ASAP: Aligning Simulation and Real-World Physics for Learning Agile Humanoid Whole-Body Skills

Tairan He[†1,2] Jiawei Gao[†1] Wenli Xiao[†1,2] Yuanhang Zhang[†1] Zi Wang[1] Jiashun Wang[1]
Zhengyi Luo[1,2] Guanqi He[1] Nikhil Sobanbab[1] Chaoyi Pan[1] Zeji Yi[1] Guannan Qu[1]
Kris Kitani[1] Jessica Hodgins[1] Linxi "Jim" Fan[2] Yuke Zhu[2] Changliu Liu[1] Guanya Shi[1]
[1]Carnegie Mellon University [2]NVIDIA [†]Equal Contributions
Page: https://agile.human2humanoid.com Code: https://github.com/LeCAR-Lab/ASAP

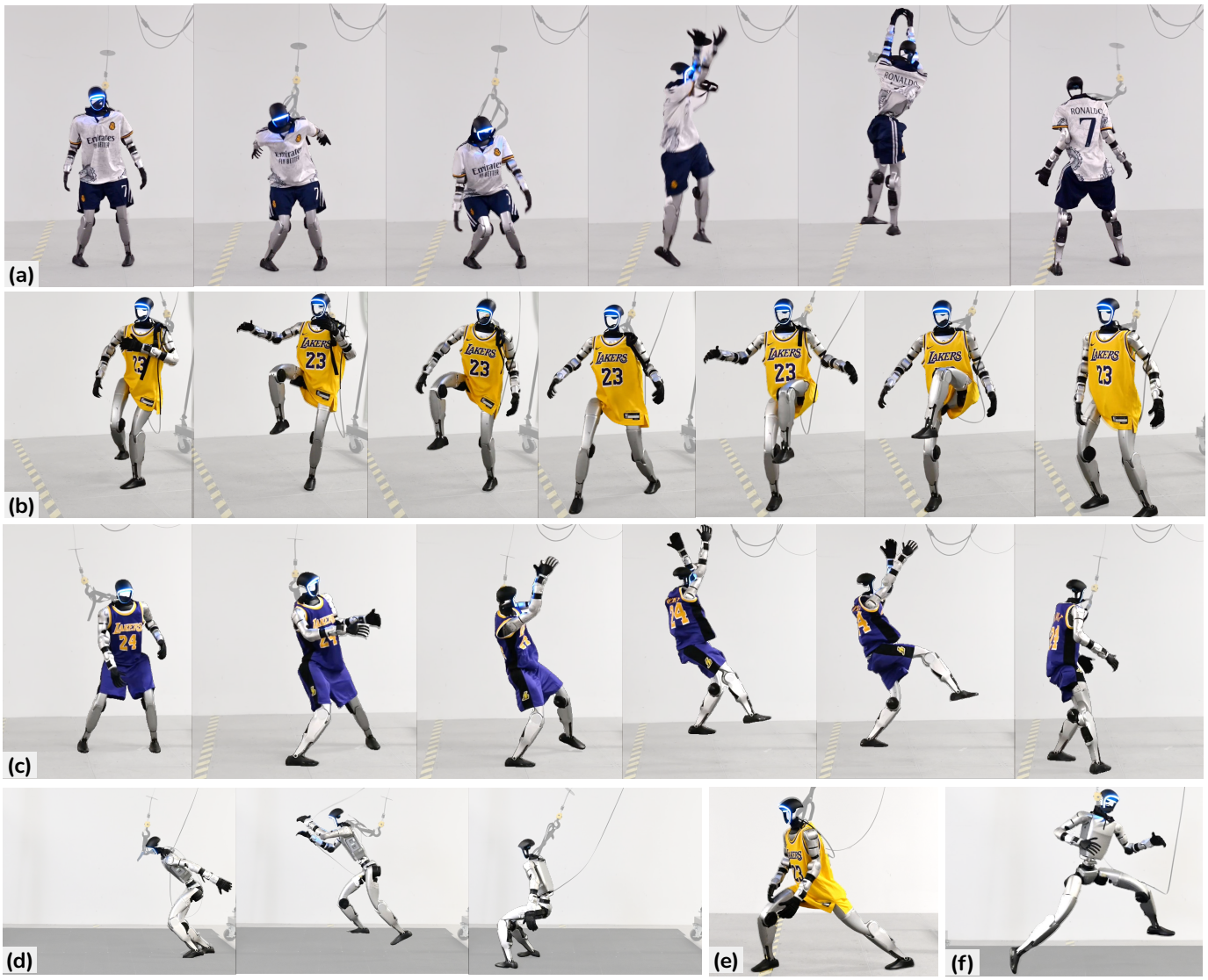

Fig. 1. The humanoid robot (Unitree G1) demonstrates diverse agile whole-body skills, showcasing the control policies' agility: (a) Cristiano Ronaldo's signature celebration involving a jump with a 180-degree mid-air rotation; (b) LeBron James's "Silencer" celebration involving single-leg balancing; 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.

***Abstract*— Humanoid robots hold the potential for unparalleled versatility for performing human-like, whole-body skills. However, achieving agile and coordinated whole-body motions remains a significant challenge due to the dynamics mismatch between simulation and the real world. Existing approaches, such as system identification (SysID) and domain randomization (DR) methods, often rely on labor-intensive parameter tuning or result in overly conservative policies that sacrifice agility. In this paper, we present ASAP (Aligning Simulation and Real Physics), a two-stage framework designed to tackle the dynamics mismatch and enable agile humanoid whole-body skills. In the first stage, we pre-train motion tracking policies in simulation using retargeted human motion data. In the second stage, we deploy the policies in the real world and collect real-world data**

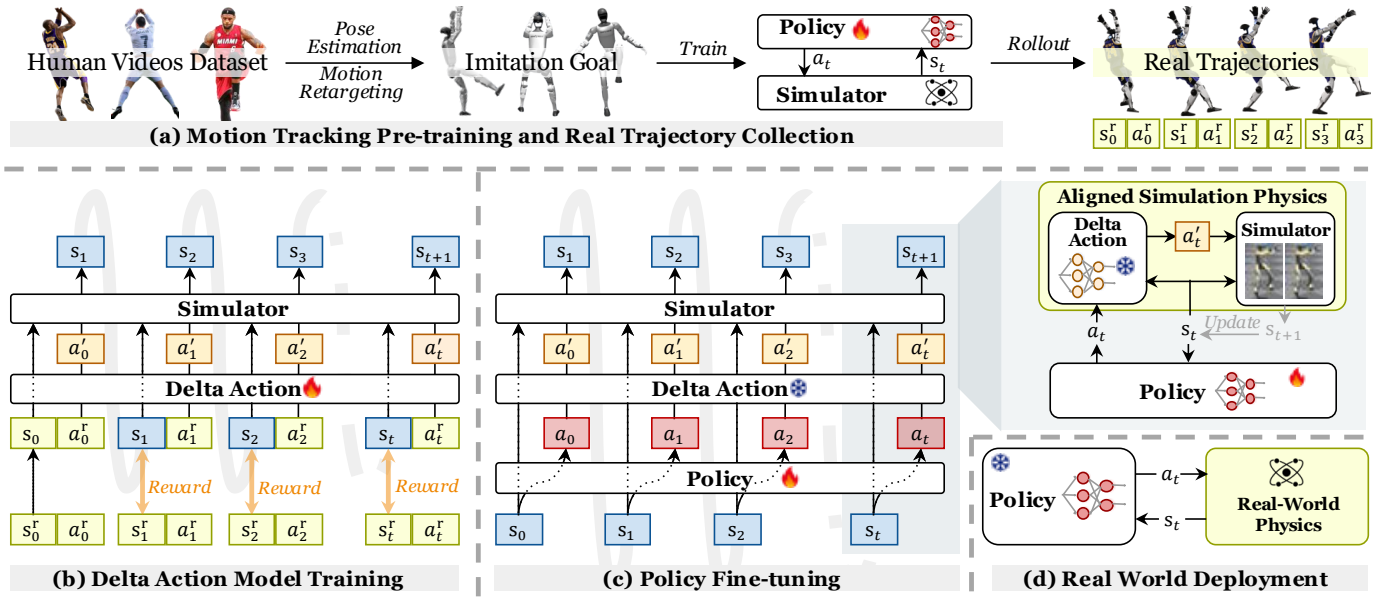

Fig. 2. Overview of ASAP. (a) **Motion Tracking Pre-training and Real Trajectory Collection**: With the humanoid motions retargeted from human videos, we pre-train multiple motion tracking policies to roll out real-world trajectories. (b) **Delta Action Model Training**: Based on the real-world rollout 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.

