07773*, 2024. [2.1](#)
- [54] Shixin Luo, Songbo Li, Ruiqi Yu, Zhicheng Wang, Jun Wu, and Qiuguo Zhu. Pie: Parkour with implicit-explicit learning framework for legged robots. *IEEE Robotics and Automation Letters*, 2024. [2.2](#)
- [55] Zhengyi Luo, Jinkun Cao, Kris Kitani, Weipeng Xu, et al. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10895–10904, 2023. [2.1](#)
- [56] Zhengyi Luo, Jiashun Wang, Kangni Liu, Haotian Zhang, Chen Tessler, Jingbo Wang, Ye Yuan, Jinkun Cao, Zihui Lin, Fengyi Wang, et al. Smpolympics: Sports environments for physically simulated humanoids. *arXiv preprint arXiv:2407.00187*, 2024. [2.1](#)
- [57] Shangke Lyu, Xin Lang, Han Zhao, Hongyin Zhang, Pengxiang Ding, and Donglin Wang. Rl2ac: Reinforcement learning-based rapid online adaptive control for legged robot robust locomotion. In *Proceedings of the Robotics: Science and Systems*, 2024. [2.2](#)
- [58] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu based physics simulation for robot learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [2.1](#), [3.1.1](#), [4.1](#)
- [59] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022. [1](#)
- [60] Gabriel B Margolis, Xiang Fu, Yandong Ji, and Pulkit Agrawal. Learning to see physical properties with active sensing motor policies. *arXiv preprint arXiv:2311.01405*, 2023. [2.2](#)
- [61] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 43(4):572–587, 2024. [2.2](#)
- [62] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science robotics*, 7(62):eabk2822, 2022. [2.2](#)
- [63] Mayank Mittal, Calvin Yu, Qinxu Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A

- unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi: 10.1109/LRA.2023.3270034. [2.1](#), [4.1](#)
- [64] Fabio Muratore, Christian Eilers, Michael Gienger, and Jan Peters. Data-efficient domain randomization with bayesian optimization. *IEEE Robotics and Automation Letters*, 6(2):911–918, 2021. [2.2](#)
- [65] I Made Aswin Nahrendra, Byeongho Yu, and Hyun Myung. Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5078–5084. IEEE, 2023. [2.2](#)
- [66] Michael O’Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural-fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66):eabm6597, 2022. [2.2](#), [2.3](#)
- [67] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018. [1](#), [2.1](#), [3.1.2](#), [3.1.2](#)
- [68] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018. [1](#)
- [69] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions On Graphics (TOG)*, 37(6):1–14, 2018. [2.1](#)
- [70] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020. [2.2](#)
- [71] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions On Graphics (TOG)*, 41(4):1–17, 2022. [2.1](#)
- [72] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, pages 1722–1732. PMLR, 2023. [2.2](#)
- [73] I Radosavovic, B Zhang, B Shi, J Rajasegaran, S Kamat, T Darrell, K Sreenath, and J Malik. Humanoid locomotion as next token prediction. arxiv. 2024. *arXiv preprint arXiv:2402.19469*, 2024. [1](#), [2.1](#)
- [74] Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and

- Koushil Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024. 1, 2.1
- [75] Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019. 2.2
- [76] Allen Z Ren, Hongkai Dai, Benjamin Burchfiel, and Anirudha Majumdar. Adaptsim: Task-driven simulation adaptation for sim-to-real transfer. *arXiv preprint arXiv:2302.04903*, 2023. 2.2
- [77] Unitree Robotics. Unitree g1 humanoid agent ai avatar, 2024. URL <https://www.unitree.com/g1>. 1, 4.1.3
- [78] Unitree Robotics. Unitree h1 the world’s first full-size motor drive humanoid robot flips on ground, 2024. URL <https://www.youtube.com/watch?v=V1LyWsiTgms>. 2.1
- [79] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022. 1
- [80] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2.1, 3.1.2, 4
- [81] Guanya Shi, Xichen Shi, Michael O’Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. In *2019 international conference on robotics and automation (icra)*, pages 9784–9790. IEEE, 2019. 1
- [82] Guanya Shi, Wolfgang Hönig, Xichen Shi, Yisong Yue, and Soon-Jo Chung. Neural-swarm2: Planning and control of heterogeneous multirotor swarms using learned interactions. *IEEE Transactions on Robotics*, 38(2):1063–1079, 2021. 2.3
- [83] Aditya Shirwatkar, Naman Saxena, Kishore Chandra, and Shishir Kolathaya. Pip-loco: A proprioceptive infinite horizon planning framework for quadrupedal robot locomotion. *arXiv preprint arXiv:2409.09441*, 2024. 2.2
- [84] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018. 2.3
- [85] 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. 1, 2.2
- [86] Chen Tessler, Yunrong Guo, Ofir Nabati, Gal Chechik, and Xue Bin Peng.

