Adversarial Locomotion and Motion Imitation for Humanoid Policy Learning

Jiyuan Shi [1*] **Xinzhe Liu** [1,2*] **Dewei Wang** [1,3*] **Ouyang Lu** [1,4] **Sören Schwertfeger** [2]
Chi Zhang [1] **Fuchun Sun** [5] **Chenjia Bai**[1†] **Xuelong Li** [1†]

[1]Institute of Artificial Intelligence (TeleAI), China Telecom
[2]ShanghaiTech University [3]University of Science and Technology of China
[4]Northwestern Polytechnical University [5]Tsinghua University

Abstract

Humans exhibit diverse and expressive whole-body movements. However, attaining human-like whole-body coordination in humanoid robots remains challenging, as conventional approaches that mimic whole-body motions often neglect the distinct roles of upper and lower body. This oversight leads to computationally intensive policy learning and frequently causes robot instability and falls during real-world execution. To address these issues, we propose Adversarial Locomotion and Motion Imitation (ALMI), a novel framework that enables adversarial policy learning between upper and lower body. Specifically, the lower body aims to provide robust locomotion capabilities to follow velocity commands while the upper body tracks various motions. Conversely, the upper-body policy ensures effective motion tracking when the robot executes velocity-based movements. Through iterative updates, these policies achieve coordinated whole-body control, which can be extended to loco-manipulation tasks with teleoperation systems. Extensive experiments demonstrate that our method achieves robust locomotion and precise motion tracking in both simulation and on the full-size Unitree H1-2 robot. Additionally, we release a large-scale whole-body motion control dataset featuring high-quality episodic trajectories from MuJoCo simulations. The project page is `https://almi-humanoid.github.io`.

1 Introduction

Humans exhibit diverse and expressive whole-body movements in various activities [1, 2]. For example, in dancing, humans depend on the lower body for stable movements and gait control, and the upper body executes precise actions to accomplish specific movements. This coordination allows for expressive and purposeful motions that are highly adaptable to different situations. However, achieving human-like whole-body coordination remains a highly challenging task for humanoid robots. Current methods employ motion re-targeting and Reinforcement Learning (RL) to learn a whole-body policy that can track human motions by taking the tracking errors as rewards and optimizing a whole-body policy to maximize such rewards [3, 4].

However, this kind of approach has significant limitations. First, due to the high number of Degrees of Freedom (DoF) in the humanoid robot, directly learning a whole-body control policy demands a complex reward structure and makes the training process highly expensive. Second, the differences between various motions, along with some human movements beyond a robot's physical capabilities, make it difficult for the RL policy to converge. In practice, since such whole-body learning approaches

[*]Equal Contribution [†]Correspondence to: Chenjia Bai (baicj@chinatelecom.cn)

prioritize precise motion tracking over balance maintenance, the policy often neglects the fundamental need for robot stability. Meanwhile, the poor stability of the lower body in turn affects the execution of upper-body motion and reveals challenges in real-world deployment, such as frequent robot falls. We identify that the main reason is that above methods do not separately consider the unique roles of the upper and lower bodies in motion learning, specifically, the lower body's role in robust locomotion and the upper body's task of precise motion imitation.

In this paper, we propose a novel framework, named *Adversarial Locomotion and Motion Imitation* (**ALMI**), that separately learns robust locomotion and motion imitation policies for the upper and lower bodies, respectively. Specifically, the lower body learns to follow different velocity commands while withstanding adversarial disturbances from the upper body, which leads to robust locomotion even when the upper-body movements significantly disrupt the stability. Conversely, the upper body learns to track reference motions accurately despite adversarial disturbances caused by lower-body instability, which leads to expressive motion imitation even when the lower body is commanded to move rapidly over uneven terrain. Through iterative updates of the upper and lower policies, these policies achieve coordinated whole-body control, which can be extended to loco-manipulation tasks with open-loop upper-body control via teleoperation systems. Unlike recent works [5–7] that adopt separate control for the upper and lower body, ALMI involves an adversarial training process to obtain coordinated behavior, which ensures the upper and lower policies converge to an equilibrium under mild conditions. Extensive experiments demonstrate that ALMI achieves robust locomotion and precise motion tracking in both simulation and on a full-size Unitree H1-2 humanoid robot.