to train a delta (residual) action model that compensates for the dynamics mismatch. Then ASAP fine-tunes pre-trained policies with the delta action model integrated into the simulator to align effectively with real-world dynamics. We evaluate ASAP across three transfer scenarios—IsaacGym to IsaacSim, IsaacGym to Genesis, and IsaacGym to the real-world Unitree G1 humanoid robot. Our approach significantly improves agility and whole-body coordination across various dynamic motions, reducing tracking error compared to SysID, DR, and delta dynamics learning baselines. ASAP enables highly agile motions that were previously difficult to achieve, demonstrating the potential of delta action learning in bridging simulation and real-world dynamics. These results suggest a promising sim-to-real direction for developing more expressive and agile humanoids.

I. Introduction

For decades, we have envisioned humanoid robots achieving or even surpassing human-level agility. However, most prior work [46, 74, 47, 73, 107, 20, 95, 50] has primarily focused on locomotion, treating the legs as a means of mobility. Recent studies [11, 26, 25, 27, 33] have introduced whole-body expressiveness in humanoid robots, but these efforts have primarily focused on upper-body motions and have yet to achieve the agility seen in human movement. Achieving agile, whole-body skills in humanoid robots remains a fundamental challenge due to not only hardware limits but also the mismatch between simulated dynamics and real-world physics.

Three main approaches have emerged to bridge the dynamics mismatch: System Identification (SysID) methods, domain randomization (DR), and learned dynamics methods. SysID methods directly estimate critical physical parameters, such as motor response characteristics, the mass of each robot link, and terrain properties [102, 20]. However, these methods require a pre-defined parameter space [49], which may not fully capture the sim-to-real gap, especially when real-world dynamics fall outside the modeled distribution. SysID also often relies on ground truth torque measurements [30], which are unavailable on many widely used hardware platforms, limiting its practical applicability. DR methods, in contrast, first train control policies in simulation before deploying them on real-world hardwares [85, 79, 59]. To mitigate the dynamics mismatch between simulation and real-world physics, DR methods rely on randomizing simulation parameters [87, 68]; but this can lead to overly conservative policies [26], ultimately hindering the development of highly agile skills. Another approach to bridge dynamics mismatch is learning a dynamics model of real-world physics using real-world data. While this approach has demonstrated success in low-dimensional systems such as drones [81] and ground vehicles [97], its effectiveness for humanoid robots remains unexplored.

To this end, we propose ASAP, a two-stage framework that aligns the dynamics mismatch between simulation and real-world physics, enabling agile humanoid whole-body skills. ASAP involves a pre-training stage where we train base policies in simulation and a post-training stage that finetunes the policy by aligning simulation and real-world dynamics. In the **pre-training** stage, we train a motion tracking policy in simulation using human motion videos as data sources. These motions are first retargeted to humanoid robots [26], and a phase-conditioned motion tracking policy [67] is trained to follow the retargeted movements. However, directly deploying this policy on real hardware results in degraded performance due to the dynamics mismatch. To address this, the **post-training** stage collects real-world rollout data, including proprioceptive states and positions recorded by the motion capture

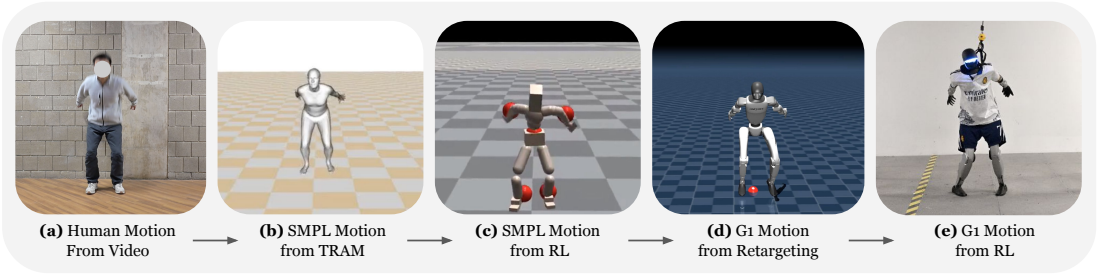

Fig. 3. Retargeting Human Video Motions to Robot Motions: (a) Human motions are captured from video. (b) Using TRAM [93], 3D human motion is reconstructed in the SMPL parameter format. (c) A reinforcement learning (RL) policy is trained in simulation to track the SMPL motion. (d) The learned SMPL motion 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.

system. The collected data are then replayed in simulation, where the dynamics mismatch manifests as tracking errors. We then train a delta action model that learns to compensate for these discrepancies by minimizing the difference between real-world and simulated states. This model effectively serves as a residual correction term for the dynamics gap. Finally, we fine-tune the pre-trained policy using the delta action model, allowing it to adapt effectively to real-world physics.

We validate ASAP on diverse agile motions and successfully achieve whole-body agility on the Unitree G1 humanoid robot [77]. Our approach significantly reduces motion tracking error compared to prior SysID, DR, and delta dynamics learning baselines in both sim-to-sim (IsaacGym to IsaacSim, IsaacGym to Genesis) and sim-to-real (IsaacGym to Real) transfer scenarios. Our contributions are summarized below.

1) We introduce ASAP, a framework that bridges the sim-to-real gap by leveraging a delta action model trained via reinforcement learning (RL) with real-world data.
2) We successfully deploy RL-based whole-body control policies in the real world, achieving previously difficult-to-achieve humanoid motions.
3) Extensive experiments in both simulation and real-world settings demonstrate that ASAP effectively reduces dynamics mismatch, enabling highly agile motions on robots and significantly reducing motion tracking errors.
4) To facilitate smooth transfer between simulators, we develop and open-source a multi-simulator training and evaluation codebase for help accelerate further research.

