2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Towards Adaptable Humanoid Control via Adaptive Motion Tracking

Tao Huang[1,2] Huayi Wang[1,2] Junli Ren[2] Kangning Yin[1,2] Zirui Wang[2] Xiao Chen[2]
Feiyu Jia[2] Wentao Zhang[2] Junfeng Long[2] Jingbo Wang[2,†] Jiangmiao Pang[2,†]
[1]Shanghai Jiao Tong University [2]Shanghai AI Laboratory [†]Equal Advising
Website: adamimic.github.io Code: https://github.com/InternRobotics/AdaMimic

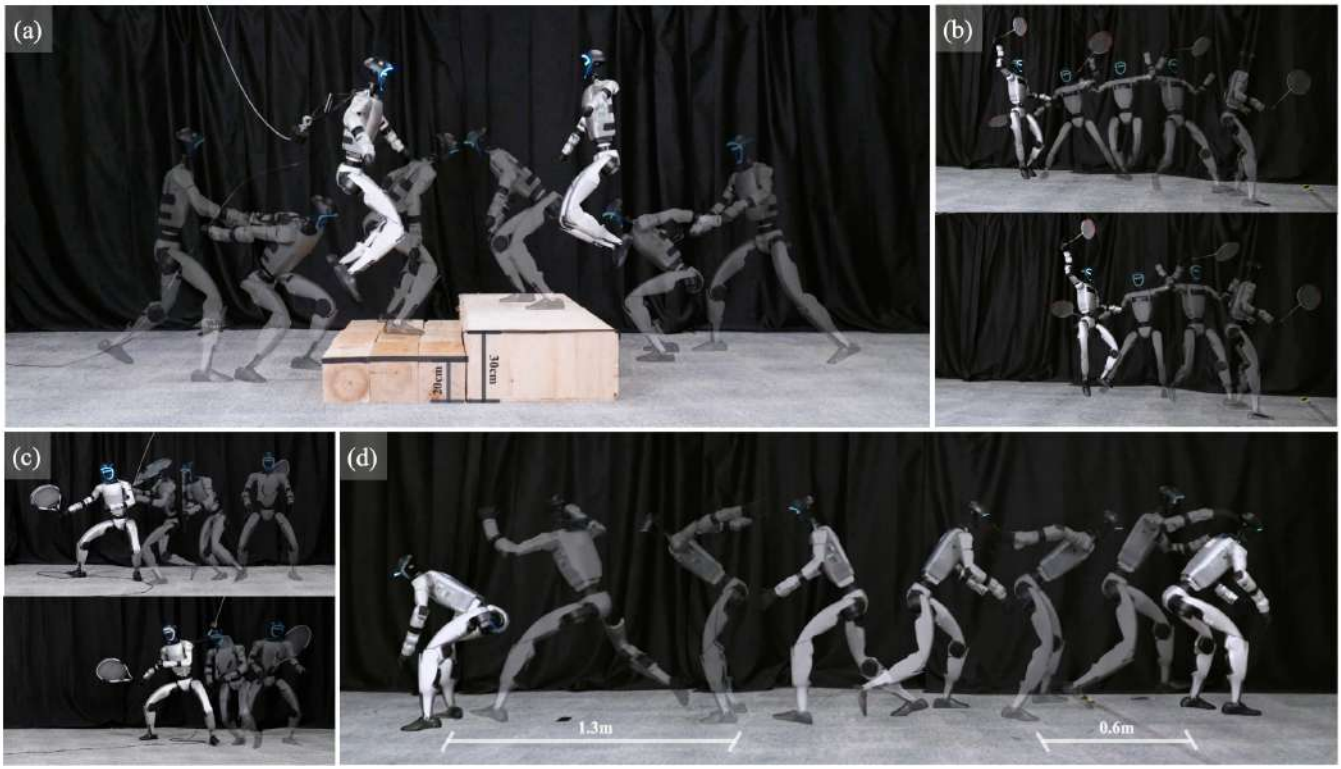

Fig. 1: Overview. Our method, **AdaMimic** (adaptive motion tracking), achieves agile humanoid whole-body adaptation from only a single reference motion while consistently preserving the underlying motion patterns. This enables adaptable control across diverse tasks, as illustrated by the varied outcomes: (a) higher jumping height, (b) extended movement in badminton hitting, (c) extended movement in tennis hitting, and (d) longer jumping distance.

***Abstract*— Humanoid robots are envisioned to adapt demonstrated motions to diverse real-world conditions while accurately preserving motion patterns. Existing motion prior approaches enable well adaptability with a few motions but often sacrifice imitation accuracy, whereas motion-tracking methods achieve accurate imitation yet require many training motions and a test-time target motion to adapt. To combine their strengths, we introduce AdaMimic, a novel motion tracking algorithm that enables adaptable humanoid control from a single reference motion. To reduce data dependence while ensuring adaptability, our method first creates an augmented dataset by sparsifying the single reference motion into keyframes and applying light editing with minimal physical assumptions. A policy is then initialized by tracking these sparse keyframes to generate dense intermediate motions, and adapters are subsequently trained to adjust tracking speed and refine low-level actions based on the adjustment, enabling flexible time warping that further improves imitation accuracy and adaptability. We validate these significant improvements in our approach in both simulation and the real-world Unitree G1 humanoid robot in multiple tasks across a wide range of adaptation conditions (Fig. 1). Videos and code are available on our project page.**

I. Introduction

Humans are good at acquiring whole-body skills by first mimicking experts and then adapting to new situations. For example, a tennis player reproduces the expert stroke pattern given different ball positions. Humanoid robots are expected to have a similar capability: adapt from reference motions to diverse real-world conditions, while accurately imitating motion patterns. We refer to this ability as adaptable humanoid control from reference motions.

Approaches to this motion-based adaptable control fall into two main paradigms. Methods that incorporate reference motions as prior into reinforcement learning (RL) enable adaptation beyond the data [1–4], but they often sacrifice imitation accuracy or require extensive reward tuning. Motion tracking methods, in contrast, can reproduce reference motions accurately with lower reward engineering burden [5–14]. However, their adaptability may be limited by the dependence on large-scale training motions to cover diverse

conditions and on possibly unavailable reference motions for each test-time deployment [15, 16]. How to combine the strengths of both paradigms—accurate imitation and broad adaptability—remains an open challenge.

In this work, we take an initial step towards this challenge by introducing AdaMimic, novel motion tracking algorithm that enables humanoid robots to adapt from a single reference motion while accurately preserving motion patterns. To reduce data dependence while supporting adaptation, we first generate an augmented dataset by sparsifying the reference motion into keyframes and editing a few keyframes with minimal physical assumptions, preserving the essential local pattern of the reference data. In the first stage, the policy is trained to track these sparse keyframes, producing dense intermediate motions that maintain local patterns to the reference. In a second stage, two adapters are jointly learned: a phase adapter that modulates motion speed, and a tracking adapter that compensates for the low-level actions based on the adjustment. This two-stage design enables flexible time warping [17, 18], enhancing both imitation accuracy and adaptability. The resulting policies can be deployed on hardware without requiring additional reference motions.