- Maskedmimic: Unified physics-based character control through masked motion inpainting. *ACM Transactions on Graphics (TOG)*, 43(6):1–21, 2024. [2.1](#), [3.1.1](#)
- [87] 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. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017. [1](#)
- [88] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109. [2.1](#)
- [89] Hongxi Wang, Haoxiang Luo, Wei Zhang, and Hua Chen. Cts: Concurrent teacher-student reinforcement learning for legged locomotion. *IEEE Robotics and Automation Letters*, 2024. [2.2](#)
- [90] Jiashun Wang, Jessica Hodgins, and Jungdam Won. Strategy and skill learning for physics-based table tennis animation. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. [2.1](#)
- [91] Yinhuai Wang, Jing Lin, Ailing Zeng, Zhengyi Luo, Jian Zhang, and Lei Zhang. Physhoi: Physics-based imitation of dynamic human-object interaction. *arXiv preprint arXiv:2312.04393*, 2023. [2.1](#)
- [92] Yinhuai Wang, Qihan Zhao, Runyi Yu, Ailing Zeng, Jing Lin, Zhengyi Luo, Hok Wai Tsui, Jiwen Yu, Xiu Li, Qifeng Chen, et al. Skillmimic: Learning reusable basketball skills from demonstrations. *arXiv preprint arXiv:2408.15270*, 2024. [2.1](#)
- [93] Yufu Wang, Ziyun Wang, Lingjie Liu, and Kostas Daniilidis. Tram: Global trajectory and motion of 3d humans from in-the-wild videos. In *European Conference on Computer Vision*, pages 467–487. Springer, 2025. ([document](#)), [3.1.1](#), [3.1](#)
- [94] Zhicheng Wang, Wandi Wei, Ruiqi Yu, Jun Wu, and Qiuguo Zhu. Toward understanding key estimation in learning robust humanoid locomotion. *arXiv preprint arXiv:2403.05868*, 2024. [2.2](#)
- [95] Lior Wolf. Ai for humanoid robotics - a lecture by mentee robotics’ ceo, prof. lior wolf. YouTube video, 2025. URL <https://www.youtube.com/watch?v=y1LG4YwUtoo>. Accessed: 2025-01-31. [1](#)
- [96] Peilin Wu, Weiji Xie, Jiahang Cao, Hang Lai, and Weinan Zhang. Loopsr: Looping sim-and-real for lifelong policy adaptation of legged robots. *arXiv preprint arXiv:2409.17992*, 2024. [2.2](#)
- [97] Wenli Xiao, Haoru Xue, Tony Tao, Dvij Kalaria, John M Dolan, and Guanya

- Shi. Anycar to anywhere: Learning universal dynamics model for agile and adaptive mobility. *arXiv preprint arXiv:2409.15783*, 2024. [1](#)
- [98] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonanthan Hurst, and Michiel Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In *Conference on Robot Learning*, pages 317–329. PMLR, 2020. [2.1](#)
- [99] Yuxiang Yang, Guanya Shi, Changyi Lin, Xiangyun Meng, Rosario Scalise, Mateo Guaman Castro, Wenhao Yu, Tingnan Zhang, Ding Zhao, Jie Tan, et al. Agile continuous jumping in discontinuous terrains. *arXiv preprint arXiv:2409.10923*, 2024. [2.2](#)
- [100] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017. [2.2](#)
- [101] Wenhao Yu, C Karen Liu, and Greg Turk. Policy transfer with strategy optimization. *arXiv preprint arXiv:1810.05751*, 2018. [2.2](#)
- [102] Wenhao Yu, Visak CV Kumar, Greg Turk, and C Karen Liu. Sim-to-real transfer for biped locomotion. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3503–3510. IEEE, 2019. [1](#), [2.2](#)
- [103] Wenhao Yu, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with meta strategy optimization. *IEEE Robotics and Automation Letters*, 5(2):2950–2957, 2020. [2.2](#)
- [104] YE YUAN and Viktor Makoviychuk. Learning physically simulated tennis skills from broadcast videos. 2023. [2.1](#)
- [105] Chong Zhang, Wenli Xiao, Tairan He, and Guanya Shi. Wococo: Learning whole-body humanoid control with sequential contacts. *arXiv preprint arXiv:2406.06005*, 2024. [2.1](#)
- [106] Qiang Zhang, Peter Cui, David Yan, Jingkai Sun, Yiqun Duan, Gang Han, Wen Zhao, Weining Zhang, Yijie Guo, Arthur Zhang, et al. Whole-body humanoid robot locomotion with human reference. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11225–11231. IEEE, 2024. [2.1](#)
- [107] Ziwen Zhuang, Shenzhe Yao, and Hang Zhao. Humanoid parkour learning. *arXiv preprint arXiv:2406.10759*, 2024. [1](#), [2.1](#)