II. Pre-training: Learning Agile Humanoid Skills

A. Data Generation: Retargeting Human Video Data

To track expressive and agile motions, we collect a video dataset of human movements and retarget it to robot motions, creating imitation goals for motion-tracking policies, as shown in Figure 3 and Figure 2 (a).

a) Transforming Human Video to SMPL Motions: We begin by recording videos (see Figure 3 (a) and Figure 12) of humans performing expressive and agile motions. Using TRAM [93], we reconstruct 3D motions from videos. TRAM estimates the global trajectory of the human motions in SMPL parameter format [52], which includes global root translation, orientation, body poses, and shape parameters, as shown in Figure 3 (b). The resulting motions are denoted as $\mathcal{D}_{\text{SMPL}}$.

b) Simulation-based Data Cleaning: Since the reconstruction process can introduce noise and errors [26], some estimated motions may not be physically feasible, making them unsuitable for motion tracking in the real world. To address this, we employ a "sim-to-data" cleaning procedure. Specifically, we use MaskedMimic [86], a physics-based motion tracker, to imitate the SMPL motions from TRAM in IsaacGym simulator [58]. The motions (Figure 3 (c)) that pass this simulation-based validation are saved as the cleaned dataset $\mathcal{D}_{\text{SMPL}}^{\text{Cleaned}}$.

c) Retargeting SMPL Motions to Robot Motions: With the cleaned dataset $\mathcal{D}_{\text{SMPL}}^{\text{Cleaned}}$ in SMPL format, we retarget the motions into robot motions following the shape-and-motion two-stage retargeting process [26]. Since the SMPL parameters estimated by TRAM represent various human body shapes, we first optimize the shape parameter $\boldsymbol{\beta}'$ to approximate a humanoid shape. By selecting 12 body links with correspondences between humans and humanoids, we perform gradient descent on $\boldsymbol{\beta}'$ to minimize joint distances in the rest pose. Using the optimized shape $\boldsymbol{\beta}'$ along with the original translation $\boldsymbol{p}$ and pose $\boldsymbol{\theta}$, we apply gradient descent to further minimize the distances of the body links. This process ensures accurate motion retargeting and produces the cleaned robot trajectory dataset $\mathcal{D}_{\text{Robot}}^{\text{Cleaned}}$, as shown in Figure 3 (d).

B. Phase-based Motion Tracking Policy Training

We formulate the motion-tracking problem as a goal-conditioned reinforcement learning (RL) task, where the policy π is trained to track the retargeted robot movement trajectories in the dataset $\mathcal{D}_{\text{Robot}}^{\text{Cleaned}}$. Inspired by [67], the state s_t includes the robot's proprioception s_t^{p} and a time phase variable $\phi \in [0, 1]$, where $\phi = 0$ represents the start of a motion and $\phi = 1$ represents the end. This time phase variable alone ϕ is proven to be sufficient to serve as the goal state $\boldsymbol{s}_t^{\text{g}}$ for single-motion tracking [67]. The proprioception s_t^{p} is defined as $s_t^{\text{p}} \triangleq \left[\boldsymbol{q}_{t-4:t}, \dot{\boldsymbol{q}}_{t-4:t}, \boldsymbol{\omega}_{t-4:t}^{root}, \boldsymbol{g}_{t-4:t}, \boldsymbol{a}_{t-5:t-1}\right]$, with 5-step history of joint position $\boldsymbol{q}_t \in \mathbb{R}^{23}$, joint velocity $\dot{\boldsymbol{q}}_t \in \mathbb{R}^{23}$, root angular velocity $\boldsymbol{\omega}_t^{root} \in \mathbb{R}^3$, root projected gravity $\boldsymbol{g}_t \in \mathbb{R}^3$, and last action $\boldsymbol{a}_{t-1} \in \mathbb{R}^{23}$. Using the agent's proprioception s_t^{p} and the goal state $\boldsymbol{s}_t^{\text{g}}$, we define the reward as $r_t = \mathcal{R}\left(s_t^{\text{p}}, s_t^{\text{g}}\right)$, which is used for policy optimization. The specific reward terms can be found in Table I. The action $\boldsymbol{a}_t \in \mathbb{R}^{23}$ corresponds to the target joint positions and is passed to a

PD controller that actuates the robot's degrees of freedom. To optimize the policy, we use the proximal policy optimization (PPO) [80], aiming to maximize the cumulative discounted reward $\mathbb{E}\left[\sum_{t=1}^{T}\gamma^{t-1}r_t\right]$. We identify several design choices that are crucial for achieving stable policy training:

a) Asymmetric Actor-Critic Training: Real-world humanoid control is inherently a partially observable Markov decision process (POMDP), where certain task-relevant properties that are readily available in simulation become unobservable in real-world scenarios. However, these missing properties can significantly facilitate policy training in simulation. To bridge this gap, we employ an asymmetric actor-critic framework, where the critic network has access to privileged information such as the global positions of the reference motion and the root linear velocity, while the actor network relies solely on proprioceptive inputs and a time-phase variable. This design not only enhances phase-based motion tracking during training but also enables a simple, phase-driven motion goal for sim-to-real transfer. Crucially, because the actor does not depend on position-based motion targets, our approach eliminates the need for odometry during real-world deployment—overcoming a well-documented challenge in prior work on humanoid robots [26, 25].

b) Termination Curriculum of Tracking Tolerance: Training a policy to track agile motions in simulation is challenging, as certain motions can be too difficult for the policy to learn effectively. For instance, when imitating a jumping motion, the policy often fails early in training and learns to remain on the ground to avoid landing penalties. To mitigate this issue, we introduce a termination curriculum that progressively refines the motion error tolerance throughout training, guiding the policy toward improved tracking performance. Initially, we set a generous termination threshold of 1.5m, meaning the episode terminates if the robot deviates from the reference motion by this margin. As training progresses, we gradually tighten this threshold to 0.3m, incrementally increasing the tracking demand on the policy. This curriculum allows the policy to first develop basic balancing skills before progressively enforcing stricter motion tracking, ultimately enabling successful execution of high-dynamic behaviors.

c) Reference State Initialization: Task initialization plays a crucial role in RL training. We find that naively initializing episodes at the start of the reference motion leads to policy failure. For example, in Cristiano Ronaldo's jumping training, starting the episode from the beginning forces the policy to learn sequentially. However, a successful backflip requires mastering the landing first—if the policy cannot land correctly, it will struggle to complete the full motion from takeoff. To address this, we adopt the Reference State Initialization (RSI) framework [67]. Specifically, we randomly sample time-phase variables between 0 and 1, which effectively randomizes the starting point of the reference motion for the policy to track. We then initialize the robot's state based on the corresponding reference motion at that phase, including root position and orientation, root linear and angular velocities and joint positions and velocities. This initialization strategy significantly improves motion tracking training, particularly for agile whole-body motions, by allowing the policy to learn different motion phases in parallel rather than being constrained to a strictly sequential learning process.

d) Reward Terms: We define the reward function r_t with the sum of three terms: 1) penalty, 2) regularization, and 3) task rewards. A detailed summary of these components is provided in Table I.

TABLE I
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

e) Domain Randomizations: To improve the robustness of the pre-trained policy in Figure 2 (a), we utilized basic domain randomization techniques listed in Table VI.

III. POST-TRAINING: TRAINING DELTA ACTION MODEL AND FINE-TUNING MOTION TRACKING POLICY

The policy trained in the first stage can track the reference motion in the real-world but does not achieve high motion quality. Thus, during the second stage, as shown in Figure 2 (b) and (c), we leverage real-world data rolled out by the pre-trained policy to train a delta action model, followed by policy refinement through dynamics compensation using this learned delta action model.

A. Data Collection

We deploy the pretrained policy in the real world to perform whole-body motion tracking tasks (as depicted in Figure 9) and record the resulting trajectories, denoted as $\mathcal{D}^{\text{r}} = \{s_0^{\text{r}}, a_0^{\text{r}}, \ldots, s_T^{\text{r}}, a_T^{\text{r}}\}$, as illustrated in Figure 2 (a). At each timestep t, we use a motion capture device and onboard sensors to record the state: $s_t = [p_t^{\text{base}}, v_t^{\text{base}}, \alpha_t^{\text{base}}, \omega_t^{\text{base}}, q_t, \dot{q}_t]$, where $p_t^{\text{base}} \in \mathbb{R}^3$ represents the robot base 3D position, $v_t^{\text{base}} \in \mathbb{R}^3$ is base linear velocity, $\alpha_t^{\text{base}} \in \mathbb{R}^4$ is the robot base orientation represented as a quaternion, $\omega_t^{\text{base}} \in \mathbb{R}^3$ is the base angular velocity, $q_t \in \mathbb{R}^{23}$ is the vector of joint positions, and $\dot{q}_t \in \mathbb{R}^{23}$ represents joint velocities.