Extensive evaluations across diverse tasks and conditions, both in simulation and on the real-world Unitree G1 humanoid robot, demonstrate that AdaMimic outperforms existing methods. We overview of its real-world performance in Fig. 1 and summarize our core contributions as follows:

- We introduce AdaMimic, a novel motion tracking algorithm that enables humanoid robots to adapt from a single reference motion while accurately preserving key patterns.
- We validate the effectiveness of AdaMimic in extensive simulations, demonstrating significantly improved imitation accuracy and adaptability than existing methods.
- We deploy the trained policies on the real-world Unitree G1 humanoid robot, showing good performance across wide adaptation conditions in multiple tasks.

II. Related Work

A. Adaptable Humanoid Control

Adaptation to diverse real-world conditions is crucial in humanoid control, spanning both locomotion [20–27] and whole-body manipulation [28–35]. Approaches to these tasks often rely on sophisticated reward engineering and carefully designed sim-to-real transfer pipelines. In parallel, another line of work leverages human motion references to acquire adaptable whole-body skills [1–4, 13]. Their methods typically utilize motion data as priors, prioritizing adaptability over the strict preservation of original motion patterns. In contrast, our work emphasizes accurate tracking alongside adaptation.

B. Humanoid Motion Tracking

Motion tracking is proven effective for accurate imitation of a single reference motion [5–8]. However, its adaptability is often constrained by the scale of data. Recent advances address this limitation by training from massive public datasets [9–16, 36, 37]. While such approaches demonstrate impressive generalization, they typically require substantial data curation, rely on teleoperation or pre-collected reference motions during deployment, and may face challenges in maintaining tracking accuracy for highly agile behaviors. In contrast, our work explores how adaptive tracking can be achieved in agile tasks from only a single reference motion, without the need for additional pre-collected trajectories.

C. Motion Adaptation

A classical approach to adapting humanoid motions is motion editing, which can be achieved either through spacetime constraints [17, 38, 39] or data-driven generation [40, 41]. While both paradigms enable reusing a single motion as input, they often face limitations in physical plausibility caused by over-assumed constraints [42] or under-assumed dynamics [43], limiting real-world deployment. This has motivated physics-based approaches that integrate simulation and reinforcement learning to improve motion plausibility [44–46]. However, such methods often depend on large-scale datasets for interpolation or rule-based plausibility heuristics. Inspired by these two lines of work, we explore an alternative strategy: selecting and editing keyframes from a single motion, and then leveraging RL-based tracking to generate physically plausible in-between motions.

III. Background: Humanoid Motion Tracking

We formulate humanoid motion tracking as a goal-conditioned reinforcement learning (RL) problem within the framework of a Markov decision process (MDP). The objective is to learn a tracking policy π_{track} that accurately reproduces a reference motion $\hat{\boldsymbol{q}}$.

1) Motion representation: The retargeted reference motion $\hat{\boldsymbol{q}}$ is sampled from a reference dataset $\mathcal{D}_{\text{ref}}^{\text{init}}$, which initially contains a single motion [5, 6]. Each motion trajectory consists of global poses $\hat{\boldsymbol{q}}^{\text{global}}$ (*e.g.*, position and orientation of robot bodies) and local poses $\hat{\boldsymbol{q}}^{\text{local}}$ (*e.g.*, joint angles). A normalized phase variable $\phi \in [0, 1]$ parameterizes the whole reference motion: $\hat{\boldsymbol{q}}_\phi = \left(\hat{\boldsymbol{q}}_\phi^{\text{global}}, \hat{\boldsymbol{q}}_\phi^{\text{local}}\right)$, which advances discretely at each timestep k as:

$$\phi_k = \phi_{k-1} + \Delta\phi_k, \quad \Delta\phi_k = \Delta t \,/\, T_{\hat{\boldsymbol{q}}}, \tag{1}$$

where Δt is typically the simulation timestep and $T_{\hat{\boldsymbol{q}}}$ is the duration of the reference motion.

2) Observations and actions: At timestep k, the observation is defined as $\boldsymbol{o}_k = \left[\hat{\boldsymbol{q}}_{\phi_k}, \boldsymbol{q}_{\phi_k}, \dot{\boldsymbol{\theta}}_k, \dot{\boldsymbol{\omega}}_k, \phi_k\right]$, where $\boldsymbol{q}_{\phi_k}$ denotes current global and local robot poses, $\dot{\boldsymbol{\theta}}$ the joint velocities, and $\dot{\boldsymbol{\omega}}_k$ the base angular velocity. The policy produces a tracking action $\boldsymbol{a}_k \sim \pi_{\text{track}}(\cdot|\boldsymbol{o}_k, \Delta\phi_k)$ conditioned on the current observation, the reference motion, and the fixed phase interval. The action is the target of a PD controller during a control period.

3) Rewards and objectives: At timestep k, the rewards $\boldsymbol{r}_k = (r_k^{\text{global}}, r_k^{\text{local}})$ are aggregated from multiple terms that evaluate different aspects of tracking performance. For clarity, we group them into two categories: global-level rewards r_k^{global} and local-level rewards r_k^{local}. The former encourages accurate tracking of global trajectories, while

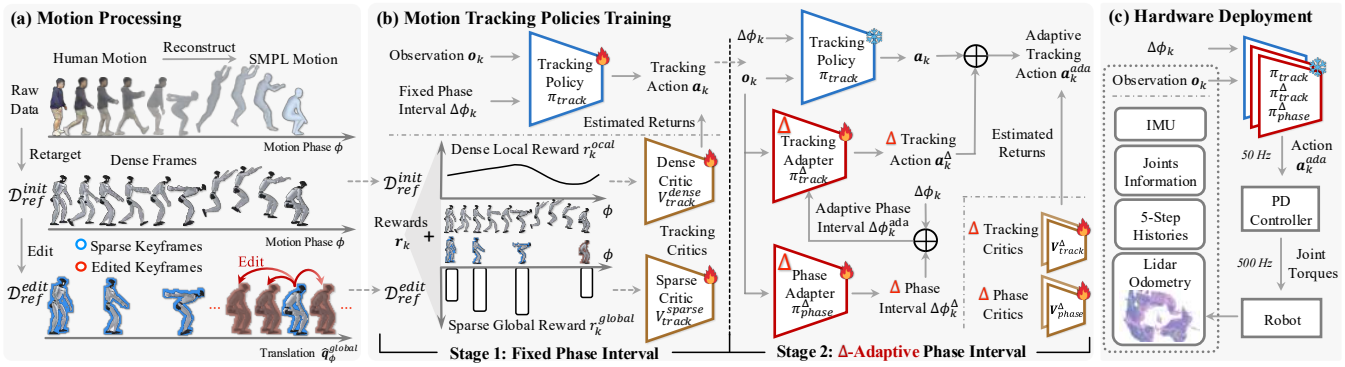

Fig. 2: Method overview. (a) Human motions are reconstructed into SMPL motions via GVHMR [19] and retargeted to the humanoid robot. Sparse keyframes are then selected and edited to form an augmented dataset for adaptive tracking. (b) Based on this dataset, AdaMimic first trains a tracking policy with fixed phase intervals and double critics for sparse global tracking and dense local tracking rewards, followed by phase and tracking adapters that enable effective time warping for improved tracking performance. (c) The resulting policies can be directly deployed on the real Unitree G1 robot.

the latter focuses on matching local motion patterns. The tracking policy π_{track} is then optimized to maximize the expected return:

$$\max_{\pi_{\text{track}}} \mathbb{E}_{\pi_{\text{track}}} \left[\sum_{k=0} \gamma^k \left(\boldsymbol{w} \cdot \boldsymbol{r}_k \right) \middle| \Delta\phi_k, \hat{\boldsymbol{q}} \sim \mathcal{D}_{\text{ref}}^{\text{init}} \right], \quad (2)$$

where $\boldsymbol{w}$ denotes the weighting between the two reward groups, and $\gamma \in [0, 1)$ is the discount factor.

IV. Method: Adaptive Motion Tracking

A. Problem Reformulation

The key insight of adaptive motion tracking is to extend the standard motion tracking problem by allowing variations in global trajectories while strictly preserving the local motion pattern of a given motion. Formally, starting from an initial reference dataset $\mathcal{D}_{\text{ref}}^{\text{init}} = \{\hat{\boldsymbol{q}}\}$ that contains a single motion trajectory, we assume access to an augmented dataset $\mathcal{D}_{\text{ref}}^{\text{edit}} = \{\hat{\boldsymbol{q}}_i^{\text{edit}}\}_{i=0}^{M}$. Each edited motion $\hat{\boldsymbol{q}}_i^{\text{edit}}$ shares the same local joint trajectories as the original motion:

$$\hat{\boldsymbol{q}}_{i,\phi_k}^{\text{edit, local}} = \hat{\boldsymbol{q}}_{\phi_k}^{\text{local}}, \quad \forall\, \hat{\boldsymbol{q}}_i^{\text{edit}} \in \mathcal{D}_{\text{ref}}^{\text{edit}}, \ \forall\, \phi_k \in [0, 1], \quad (3)$$

while the global component $\hat{\boldsymbol{q}}_{\phi_k}^{\text{edit, global}}$ may vary to reflect different displacements or base translations. For clarity, we interchangeably refer to $\hat{\boldsymbol{q}}_i^{\text{edit}}$ and $\hat{\boldsymbol{q}}$ without causing ambiguity in the rest of the paper.

The tracking policy π_{track} is then optimized to reproduce motions sampled from $\mathcal{D}_{\text{ref}}^{\text{edit}}$:

$$\max_{\pi_{\text{track}}} \mathbb{E}_{\pi_{\text{track}}} \left[\sum_{k=0} \gamma^k \left(\boldsymbol{w} \cdot \boldsymbol{r}_k \right) \middle| \Delta\phi_k, \hat{\boldsymbol{q}} \sim \mathcal{D}_{\text{ref}}^{\text{edit}} \right]. \quad (4)$$

This formulation abstracts away the specific mechanism for constructing $\mathcal{D}_{\text{ref}}^{\text{edit}}$, which will be discussed below. Adaptation is achieved by specifying a different reference motion.

B. Keyframing and Editing: $\mathcal{D}_{\text{ref}}^{\text{init}} \rightarrow \mathcal{D}_{\text{ref}}^{\text{edit}}$

A classical approach to construct $\mathcal{D}_{\text{ref}}^{\text{edit}}$ is to sparsely edit keyframes and interpolate the intermediate frames under trajectory-level consistency constraints [17, 38], denoted as $\mathcal{D}_{\text{ref}}^{\text{rule}}$. However, such optimization-based methods may yield physically implausible motions, which may impede hardware deployment (see Fig. 7), due to the absence of dynamics [42, 47]. Inspired by these works, we also adopt a keyframe-based editing scheme to preserve the global motion structure, but go beyond their limitations by using RL to generate more dynamically plausible intermediate frames, drawing from the ideas of [44, 45, 48].

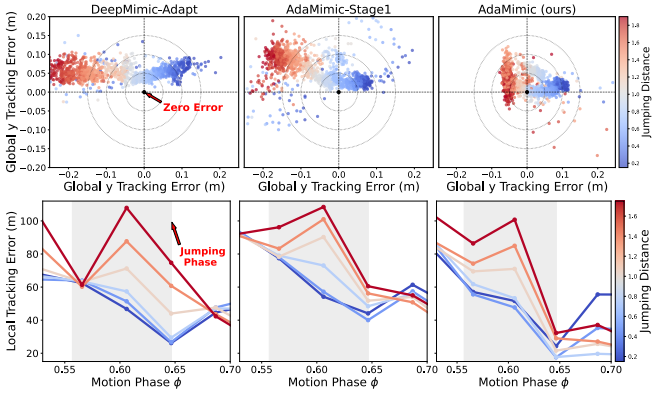

Fig. 3: Motivations of keyframing and adapters. (Top) Global tracking errors at the landing moment of far jumping indicate that AdaMimic outperforms baselines—DeepMimic-Adapt, which augments motions with rules, and AdaMimic-Stage1, which tracks at a fixed speed—by incorporating sparse keyframes and adaptive time warping to improve physical plausibility. (Bottom) Local tracking errors further verify the improvements.

Concretely, we select N keyframes, some of which are associated with semantic contexts (e.g., start, take-off, and landing in far jumping), and denote their phases as $\boldsymbol{\Phi}^{\text{key}} = \{\phi_0^{\text{key}}, \phi_1^{\text{key}}, \ldots, \phi_{N-1}^{\text{key}}\}$. A subset $\boldsymbol{\Phi}^{\text{edit}} \subset \boldsymbol{\Phi}^{\text{key}}$ is then chosen for editing, with the principle of introducing as few physical assumptions as possible. For each edited keyframe at phase $\phi_k \in \boldsymbol{\Phi}^{\text{edit}}$, we apply a transformation to its global pose based on a variable ψ_i (e.g., jumping distance):

$$\hat{\boldsymbol{q}}_{i,\phi_k}^{\text{edit, global}} = f_{\text{edit}}\left(\hat{\boldsymbol{q}}_{\phi_k}^{\text{global}}, \psi_i\right), \quad \forall\, \phi_k \in \boldsymbol{\Phi}^{\text{edit}}, \quad (5)$$

while keeping the local joint path unchanged. This editing function f_{edit} is task-dependent; for instance, in far jumping, it may translate post-landing frames forward or backward. The resulting set of edited keyframes forms $\mathcal{D}_{\text{ref}}^{\text{edit}}$, which serves as reference motions for adaptive motion tracking. The whole editing process is illustrated in Fig. 4.

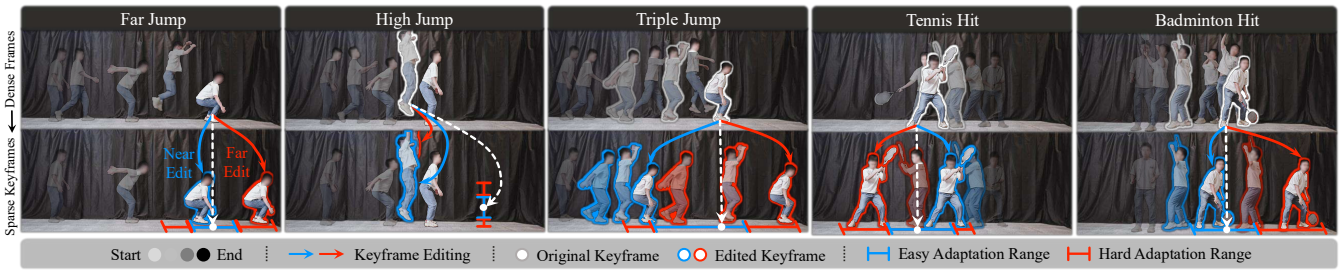

Fig. 4: (Top) Task visualization: Five representative motions used as input for humanoid retargeting. **(Bottom) Keyframing and editing:** Sparse keyframes are extracted from each motion, and few selected ones are further edited to enable adaptation. Colors denote adaptation difficulty relative to the original keyframe (gray): blue indicates easy adaptation cases, while red indicates hard adaptation cases. The adaptation ranges are presented in Table I.