B. Training Delta Action Model

Due to the sim-to-real gap, when we replay the real-world trajectories in simulation, the resulting simulated trajectory will likely deviate significantly from real-world recorded trajectories. This discrepancy is a valuable learning signal for learning the mismatch between simulation and real-world

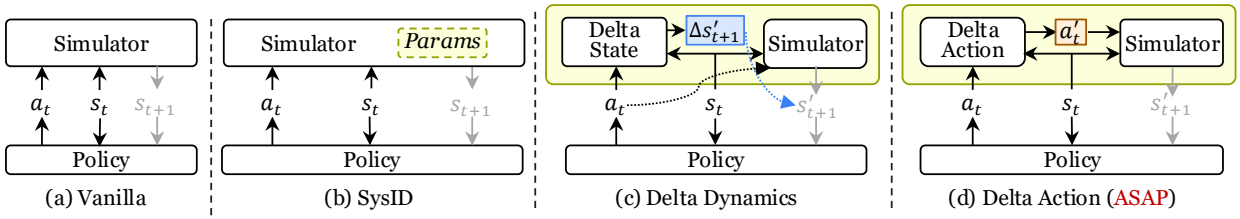

Fig. 4. Baselines of ASAP. (a) Model-free RL training. (b) System ID from real to sim using real-world data. (c) Learning delta dynamics model using real-world data. (d) Our proposed method, learning delta action model using real-world data.

physics. We leverage an RL-based delta/residual action model to compensate for the sim-to-real physics gap.

As illustrated in Figure 2 (b), the delta action model is defined as $\Delta a_t = \pi^{\Delta}_{\theta}(s_t, a_t)$, where the policy π^{Δ}_{θ} learns to output corrective actions based on the current state s_t and the action a_t. These corrective actions (Δa_t) are added to the real-world recorded actions (a^r_t) to account for discrepancies between simulation and real-world dynamics.

The RL environment incorporates this delta action model by modifying the simulator dynamics as follows: $s_{t+1} = f^{\text{sim}}(s_t, a^r_t + \Delta a_t)$ where f^{sim} represents the simulator's dynamics, a^r_t is the reference action recorded from real-world rollouts, and Δa_t introduces corrections learned by the delta action model.

TABLE II
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

During each RL step:

1) The robot is initialized at the real-world state s^r_t.
2) A reward signal is computed to minimize the discrepancy between the simulated state s_{t+1} and the recorded real-world state s^r_{t+1}, with an additional action magnitude regularization term $\exp(-\|a_t\|) - 1$), as specified in Table II. The workflow is illustrated in Figure 2 (b).
3) PPO is used to train the delta action policy π^{Δ}_{θ}, learning corrected Δa_t to match simulation and the real world.

By learning the delta action model, the simulator can accurately reproduce real-world failures. For example, consider a scenario where the simulated robot can jump because its motor strength is overestimated, but the real-world robot cannot jump due to weaker motors. The delta action model π^{Δ}_{θ} will learn to reduce the intensity of lower-body actions, simulating the motor limitations of the real-world robot. This allows the simulator to replicate the real-world dynamics and enables the policy to be fine-tuned to handle these limitations effectively.

C. Fine-tuning Motion Tracking Policy under New Dynamics

With the learned delta action model $\pi^{\Delta}(s_t, a_t)$, we can reconstruct the simulation environment with

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

As shown in Figure 2 (c), we keep the π^{Δ} model parameters frozen, and fine-tune the pretrained policy with the same reward summarized in Table I.

D. Policy Deployment

Finally, we deploy the fine-tuned policy without delta action model in the real world as shown in Figure 2 (d). The fine-tuned policy shows enhanced real-world motion tracking performance compared to the pre-trained policy. Quantitative improvements will be discussed in Section IV.

IV. PERFORMANCE EVALUATION OF ASAP

In this section, we present extensive experimental results on three policy transfers: IsaacGym [58] to IsaacSim [63], IsaacGym to Genesis [7], 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 IV-A and Section IV-B) and real-world settings (Section IV-C).

In the simulation, we use the retargeted motion dataset from the videos we shoot, denoted as $\mathcal{D}^{\text{Cleaned}}_{\text{Robot}}$, which contains diverse human motion sequences. We select 43 motions categorized into three difficulty levels: easy, medium, and hard (as partially visualized in Figure 6), 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 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

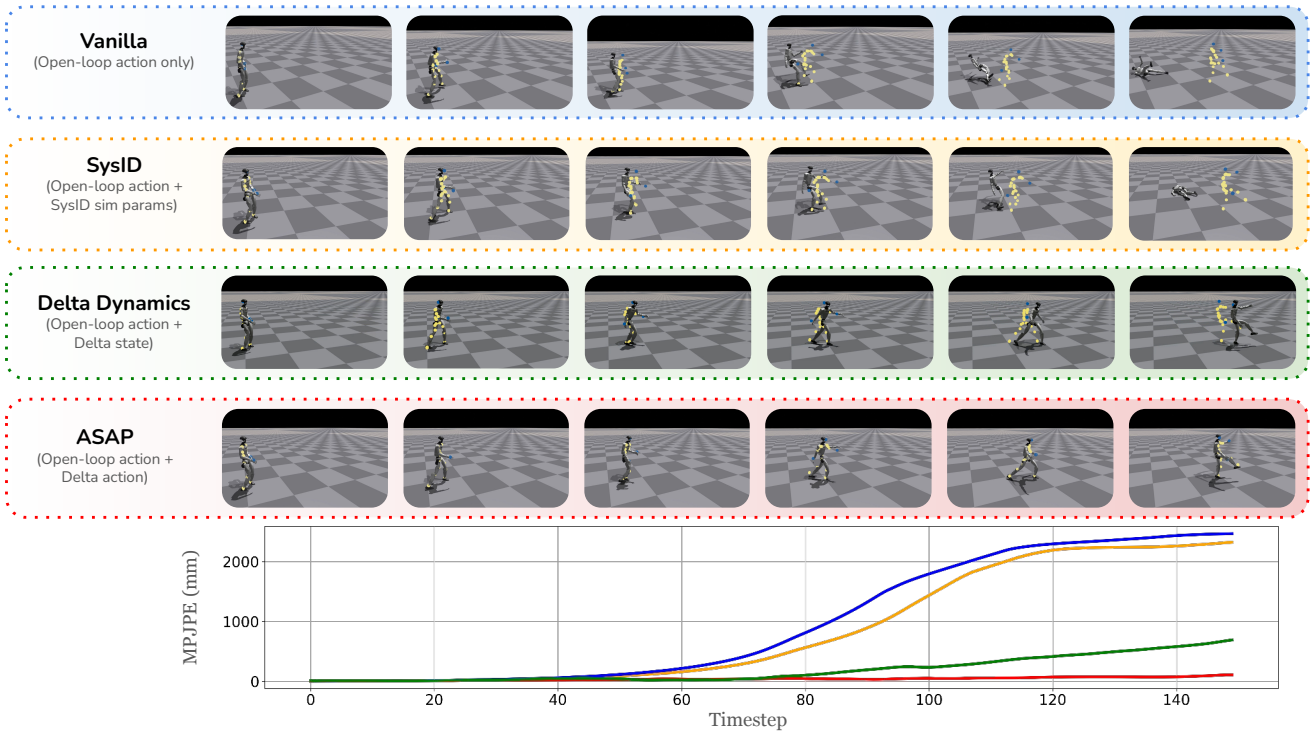

Fig. 5. Replaying IsaacSim State-Action trajecories in IsaacGym. The upper four panels visualize the Unitree G1 humanoid executing a soccer-shooting motion under four distinct open-loop actions. Corresponding metric curves (bottom) quantify tracking performance. Importantly, our delta action model (ASAP) is trained across multiple motions and is not overfitted to this specific example.