TABLE I: Adaptation ranges of seven tasks during training and testing.

Task	Raw Data	Simulation Train/Test (m)		Hardware Test (m)	
		Easy	Hard	Easy	Hard.
Far Jump →	1.1m	[0.7, 1.6]	[0.2, 0.7] ∪	{0.7, 1.0}	{1.2, 0.4}
High Jump ↑	0.2m	[0.1, 0.35]	[0.05, 0.1] ∪ [0.35, 0.6]	{0.1, 0.2}	{0.3, 0.4}
Triple Jump →	2.4m	[1.65, 3.15]	[1.2, 1.65] ∪ [3.15, 4.2]	{2.1, 2.7}	{1.5, 3.3}
Step Jump ↕	0.2m	[0.05, 0.25]	[0.25, 0.5]	{0.2}	{0.3}
Tennis →	1.0m	[0.8, 1.2]	[1.2, 1.7]	{0.8, 1.2}	{1.4, 1.6}
Badminton ↘	1.3m	[1.3, 1.9]	[1.9, 2.5]	{1.3, 1.6}	{1.9, 2.2}

Remark: The current scope of tasks we consider is limited to those where the keyframes $\mathbf{\Phi}^{\text{key}}$ and editing function f_{edit} can be defined in a relatively straightforward and task-specific way. Extending to more tasks requires developing more general mechanisms for constructing $\mathbf{\Phi}^{\text{key}}$ and f_{edit}, which we leave as an important direction for future work.

C. Stage 1: Motion Tracking with Fixed Phase Interval

Given the edited motions, we first employ existing motion-tracking algorithms [5, 6] to train a tracking policy π_{track} using a fixed phase interval $\Delta\phi$. This stage aims to provide the agent with an initial capability of imitating the edited keyframe motions under a simple and stable training setup.

1) Sparse global reward: To ensure global-space motion alignment, we design a sparse global reward function:

$$r_k^{\text{global}} = \mathcal{R}_{\text{track}}^{\text{global}}(\boldsymbol{q}_{\phi_{k+1}}^{\text{global}}, \hat{\boldsymbol{q}}_{\phi_{k+1}}^{\text{global}}) \cdot \mathbb{1}(\phi_{k+1} \in \mathbf{\Phi}^{\text{key}}), \quad (6)$$

which is only activated when the current phase matches one of the keyframe phases $\mathbf{\Phi}^{\text{key}}$. This design avoids over-constraining the motion in global space and instead enforces accurate alignment only at sparse but crucial keyframes. To stabilize training under this sparse signal, we follow prior works [25, 44] and introduce a separate value function $V_{\text{track}}^{\text{sparse}}$ to better estimate the return from such sparse rewards.

2) Dense local reward: In addition, we design a dense local reward to preserve the local joint-space consistency with the reference motion. This reward encourages the reproduced motion to mimic fine-grained patterns from the references:

$$r_k^{\text{local}} = \mathcal{R}_{\text{track}}^{\text{local}}(\boldsymbol{q}_{\phi_{k+1}}^{\text{local}}, \hat{\boldsymbol{q}}_{\phi_{k+1}}^{\text{local}}), \quad (7)$$

A separate value function $V_{\text{track}}^{\text{local}}$ is employed to estimate the return induced by this dense signal, complementing the sparse global reward. The two value functions $\boldsymbol{V}_{\text{track}} = (V_{\text{track}}^{\text{sparse}}, V_{\text{track}}^{\text{dense}})$ are optimized separately following [49, 50]. All reward functions are listed in Table II.

TABLE II: Reward functions. The tracking rewards largely follow [6] and are categorized into sparse and dense reward groups, with additional regularization rewards to ensure hardware deployability.

Term	Weight	Term	Weight
Sparse Reward → $\boldsymbol{V}^{\text{sparse}}$			
Global body position	10	Global body rotation	5
Global feet position	10	Termination	-200
Dense Reward → $\boldsymbol{V}^{\text{dense}}$			
Local body position	0.75	Local body rotation	0.5
Local DoF position	0.75	Feet orientation	$-5e^{-2}$
DoF acceleration	$-2.5e^{-7}$	DoF velocity	$-5e^{-4}$
Action rate	$-5e^{-1}$	Smoothness	$-1e^{-2}$
Torques	$-1e^{-6}$	Torque limits	5
DoF position limits	-10	DoF velocity limits	-5

D. Stage 2: Adapters Learning with Adaptive Phase Interval

While stage 1 provides a good policy initialization, its adaptation ability to edited motions remains limited. We posit that the fixed phase interval is a key limitation. As illustrated in Fig. 3, this rigidity results in substantial global and local errors, especially when the required adaptation is large. In practice, such errors may manifest as unnatural pacing or artifacts such as unstable landings.

1) Phase adapter $\pi_{\text{phase}}^{\Delta}$: This motivates our design of the phase adapter $\pi_{\text{phase}}^{\Delta}$. It is inspired by the time-warping mechanism in classical motion path editing works, where motions are temporally re-parameterized to preserve naturalness and pacing under spatial edits [18, 45, 48]. Formally, the adapter takes the observation as input and outputs a delta phase interval $\Delta\phi_k^{\Delta}$, which is added to the base interval $\Delta\phi_k$ to obtain an adaptive phase interval $\Delta\phi_k^{\text{ada}}$:

$$\Delta\phi_k^{\text{ada}} = \Delta\phi_k + \Delta\phi_k^{\Delta}, \quad \Delta\phi_k^{\Delta} \sim \pi_{\text{phase}}^{\Delta}(\cdot|\boldsymbol{o}_k). \quad (8)$$

2) Tracking adapter $\pi_{\text{track}}^{\Delta}$: Given the adaptive phase interval, we need a tracking adapter $\pi_{\text{track}}^{\Delta}$ to compensate for the tracking action to track the next-step reference motion:

$$\boldsymbol{a}_k^{\text{ada}} = \boldsymbol{a}_k + \Delta\phi_k^{\Delta} \cdot \boldsymbol{a}_k^{\Delta}, \quad \boldsymbol{a}_k^{\Delta} \sim \pi_{\text{track}}^{\Delta}(\cdot|\boldsymbol{o}_k, \Delta\phi_k^{\text{ada}}), \quad (9)$$

where the scaling by $\Delta\phi_k^{\Delta}$ ensures that when the delta phase interval is zero, the adaptive action degenerates to the original tracking action. This design eases optimization empirically.

3) Optimization: During training, the two adapters $\boldsymbol{\pi}^{\Delta} = (\pi_{\text{phase}}^{\Delta}, \pi_{\text{track}}^{\Delta})$ are paired with separate double critics $\boldsymbol{V}^{\Delta} = (\boldsymbol{V}_{\text{phase}}^{\Delta}, \boldsymbol{V}_{\text{track}}^{\Delta})$ to estimate the sparse and dense rewards described in Section IV-C. The overall objective is

$$\max_{\boldsymbol{\pi}^{\Delta}} \mathbb{E}_{\boldsymbol{\pi}^{\Delta}} \left[\sum_{k=0} \gamma^k (\boldsymbol{w} \cdot \boldsymbol{r}_k) \,\middle|\, \Delta\phi_k, \hat{\boldsymbol{q}} \sim \mathcal{D}_{\text{ref}}^{\text{edit}} \right]. \quad (10)$$

TABLE III: Main simulation results. AdaMimic is compared against multiple baselines adapted to our problem setting and its ablated versions in a fair setup. Results report mean and standard deviation over three evaluations, each comprising more than ten thousand simulation episodes.

Comparison Methods	Components			Easy Adaptation				Hard Adaptation				Overall			
(a) Baselines	r^{global}	$\mathcal{D}_{\text{ref}}$	$\Delta\phi^{\text{ada}}$	Success ↑	$E^{\text{dense}}_{\text{l-bpe}}$ ↓	$E^{\text{sparse}}_{\text{g-bpe}}$ ↓	$E^{\text{dense}}_{\text{smth}}$ ↓	Success ↑	$E^{\text{dense}}_{\text{l-bpe}}$ ↓	$E^{\text{sparse}}_{\text{g-bpe}}$ ↓	$E^{\text{dense}}_{\text{smth}}$ ↓	Success ↑	$E^{\text{dense}}_{\text{l-bpe}}$ ↓	$E^{\text{sparse}}_{\text{g-bpe}}$ ↓	$E^{\text{dense}}_{\text{smth}}$ ↓
AMP-Style	Sparse	$\mathcal{D}^{\text{init}}_{\text{ref}}$	•	95.5%$_{\pm 0.1\%}$	44.5$_{\pm 0.0}$	211.2$_{\pm 0.1}$	19.1$_{\pm 0.3}$	70.3%$_{\pm 0.0\%}$	44.5$_{\pm 0.0}$	247.9$_{\pm 0.1}$	22.3$_{\pm 0.3}$	82.7%$_{\pm 0.0\%}$	44.5$_{\pm 0.0}$	229.8$_{\pm 0.2}$	20.7$_{\pm 0.3}$
AMP-Mimic	Sparse	$\mathcal{D}^{\text{init}}_{\text{ref}}$	•	96.8%$_{\pm 0.0\%}$	35.8$_{\pm 0.0}$	164.4$_{\pm 0.2}$	19.9$_{\pm 0.4}$	62.3%$_{\pm 0.0\%}$	35.7$_{\pm 0.0}$	190.3$_{\pm 0.1}$	21.3$_{\pm 0.4}$	79.1%$_{\pm 0.0\%}$	35.7$_{\pm 0.0}$	177.9$_{\pm 0.1}$	20.6$_{\pm 0.4}$
DeepMimic-NoAdapt	Dense	$\mathcal{D}^{\text{init}}_{\text{ref}}$	•	92.6%$_{\pm 0.0\%}$	36.6$_{\pm 0.0}$	205.1$_{\pm 0.1}$	17.6$_{\pm 0.6}$	**81.0%**$_{\pm 0.0\%}$	36.7$_{\pm 0.0}$	484.6$_{\pm 0.2}$	19.4$_{\pm 0.4}$	86.8%$_{\pm 0.0\%}$	36.6$_{\pm 0.0}$	351.8$_{\pm 0.2}$	18.6$_{\pm 0.5}$
DeepMimic-Adapt	Dense	$\mathcal{D}^{\text{rule}}_{\text{ref}}$	•	95.8%$_{\pm 0.0\%}$	33.3$_{\pm 0.0}$	123.6$_{\pm 0.0}$	**15.1**$_{\pm 0.3}$	74.8%$_{\pm 0.0\%}$	33.3$_{\pm 0.0}$	142.8$_{\pm 0.1}$	**16.7**$_{\pm 0.3}$	85.1%$_{\pm 0.0\%}$	33.3$_{\pm 0.0}$	133.5$_{\pm 0.1}$	**15.9**$_{\pm 0.3}$
DeepMimic-Adapt-$\Delta\phi^{\text{ada}}$	Dense	$\mathcal{D}^{\text{rule}}_{\text{ref}}$	✓	93.7%$_{\pm 0.0\%}$	38.7$_{\pm 0.0}$	170.9$_{\pm 0.0}$	17.2$_{\pm 0.5}$	70.0%$_{\pm 0.0\%}$	38.7$_{\pm 0.0}$	194.2$_{\pm 0.0}$	17.8$_{\pm 0.8}$	81.4%$_{\pm 0.1\%}$	38.8$_{\pm 0.0}$	182.8$_{\pm 0.0}$	17.4$_{\pm 0.5}$
AdaMimic-Dense	Dense	$\mathcal{D}^{\text{rule}}_{\text{ref}}$	✓	98.4%$_{\pm 0.0\%}$	36.3$_{\pm 0.0}$	107.6$_{\pm 0.0}$	17.6$_{\pm 0.1}$	78.0%$_{\pm 0.0\%}$	36.2$_{\pm 0.0}$	124.7$_{\pm 0.0}$	20.0$_{\pm 0.1}$	**88.0%**$_{\pm 0.0\%}$	36.2$_{\pm 0.0}$	115.0$_{\pm 0.0}$	18.8$_{\pm 0.1}$
AdaMimic (ours)	Sparse	$\mathcal{D}^{\text{edit}}_{\text{ref}}$	✓	**99.6%**$_{\pm 0.0\%}$	**30.3**$_{\pm 0.0}$	**87.9**$_{\pm 0.0}$	16.0$_{\pm 0.2}$	74.2%$_{\pm 0.0\%}$	**30.3**$_{\pm 0.0}$	**99.8**$_{\pm 0.1}$	17.3$_{\pm 0.2}$	86.8%$_{\pm 0.0\%}$	**30.3**$_{\pm 0.0}$	**94.8**$_{\pm 0.0}$	16.6$_{\pm 0.2}$
(b) Ablations	Stage 2	Freeze	$\Delta\phi^{\text{ada}}$	Success ↑	$E^{\text{dense}}_{\text{l-bpe}}$ ↓	$E^{\text{sparse}}_{\text{g-bpe}}$ ↓	$E^{\text{dense}}_{\text{smth}}$ ↓	Success ↑	$E^{\text{dense}}_{\text{l-bpe}}$ ↓	$E^{\text{sparse}}_{\text{g-bpe}}$ ↓	$E^{\text{dense}}_{\text{smth}}$ ↓	Success ↑	$E^{\text{dense}}_{\text{l-bpe}}$ ↓	$E^{\text{sparse}}_{\text{g-bpe}}$ ↓	$E^{\text{dense}}_{\text{smth}}$ ↓
AdaMimic-Stage1	•	•	•	96.7%$_{\pm 0.0\%}$	43.4$_{\pm 0.0}$	188.1$_{\pm 0.0}$	17.8$_{\pm 0.4}$	75.2%$_{\pm 0.0\%}$	43.4$_{\pm 0.0}$	211.8$_{\pm 0.1}$	18.7$_{\pm 0.4}$	85.7%$_{\pm 0.0\%}$	43.4$_{\pm 0.0}$	200.4$_{\pm 0.1}$	18.2$_{\pm 0.4}$
AdaMimic-Stage1-$\Delta\phi^{\text{ada}}$	•	•	✓	92.7%$_{\pm 0.0\%}$	45.3$_{\pm 0.0}$	195.0$_{\pm 0.1}$	19.8$_{\pm 0.3}$	**83.2%**$_{\pm 0.0\%}$	45.3$_{\pm 0.0}$	219.8$_{\pm 0.0}$	20.5$_{\pm 0.2}$	**88.0%**$_{\pm 0.0\%}$	45.3$_{\pm 0.0}$	208.1$_{\pm 0.0}$	20.2$_{\pm 0.2}$
AdaMimic-NoFreeze	✓	•	✓	85.0%$_{\pm 0.0\%}$	44.8$_{\pm 0.0}$	203.4$_{\pm 0.0}$	25.8$_{\pm 0.1}$	71.2%$_{\pm 0.0\%}$	43.6$_{\pm 0.0}$	224.2$_{\pm 0.0}$	26.9$_{\pm 0.1}$	77.8%$_{\pm 0.0\%}$	44.1$_{\pm 0.1}$	214.1$_{\pm 0.1}$	26.4$_{\pm 0.1}$
AdaMimic (default)	✓	✓	✓	**99.6%**$_{\pm 0.0\%}$	**30.3**$_{\pm 0.0}$	**87.9**$_{\pm 0.0}$	**16.0**$_{\pm 0.2}$	74.2%$_{\pm 0.0\%}$	**30.3**$_{\pm 0.0}$	**99.8**$_{\pm 0.1}$	17.3$_{\pm 0.2}$	86.8%$_{\pm 0.0\%}$	**30.3**$_{\pm 0.0}$	**94.8**$_{\pm 0.0}$	**16.6**$_{\pm 0.2}$