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

`SysID` (Figure 4 b): We identify the following representative parameters in our 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 $i = 1, 2, ..., 23$. Specifically, we search the best parameters among certain discrete ranges by replaying the recorded trajectories in real with different simulation parameters summarized in Table VII. We then finetune the pre-trained policy in IsaacGym with the best SysID parameters.

`DeltaDynamics` (Figure 4 c): We train a residual dynamics model $f_\theta^\Delta(s_t, a_t)$ to capture the discrepancy between simulated and real-world physics. The detailed implementation is introduced in Section VIII-C

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 $E_{\text{g-mpjpe}}$ (mm), the root-relative mean per-joint (MPJPE) E_{mpjpe} (mm), acceleration error E_{acc} (mm/frame^2), and root velocity E_{vel} (mm/frame). The mean values of the metrics are computed across all motion sequences used.

TABLE III
OPEN-LOOP PERFORMANCE COMPARISON ACROSS SIMULATORS AND MOTION LENGTHS.

Simulator & Length		IsaacSim				Genesis			
Length	Method	$E_{\text{g-mpjpe}}$	E_{mpjpe}	E_{acc}	E_{vel}	$E_{\text{g-mpjpe}}$	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**

A. 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 effectiveness 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-

TABLE IV

CLOSED-LOOP MOTION IMITATION EVALUATION ACROSS DIFFERENT SIMULATORS. ALL VARIANTS ARE TRAINED WITH IDENTICAL REWARDS.

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

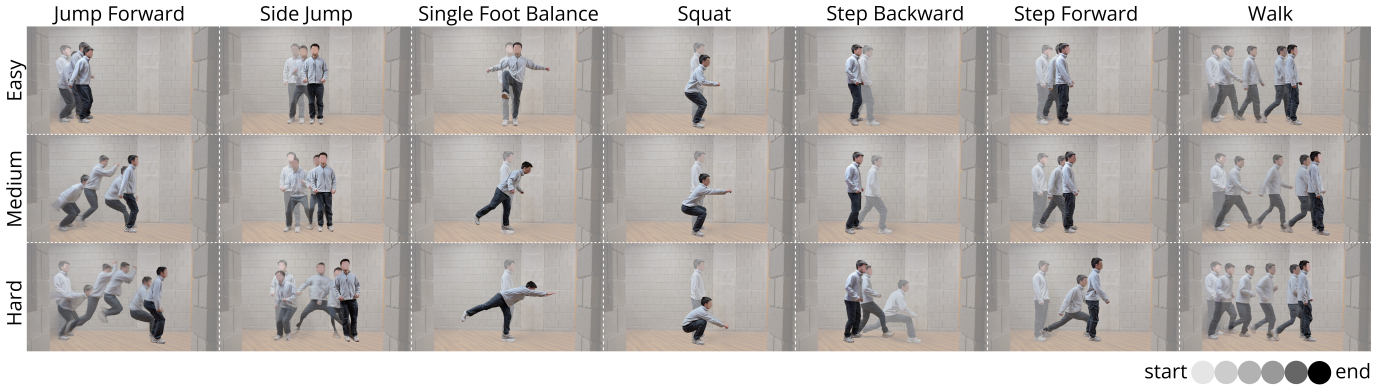

Fig. 6. Visual comparisons of motion imitation results across different difficulty levels (Easy, Medium, Hard) for various tasks including Jump Forward, Side Jump, Single Foot Balance, Squat, Step Backward, Step Forward, and Walk.

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 III demonstrate that ASAP consistently outperforms the OpenLoop baseline across all replayed motion lengths, achieving lower $E_{\text{g-mpjpe}}$ and E_{mpjpe} values, which 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 5. 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.

B. 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 IV, 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 $E_{\text{g-mpjpe}}$ and E_{mpjpe} in IsaacSim ($E_{\text{g-mpjpe}} = 106$ and $E_{\text{mpjpe}} = 44.3$) and Genesis ($E_{\text{g-mpjpe}} = 125$ and $E_{\text{mpjpe}} = 73.5$), with minimal acceleration (E_{acc}) and velocity (E_{vel}) errors. In more challenging tasks, such as the Hard level, our method continues to excel, significantly reducing motion-tracking errors. For instance, in Genesis, it achieves $E_{\text{g-mpjpe}} = 129$ 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 7, comparing ASAP with