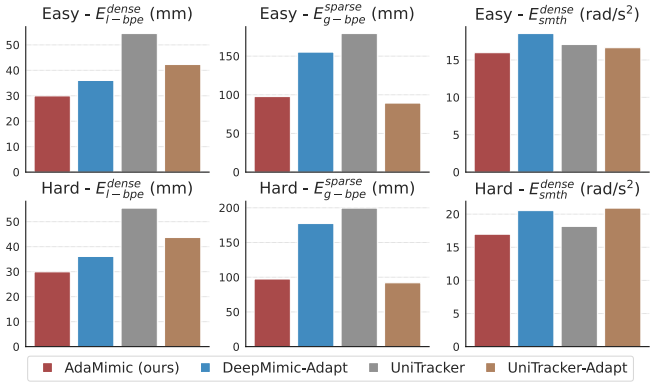

Fig. 5: Baselines trained with large-scale motion data. Across five tasks, AdaMimic achieves better performance than UniTracker [16] and its variant adapted with the rule-based motions from DeepMimic-Adapt. Besides, the results indicate that additional motion data does not provide UniTracker with obvious gains in specialization over DeepMimic-Adapt.

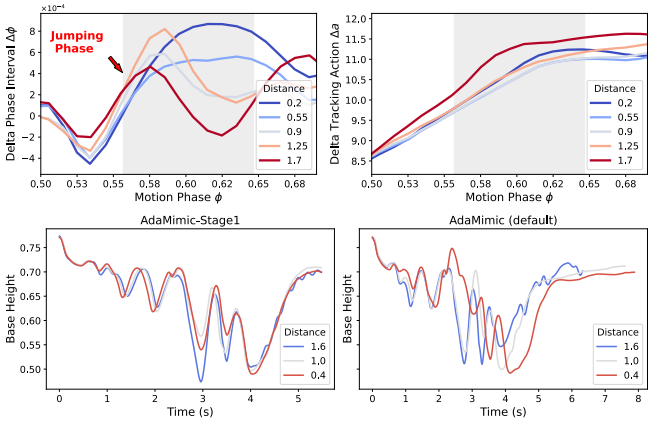

Fig. 6: Effectiveness of adapters. (Top) The phase interval and tracking action adapt to different far-jump distances, where longer distances correspond to extended airtime and larger action compensation, while shorter distances lead to reduced adjustments. (Bottom) With adapters, the policy performs effective time warping, adjusting motion speed to improve adaptation.

Finally, the resulting adaptive action $\boldsymbol{a}_k^{\text{ada}}$ is then applied to control the robot to perform adaptive motion tracking.

E. Real-World Deployment

We directly deploy the trained policies on the Unitree 29-DoFs G1 humanoid robot. For hardware stability, the waist pitch and roll joints are locked. To enhance robustness, the observation is augmented with a 5-step history. Global localization is realized by lidar odometry through FastLIO [51]. The policy, low-level control, and odometry modules operate at 50Hz, 500Hz, and 10Hz, respectively.

F. Implementation Details

The human videos are translated into SMPL motions using GVHMR [19] for retargeting. Training is conducted in the Isaac Gym simulator [52] with 4096 parallel environments, employing PPO [53] as the RL algorithm. Both the policy and value functions are parameterized as 3-layer MLPs. Each episode is initialized from a randomly sampled keyframe [5]. To improve stability for real-world deployment, we adopt L2C2 regularization [50, 54]. The reward weight vector $\boldsymbol{w}$ is set to $(1, 0.5)$ for the sparse and dense reward groups, respectively. The adaptive phase interval is defined as $\Delta\phi_k^{\Delta} \in [-0.75\Delta\phi_k, \phi_k]$. Finally, PD controllers are configured following [50] for improved simulation performance, while employing configurations from [2] during hardware deployment to improve safety and smoothness.

V. Experiments

A. Experimental Setup

1) Tasks: We record seven human videos as the evaluation task set, including five various jumping skills and two ball-hitting skills. These tasks are considered agile and difficult enough. Some representative tasks are visualized in Fig. 4 and their adaptation ranges are presented in Table I.

2) Comparison methods: We adapt the following baselines to our problem formulation:

- **Motion as priors:** Adversarial motion priors (AMP-Style; [1]), which use motions as a style regularizer combined with sparse keyframe tracking as the task reward. Its variant, AMP-Mimic, additionally conditions on phase information to better mimic the reference motion.
- **Motion tracking from target data:** DeepMimic [5], including (i) DeepMimic-NoAdapt, which tracks the original motion $\mathcal{D}^{\text{init}}_{\text{ref}}$, and (ii) DeepMimic-Adapt(-$\Delta\phi^{\text{ada}}$), which tracks the reference motions $\mathcal{D}^{\text{rule}}_{\text{ref}}$ generated with a classical linear motion editing method [48].
- **Motion tracking from prior motions:** UniTracker [16], trained on large-scale motion datasets, and its adapted variant (UniTracker-Adapt), fine-tuned on $\mathcal{D}^{\text{rule}}_{\text{ref}}$.

TABLE IV: **Main hardware results.** In the absence of accurate odometry, we report success rate, local joint tracking error, and smoothness to quantitatively compare AdaMimic with three representative baselines. The results indicate that AdaMimic achieves strong hardware deployability, particularly in challenging adaptation cases, benefiting from the proposed tracking objectives, motion editing, and adapters. '/' indicates a complete failure in that case.

Easy Adaptation	**Far Jump**			**High Jump**			**Triple Jump**			**Tennis Hit**			**Badminton Hit**		
	Succ.↑	$E^{\text{dense}}_{\text{l-dof}}$↓	$E^{\text{dense}}_{\text{smth}}$↓	Succ.↑	$E^{\text{dense}}_{\text{l-dof}}$↓	$E^{\text{dense}}_{\text{smth}}$↓	Succ.↑	$E^{\text{dense}}_{\text{l-dof}}$↓	$E^{\text{dense}}_{\text{smth}}$↓	Succ.↑	$E^{\text{dense}}_{\text{l-dof}}$↓	$E^{\text{dense}}_{\text{smth}}$↓	Succ.↑	$E^{\text{dense}}_{\text{l-dof}}$↓	$E^{\text{dense}}_{\text{smth}}$↓
AMP-Style	4/6	35.2$\pm$0.2	42.8$\pm$2.2	5/6	34.8$\pm$0.7	37.4$\pm$7.9	4/6	32.6$\pm$0.9	41.7$\pm$3.5	5/6	31.7$\pm$0.5	52.9$\pm$4.5	0/6	/	/
DeepMimic-Adapt	4/6	34.4$\pm$0.1	41.5$\pm$1.0	1/6	32.8$\pm$0.0	65.6$\pm$0.0	**6/6**	34.0$\pm$0.9	49.0$\pm$3.3	**6/6**	32.1$\pm$0.1	36.5$\pm$1.0	**6/6**	30.4$\pm$0.2	50.2$\pm$2.2
AdaMimic-Stage1	**6/6**	**33.8**$\pm$0.2	43.7$\pm$2.8	**6/6**	32.7$\pm$0.1	35.7$\pm$1.5	**6/6**	32.6$\pm$0.6	54.4$\pm$6.3	**6/6**	31.6$\pm$0.1	33.9$\pm$0.9	5/6	**28.6**$\pm$0.2	44.5$\pm$3.5
AdaMimic (our)	5/6	35.2$\pm$0.7	**38.7**$\pm$1.4	**6/6**	**30.5**$\pm$0.3	**28.7**$\pm$3.4	**6/6**	**30.7**$\pm$0.3	**31.3**$\pm$6.7	**6/6**	**30.7**$\pm$0.2	**31.9**$\pm$1.8	**6/6**	28.7$\pm$0.1	**36.4**$\pm$2.0
Hard Adaptation	**Far Jump**			**High Jump**			**Triple Jump**			**Tennis Hit**			**Badminton Hit**		
	Succ.↑	$E^{\text{dense}}_{\text{l-dof}}$↓	$E^{\text{dense}}_{\text{smth}}$↓	Succ.↑	$E^{\text{dense}}_{\text{l-dof}}$↓	$E^{\text{dense}}_{\text{smth}}$↓	Succ.↑	$E^{\text{dense}}_{\text{l-dof}}$↓	$E^{\text{dense}}_{\text{smth}}$↓	Succ.↑	$E^{\text{dense}}_{\text{l-dof}}$↓	$E^{\text{dense}}_{\text{smth}}$↓	Succ.↑	$E^{\text{dense}}_{\text{l-dof}}$↓	$E^{\text{dense}}_{\text{smth}}$↓
AMP-Style	3/6	35.0$\pm$0.0	**31.8**$\pm$1.1	0/6	/	/	3/6	33.7$\pm$0.2	42.7$\pm$8.9	4/6	31.6$\pm$0.5	66.4$\pm$7.3	0/6	/	/
DeepMimic-Adapt	3/6	34.3$\pm$0.1	46.6$\pm$4.6	0/6	/	/	**6/6**	33.8$\pm$0.1	47.7$\pm$2.1	5/6	31.1$\pm$0.1	49.5$\pm$4.3	**6/6**	30.8$\pm$0.1	55.3$\pm$1.7
AdaMimic-Stage1	2/6	**34.1**$\pm$0.3	34.3$\pm$5.4	3/6	32.5$\pm$0.1	37.1$\pm$3.9	**6/6**	32.3$\pm$0.1	38.3$\pm$2.3	**6/6**	31.9$\pm$0.1	**40.9**$\pm$2.2	5/6	29.1$\pm$0.1	50.9$\pm$2.7
AdaMimic (our)	**5/6**	35.3$\pm$0.5	46.2$\pm$7.7	**5/6**	**31.3**$\pm$0.5	**28.6**$\pm$4.2	**6/6**	**31.5**$\pm$1.9	**31.7**$\pm$4.6	**6/6**	**30.8**$\pm$0.1	42.9$\pm$1.7	**6/6**	**28.9**$\pm$0.1	**44.5**$\pm$2.0

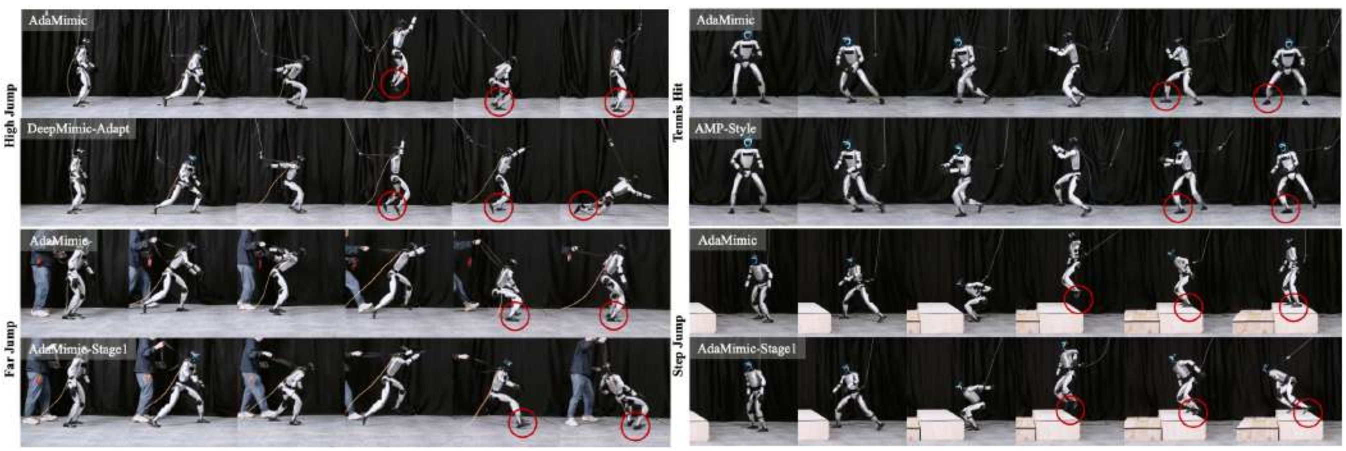

Fig. 7: Snapshots of real robot motions. Four representative groups of hardware demonstrations highlight the differences between AdaMimic and baseline methods: (1) DeepMimic-Adapt is constrained by physically implausible edited motions used for dense per-frame tracking, (2) AMP-Style exhibits jerky motions and limited imitation accuracy, and (3) AdaMimic-Stage1 yields hardware-unstable policies without the proposed adapters.

For fair comparison, we use the same reward functions and observation spaces across all methods, except for UniTracker, which cannot be directly transferred due to its special observations. For this baseline, we use its official implementation.