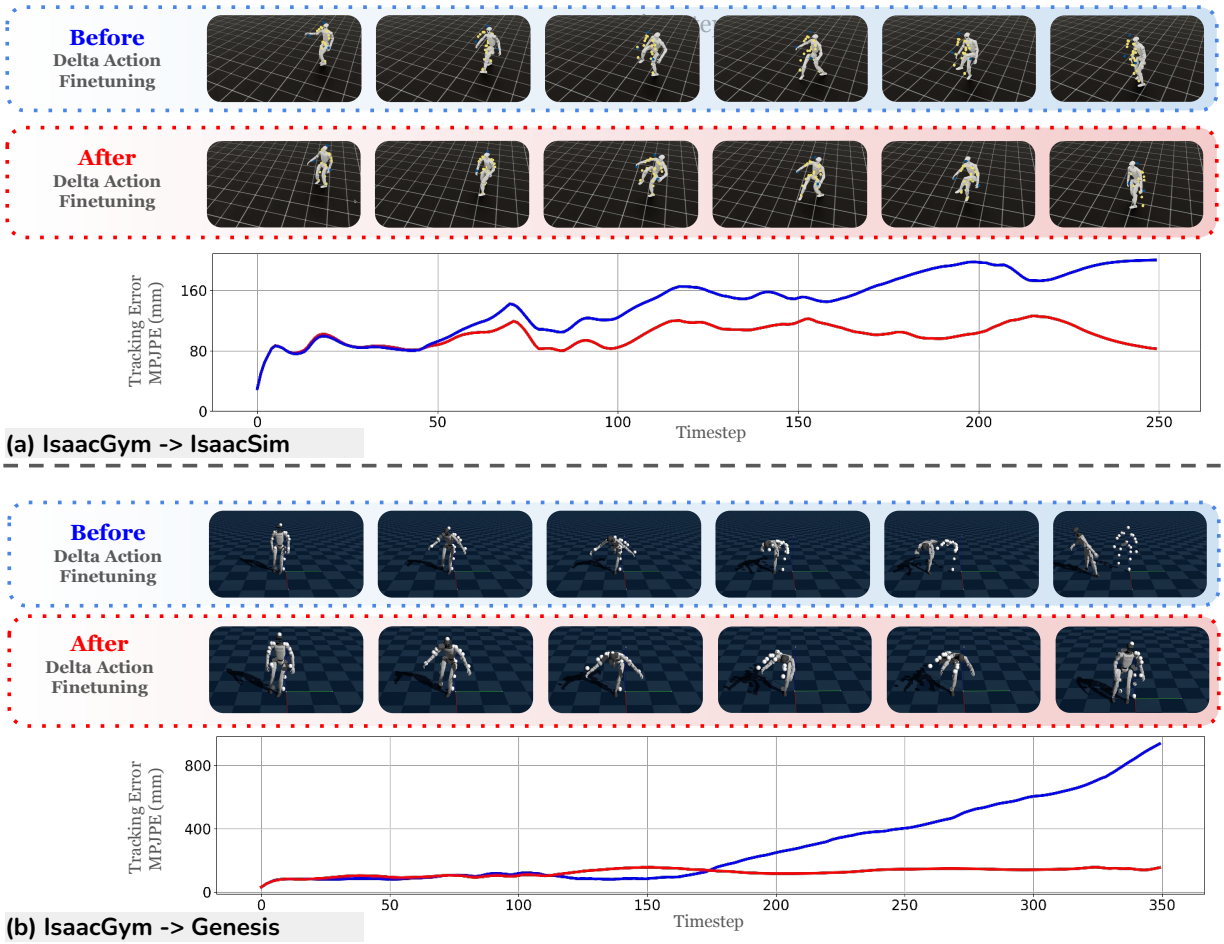

Fig. 7. Visualization of G1 motion tracking before and after ASAP fine-tuning in IsaacGym, IsaacSim and Genesis. Top: LeBron James' "Silencer" motion tracking policy fine-tuning for IsaacGym to IsaacSim. Bottom: *single foot balance* motion tracking policy fine-tuning for IsaacGym to Genesis.

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.

C. 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 III-B—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

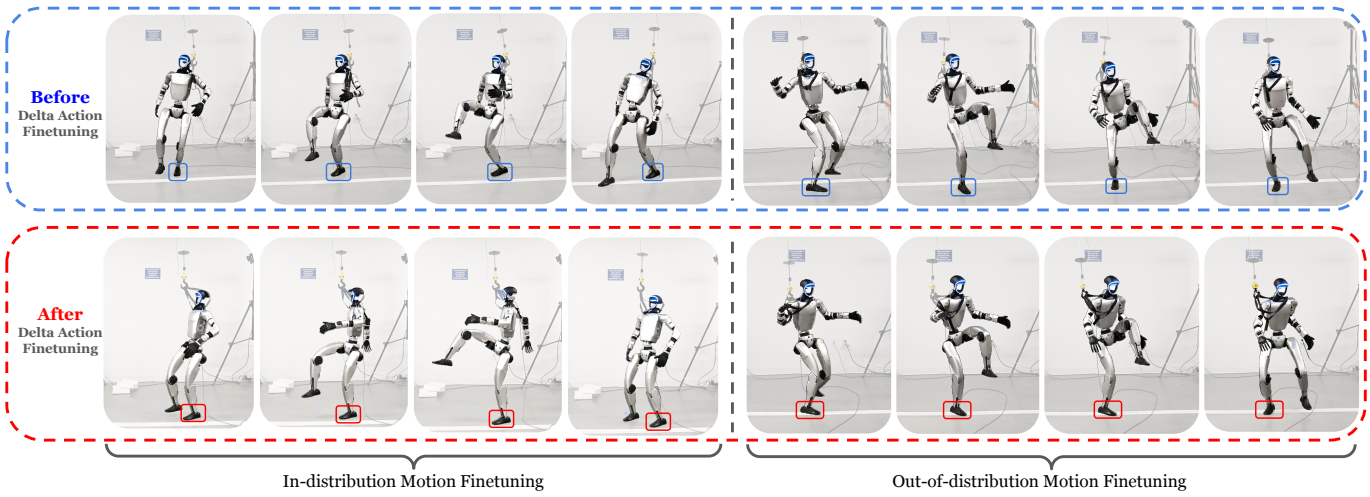

Fig. 8. Visualization of LeBron James' "Silencer" motion on the G1 robot before (upper figure enclosed in blue) and after (bottom figure enclosed in red) ASAP policy finetuning. The left half shows the policy finetuning for the in-distribution motions while the right half shows the out-of-distribution ones. After ASAP finetuning, the robot behaves more smoothly and reduces jerky lower-body motions.

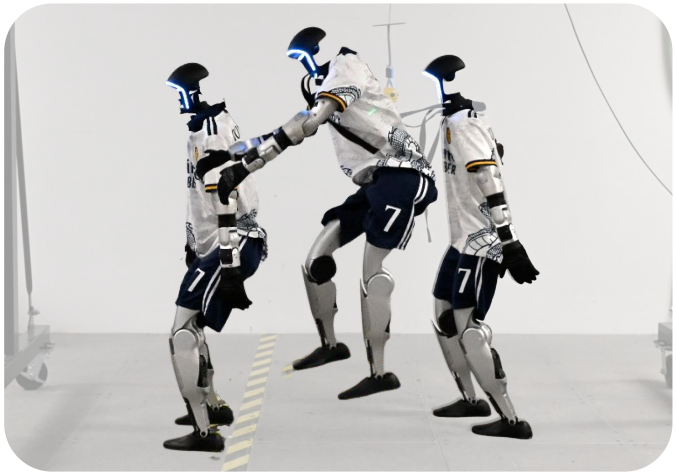

Fig. 9. We deploy the pretrained policy of a forward jump motion tracking task, challenging the 1.35m-tall Unitree G1 robot for a forward leap over 1m.

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 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 and Figure 8. The experiment data is summarized in Table V. 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 ($E_{\text{g-mpjpe}}, E_{\text{mpjpe}}, E_{acc}$ and E_{vel}). These findings highlight the effectiveness of ASAP in improving sim-to-real transfer for agile humanoid motion tracking.

TABLE V
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	$E_{\text{g-mpjpe}}$	E_{mpjpe}	E_{acc}	E_{vel}	$E_{\text{g-mpjpe}}$	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**

V. 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?

A. 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 norm weight, evaluating their effects on both open-loop and closed-loop performance. Our analysis uncovers the

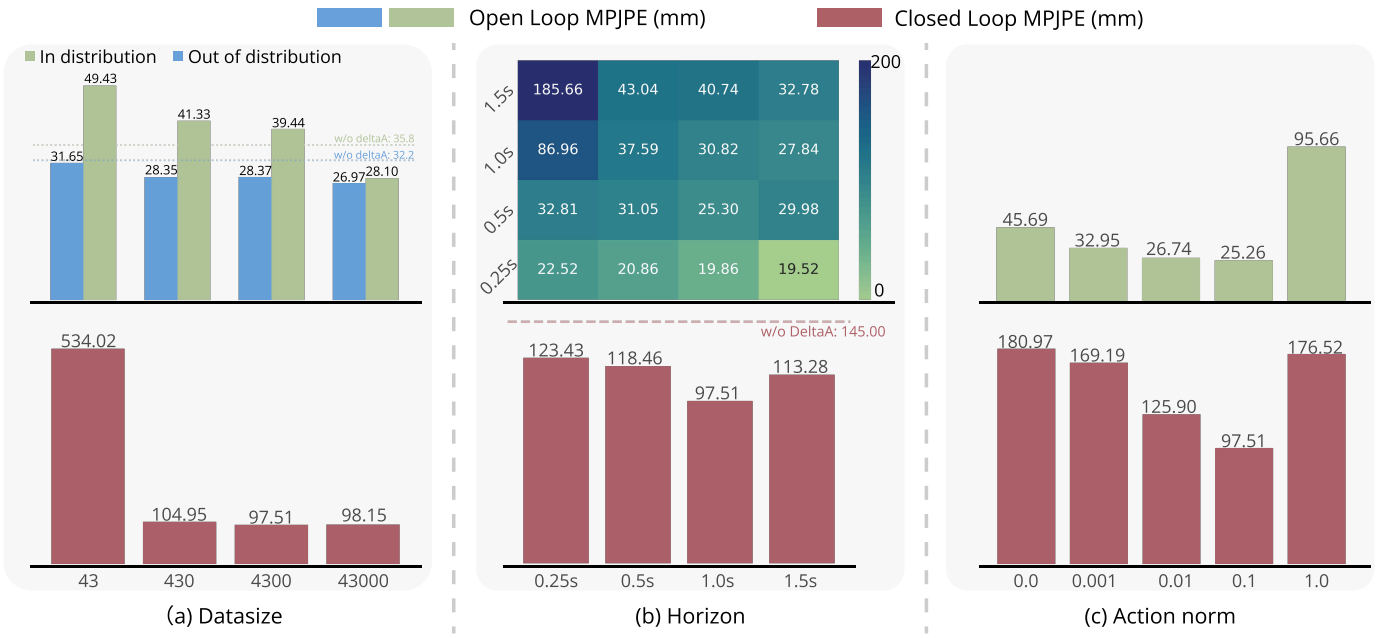

Fig. 10. Analysis of dataset size, training horizon, and action norm on the performance of π^{Δ}. (a) **Dataset Size**: Mean Per Joint Position Error (MPJPE) is evaluated for both in-distribution (green) and out-of-distribution (blue) scenarios. Increasing dataset size leads to enhanced generalization, 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.