3) Evaluation metrics: We evaluate each method using four metrics. (1) **Success rate** considers an episode failed if the average body distance error exceeds 1m at any keyframe phase; unlike prior work, we emphasize keyframes and adopt a looser criterion under the sparse tracking setup. (2) **Tracking error** measures body distance error in two forms: the *sparse global error* ($E^{\text{sparse}}_{\text{g-bpe}}$, mm), averaged in the world frame at keyframe phases to assess global mimicking accuracy, and the *dense local error* ($E^{\text{dense}}_{\text{l-bpe}}$, mm), averaged in the root frame over all timesteps to assess local accuracy. (3) **Smoothness** ($E^{\text{dense}}_{\text{smth}}$, rad/s^2) is defined as the average joint acceleration over the entire episode. Results are reported as averages over three evaluation runs, each covering more than ten thousand episodes in IsaacGym.

B. Main Simulation Results

AdaMimic exhibit both great adaptation and tracking precision across all adaptation cases, as shown in Table III.

Comparison with AMP. The AMP-Style baseline demonstrates moderate adaptability but limited imitation accuracy, primarily due to the absence of phase information in its original formulation. Incorporating phase information alleviates this issue and yields improvements in both adaptability and precision. Nonetheless, the resulting motions remain jerky because of weak motion constraints, and overall performance still falls short of other adaptable methods and AdaMimic.

Comparison with DeepMimic. As expected, DeepMimic-noAdapt fails to adapt due to relying on a single reference motion. Its adaptable variants, DeepMimic-Adapt(-$\Delta\phi^{\text{ada}}$), achieve notable performance gains, highlighting the effectiveness and generality of our formulation. However, their tracking performance degrades substantially from easy to hard adaptation scenarios (see also example in Fig. 7), reflecting limitations imposed by the underlying motion path editing. In contrast, our method avoids such restrictive editing assumptions and instead leverages RL with keyframing to produce more physically plausible motions.

Comparison with UniTracker. As shown in Fig. 5, we observe that: (1) the pre-trained model, despite being trained on massive motion data, struggles to perform agile motions; (2) finetuning with $\mathcal{D}^{\text{rule}}_{\text{ref}}$ yields some gains but does not surpass DeepMimic-Adapt trained from scratch; and (3) both versions overall underperform AdaMimic. These findings further suggest that universal trackers could be made less dependent on pre-defined data during deployment by incorporating a more physically plausible motion editing module.

C. Analyses of Key Designs

Effectiveness of adapters. Table III and Fig. 3 quantify the benefits of adapters, while Fig. 6 reveals their mechanism. The top panel shows that phase intervals and delta action scale with jump distance: longer jumps lead to extended air-time and larger compensation, whereas shorter jumps require smaller adjustments, mostly concentrated around landing. Together, this decoupling of timing and correction enables motion speed adaptation without altering the motion pattern, resulting in lower tracking error, smoother landings, and higher success rates, especially in hard adaptation cases.

Keyframe editing outperforms per-frame editing. We observe that keyframe-based AdaMimic consistently outperforms its per-frame counterpart, AdaMimic-Dense, indicating the advantage of sparse keyframe selection for adaptation. Interestingly, the trend is reversed when comparing AdaMimic-Stage1 with DeepMimic-Adapt, suggesting that the combination of keyframing and adapters is crucial for achieving both precise tracking and great adaptation.

Two-stage training is necessary. Our motivation for the two-stage design arises from the observation that training with adaptive phase intervals (AdaMimic-Stage1-$\Delta\phi^{\text{ada}}$) but inference with fixed intervals can even degrade performance compared to training with fixed intervals (AdaMimic-Stage1). We hypothesize that this is due to increased optimization difficulty incurred by the varied phase interval, which can be substantially mitigated by our two-stage design.

Freezing the tracking policy is important. Finetuning the tracking policy during the second stage (AdaMimic-NoFreeze) noticeably degrades both smoothness and tracking accuracy. We attribute this degradation to increased optimization difficulty when updating the policy alongside adapters.

D. Main Hardware Results

We present snapshots of real robot motions to compare AdaMimic with representative baselines in Fig. 7. DeepMimic-Adapt fails in extreme cases, such as high jumps, because the rule-based augmented motions are physically inconsistent and cannot be reliably executed on hardware. AMP-Style shows noticeable instability in dynamic tasks like tennis hitting: motions are jerky, and forceful strikes lead to poor balance and uncoordinated execution. AdaMimic-Stage1, lacking the proposed adapters, exhibits large sim-to-real gaps; in particular, the ankle joints become highly unstable during forceful motions, reflecting inadequate temporal and action adaptation.

In contrast, AdaMimic combines sparse keyframes with phase and tracking adapters, enabling the policy to adjust motion timing and per-step actions while maintaining original motion patterns. This results in physically plausible and robust execution across all tasks. Quantitatively, as shown in Table IV, AdaMimic consistently achieves high success rates, reduced local joint errors, and smoother motions in both easy and hard adaptation scenarios, highlighting the effectiveness and generality of our method on the real robot.

VI. Conclusion

We have presented AdaMimic, a novel motion tracking framework for adaptable humanoid control from a single reference motion. Our framework addresses the limitations of prior methods, which either sacrifice tracking accuracy for adaptation or require large-scale reference motions for each adaptation condition. By leveraging augmented sparse keyframes and a two-stage training strategy with phase and tracking adapters, AdaMimic enables accurate imitation while extending adaptability to diverse tasks and conditions. Experimental results on the Unitree G1 humanoid robot demonstrate that our controllers achieve adaptable motions across many tasks, validating both the effectiveness and generality of our framework. Looking forward, AdaMimic holds promise for extending beyond the demonstrated scenarios to more complex and interactive whole-body skills, such as perceptive and professional ball games.

VII. Limitations and Future Directions

Wide task scope. Our method currently focuses on tasks with clear parametric forms (e.g., jump or strike distances), while many skills lack such representations. Extending to less structured tasks could broaden its applicability.

General motion editing mechanism. Keyframe selection and editing are manually specified, which restricts scalability. Developing automatic editing mechanisms could make the framework more general and efficient.

Utilization of massive motion data. Universal trackers trained on large datasets still underperform, even with finetuning. Better strategies to exploit massive motion data are valuable for stronger adaptation.

Adaptation for interactive tasks. Our framework executes motions without environment feedback, limiting interactivity. Incorporating perception will enable adaptive and responsive behaviors in interactive tasks such as ball games.

Acknowledgments

This work is funded in part by the National Key R&D Program of China (2022ZD0160201), and Shanghai Artificial Intelligence Laboratory.

References

[1] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *Transactions on Graphics (TOG)*, 2021.

[2] Q. Liao, T. E. Truong, X. Huang, G. Tevet, K. Sreenath, and C. K. Liu, "Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion," *ArXiv*, vol. abs/2508.08241, 2025.

[3] H. Xue, X. Huang, D. Niu, Q. Liao, T. Kragerud, J. T. Gravdahl, X. B. Peng, G. Shi, T. Darrell, K. Screenath, and S. S. Sastry, "Leverb: Humanoid whole-body control with latent vision-language instruction," *ArXiv*, vol. abs/2506.13751, 2025.

[4] A. Allshire, H. Choi, J. Zhang, D. McAllister, A. Zhang, C. M. Kim, T. Darrell, P. Abbeel, J. Malik, and A. Kanazawa, "Visual imitation enables contextual humanoid control," in *Conference on Robot Learning (CoRL)*, 2025.

[5] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *Transactions on Graphics (TOG)*, 2018.

[6] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbabu, C. Pan, Z. Yi, G. Qu, K. Kitani, J. Hodgins, L. J. Fan, Y. Zhu, C. Liu, and G. Shi, "Asap: Aligning simulation and real-world