essential principles for effectively training a high-performing delta action model.

a) 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 10 (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.

b) Training Horizon: The rollout horizon plays a crucial role in learning π^{Δ}. As shown in Figure 10 (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.

c) Action Norm Weight: Training π^{Δ} incorporates an action norm reward to balance dynamics alignment and minimal correction. As illustrated in Figure 10 (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.

B. 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. The detailed derivation of these two baselines is summarized in Section VIII-D.

Our experiments in Figure 11 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 (more discussions in Section VIII-D).

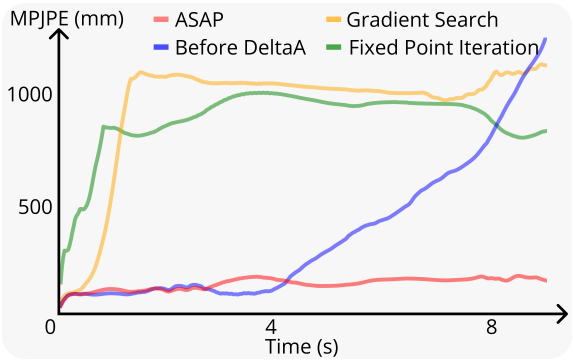

Fig. 11. MPJPE comparison over timesteps for fine-tuning methods using delta actionmodel. RL Fine-Tuning achieves the lowest error, while Fixed-Point Iteration and Gradient Search perform worse than the baseline (Before DeltaA) showing the highest error.

C. 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 [8] 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 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 12, 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 13, 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 5, demonstrate that delta action

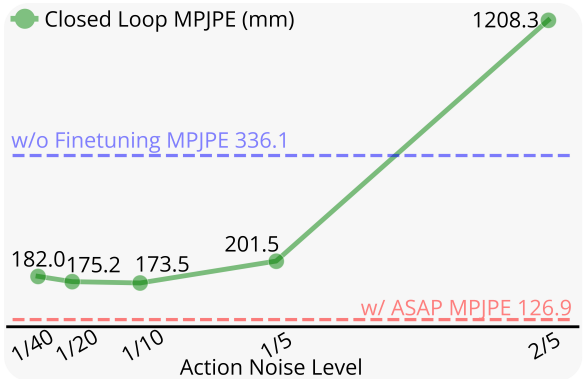

Fig. 12. MPJPE vs. Noise Level for policies fine-tuned with random action noise. Policies with noise levels $\beta \in [0.025, 0.2]$ show improved performance compared to no fine-tuning. Delta action achieves better tracking precision (126 MPJPE) compared to the best action noise (173 MPJPE).

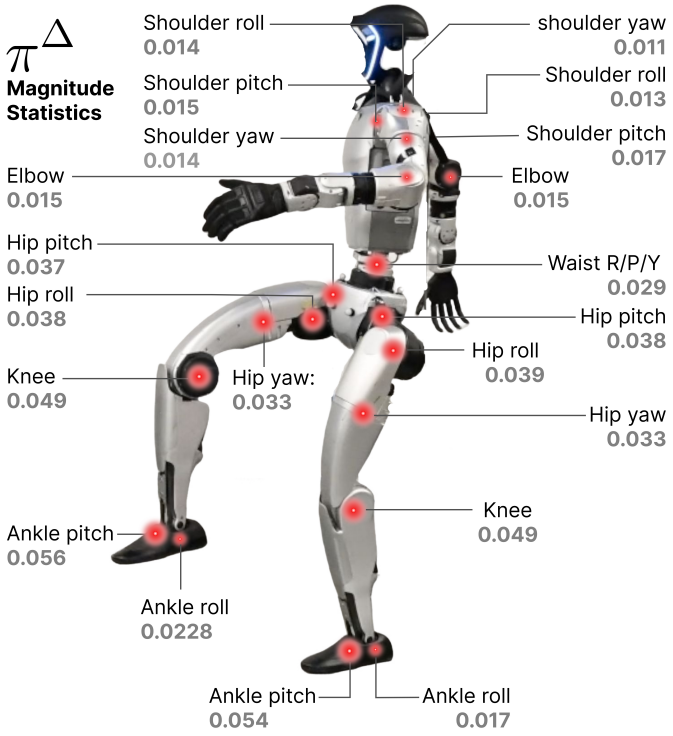

Fig. 13. Visualization of IsaacGym-to-IsaacSim π^{Δ} output magnitude. We compute the average absolute value of each joint over the 4300-episode dataset. Larger red dots indicate higher values. The results suggest that lower-body motors exhibit a larger discrepancy compared to upper-body joints, with the most significant gap observed in the ankle pitch joint of the G1 humanoid.

not only enhances policy robustness but also enables effective adaptation to real-world dynamics, outperforming naive randomization strategies.

VI. Related Works

A. Learning-based Methods for 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 [44, 98, 45, 48, 47, 74, 73, 20, 106], jumping [46], and

parkour [50, 107]. More advanced capabilities, such as dancing [105, 33, 11], loco-manipulation [26, 53, 16, 25], 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 [71, 86, 55], including cartwheels [67], backflips [69], sports movements [104, 90, 56, 91, 92], and smooth object interactions [86, 18, 23]. 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.

B. Offline and Online System Identification for Robotics

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 [39, 6], 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 [85, 30, 99], refining robot dynamics models [36, 3, 19, 22, 31], explicitly identifying critical simulation parameters [102, 10, 14, 96], learning a distribution over simulation parameters [75, 28, 5], 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 [101, 103, 43, 70], regression-based methods [100, 40, 89, 20, 32, 60, 15, 72, 61, 41, 62, 42], next-state reconstruction techniques [65, 51, 54, 94, 83], direct reward maximization [47], or by leveraging tracking and prediction errors for online adaptation [66, 57, 29, 17]. 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.

C. Residual Learning for Robotics

Learning a residual component alongside learned or predefined base models has been widely used in robotics. Prior work has explored residual policy models that refine the actions of an initial controller [84, 35, 9, 2, 13, 21, 4, 34, 42]. Other approaches leverage residual components to correct inaccuracies in dynamics models [66, 1, 38, 82, 24] or to model residual trajectories resulting from residual actions [12] for achieving precise and agile motions. Our framework builds on this idea by using residual actions to align the dynamics mismatch between simulation and real-world physics, enabling agile whole-body humanoid skills.

VII. Conclusion

We present ASAP, a two-stage framework that bridges the sim-to-real gap for agile humanoid control. By learning a universal delta action model to capture dynamics mismatch, ASAP enables policies trained in simulation to adapt seamlessly to real-world physics. Extensive experiments demonstrate significant reductions in motion tracking errors (up to 52.7% in sim-to-real tasks) and successful deployment of diverse agile skills—including agile jumps and kicks—on the Unitree G1 humanoid. Our work advances the frontier of sim-to-real transfer for agile whole-body control, paving the way for versatile humanoid robots in real-world applications.

VIII. Limitations

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 robot actuators, leading to frequent motor overheating and hardware damage during data collection. For example, 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 precise motion capture (MoCap) infrastructure 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 architectures to mitigate hardware risks, leveraging markerless pose estimation or onboard sensor fusion to reduce reliance on MoCap systems, and exploring adaptation techniques for delta action models to achieve sample-efficient few-shot adaptation.

Acknowledgments

We thank Zhenjia Xu, Yizhou Zhao, Yu Fang for help with hardware. We thank Pulkit Goyal, Hawkeye King, Peter Varvak and Haoru Xue for help with motion capture setup. We thank Ziyan Xiong, Yilin Wu and Xian Zhou for support on Genesis integration. We thank Rui Chen, Yifan Sun and Kai Yun for help with G1 hardware. We thank Xuxin Cheng, Chong Zhang and Toru Lin for always being there to help with any problem and answer any question. We thank Unitree Robotics for help with G1 support.

References

[1] Anurag Ajay, Jiajun Wu, Nima Fazeli, Maria Bauza, Leslie P Kaelbling, Joshua B Tenenbaum, and Alberto Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3066–3073. IEEE, 2018.

[2] Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Residual reinforcement learning from demonstrations. arXiv preprint arXiv:2106.08050, 2021.

[3] 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.

[4] 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.

[5] 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.

[6] Karl Johan Åström and Peter Eykhoff. System identification—a survey. Automatica, 7(2):123–162, 1971.

[7] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis.

[8] 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.

[9] Joao Carvalho, Dorothea Koert, Marek Daniv, and Jan Peters. Residual robot learning for object-centric probabilistic movement primitives. arXiv preprint arXiv:2203.03918, 2022.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] Zipeng Fu, Qingqing Zhao, Qi Wu, Gordon Wetzstein, and Chelsea Finn. Humanplus: Humanoid shadowing and imitation from humans. arXiv preprint arXiv:2406.10454, 2024.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] Siddhant Haldar, 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[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.

[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.

[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.

[39] F Kozin and HG Natke. System identification techniques. Structural safety, 3(3-4):269–316, 1986.

[40] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. arXiv preprint arXiv:2107.04034, 2021.

[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.

[42] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. Science robotics, 5(47):eabc5986, 2020.

[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.

[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.

[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.

[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.

[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.

[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.

[49] Lennart Ljung. System identification. In Signal analysis and prediction, pages 163–173. Springer.

[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.

[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.

[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.

[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.

[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.

[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.

[56] Zhengyi Luo, Jiashun Wang, Kangni Liu, Haotian Zhang, Chen Tessler, Jingbo Wang, Ye Yuan, Jinkun Cao, Zihui Lin, Fengyi Wang, et al. Smplolympics: Sports environments for physically simulated humanoids. arXiv preprint arXiv:2407.00187, 2024.

[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.

[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.

[59] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. arXiv preprint arXiv:2205.02824, 2022.

[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.

[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.

[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.

[63] Mayank Mittal, Calvin Yu, Qinxi 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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[77] Unitree Robotics. Unitree g1 humanoid agent ai avatar, 2024. URL https://www.unitree.com/g1.

[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.

[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.

[80] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[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.

[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.

[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.

[84] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. arXiv preprint arXiv:1812.06298, 2018.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[101] Wenhao Yu, C Karen Liu, and Greg Turk. Policy transfer with strategy optimization. arXiv preprint arXiv:1810.05751, 2018.

[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.

[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.

[104] YE YUAN and Viktor Makoviychuk. Learning physically simulated tennis skills from broadcast videos. 2023.

[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.

[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.

[107] Ziwen Zhuang, Shenzhe Yao, and Hang Zhao. Humanoid parkour learning. arXiv preprint arXiv:2406.10759, 2024.

APPENDIX

A. Domain Randomization in Pre-Training

To improve the robustness and generalization of the pre-trained policy in Figure 2 (a), we utilized the domain randomization technics listed in Table VI.

TABLE VI
DOMAIN RANDOMIZATIONS

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

B. SysID Parameters

We identify the following representative robot parameters in our 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 $i = 1, 2, ..., 23$, as shown in Table VII. .

TABLE VII
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	$[0.95, 1.0, 1.05]$

C. Implementation of Delta Dynamics Learning

Using the collected real-world trajectory, we replay the action sequence $\{a_0^{\text{real}}, \ldots, a_T^{\text{real}}\}$ in simulation and record the resulting trajectory $\{s_0^{\text{sim}}, \ldots, s_T^{\text{sim}}\}$. The neural dynamics model f_θ^Δ is trained to predict the difference:

$$s_{t+1}^{\text{real}} - s_{t+1}^{\text{sim}} = f_\theta^\Delta(s_t^{\text{real}}, a_t^{\text{real}}), \quad \forall t.$$

In practice, we compute the mean squared error (MSE) loss in an autoregressive setting, where the model predicts forward for K steps and uses gradient descent to minimize the loss. To balance learning efficiency and stability over long horizons, we implement a schedule that gradually increases K during training. Formally, the optimization objective is:

$$\mathcal{L} = \left\| s_{t+K}^{\text{real}} - \underbrace{f^{\text{sim}}\big(\ldots f^{\text{sim}}}_{K}(s_t, a_t) + f_\theta^\Delta(s_t, a_t), \ldots, a_{t+K}\big) \right\|.$$

After training, we freeze the residual dynamics model f_θ^Δ and integrate it into the simulator. During each simulation step, the robot's state is updated by incorporating the delta predicted by the dynamics model. In this augmented simulation environment, we finetune the previously pretrained policy to adapt to the corrected dynamics, ensuring improved alignment with real-world behavior.

D. Derivation of Training-free Methods of Using Delta Action

To formalize the problem, we start by assuming one-step consistency between real and simulated dynamics:

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

Under this assumption, one-step matching leads to the condition:

$$\pi(s) + \pi^\Delta(s, \pi(s)) = \hat{\pi}(s), \tag{1}$$

$$\Rightarrow \pi(s) = \hat{\pi}(s) - \pi^\Delta(s, \pi(s)). \tag{2}$$

To solve Equation (2), we consider:

1) **Fixed-Point Iteration**: We initialize $y_0 = \hat{\pi}(s)$ and iteratively update:

$$y_{k+1} = \hat{\pi}(s) - \pi^\Delta(s, y_k), \tag{3}$$

where y_k converges to a solution after K iterations.

2) **Gradient-Based Optimization**: Define the loss function:

$$l(y) = \|y + \pi^\Delta(s, y) - \hat{\pi}(s)\|^2. \tag{4}$$

A gradient descent method minimizes this loss to solve for y.

These methods approximate $\pi(s)$, but suffer from OOD issues when trained on limited trajectories. RL fine-tuning, in contrast, directly optimizes $\pi(s)$ for real-world deployment, resulting in superior performance.

Problem of One-Step Matching. Note that Equation (2) is derived from the one-step matching assumption (i.e., $\pi(s) + \pi^\Delta(s, \pi(s)) = \hat{\pi}(s)$). For multi-step matching, one has to differentiate through f^{sim}, which is, in general, intractable. Therefore, both fixed-point iteration and gradient-based optimization assume one-step matching. This also explains the advantages of RL-based fine-tuning: it effectively performs a gradient-free multi-step matching procedure.