Further, we construct a large-scale whole-body control dataset, named **ALMI-X**, featuring episodic trajectories for the Unitree H1-2 robot in MuJoCo simulations. Each episode is generated by the ALMI-acquired policy, where the lower body is controlled by the velocity command, and the upper body is controlled by joystick commands (i.e., desired joints) of the reference motion from the AMASS dataset [8]. We annotate language descriptions for each episode data according to both commands of the lower and upper bodies, e.g. *moving backward slowly and wave the left hand*. The ALMI-X dataset contains more than 80K trajectories with text commands and corresponding trajectories. We also give preliminary attempts to train a foundation model from the collected ALMI-X data, which serves an end-to-end whole-body control policy by leveraging a Transformer-based architecture.

Our main contributions are summarized as follows: (i) We propose a novel adversarial training framework (ALMI) for robust locomotion and precise motion imitation for humanoid robots, addressing the distinct roles of upper and lower body through separate policy learning. (ii) We create the first large-scale whole-body control dataset (ALMI-X) that integrates language descriptions with robot trajectories, facilitating the training of foundation models for humanoid whole-body control. (iii) We conduct extensive experiments to verify the effectiveness of the ALMI policy and the preliminary study for the humanoid foundation control model in both the simulation and the real world.

2 Preliminaries

In our work, we adopt an adversarial training framework for the policy learning of both the upper and lower bodies. Specifically, we consider an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}^l, \mathcal{A}^u, T, \gamma, r^l, r^u, P)$, where the lower and upper bodies share the same state space $\mathcal{S}$ while having different action space, that is, $\mathcal{A}^l$ and $\mathcal{A}^u$, respectively. r^l is the command-following reward for the locomotion policy (i.e., the lower body), and r^u is the tracking reward for motion-imitation policy (i.e., the upper body). $P(s'|s, a^l, a^u)$ is the transition probability function that denotes the probability of transitioning to the next state s' given the current state and the actions of both players. We aim to learn two policies, i.e., π^l and π^u, that control different actions for the lower and upper bodies, respectively.

We adopted the full-sized Unitree H1-2 robot in our work, which has 27 DoFs in total, and the policy controls 21 DoFs, excluding the 3 DoFs in each wrist of the hands. The state space is defined as $\boldsymbol{s} = (\boldsymbol{s}_t^{\text{prop}}, \boldsymbol{c}^l, \boldsymbol{\phi}_t, \boldsymbol{g}^u)$, where $\boldsymbol{s}_t^{\text{prop}} = [\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{\omega}_t, \boldsymbol{gv}_t, \boldsymbol{a}_{t-1}^l]$ is the proprioception of the robot, $\boldsymbol{c}^l = [\hat{v}_{x,t}, \hat{v}_{y,t}, \hat{\omega}_{\text{yaw},t}]$ is the velocity command for the lower-body, and $\boldsymbol{\phi}_t \in \mathbb{R}^2$ is the phase parameter at each time step, and $\boldsymbol{g}^u \in \mathbb{R}^9$ is the reference joint position for the upper body. In proprioception, the notation $\boldsymbol{q}_t \in \mathbb{R}^{21}, \dot{\boldsymbol{q}}_t \in \mathbb{R}^{21}, \boldsymbol{\omega}_t \in \mathbb{R}^3, \boldsymbol{gv}_t \in \mathbb{R}^3, \boldsymbol{a}_{t-1}^l \in \mathbb{R}^{12}$ is the joint position, the joint velocity, the angular velocity of the base, the projected gravity of the base, and the last action of the lower body, respectively.

For the lower body, the policy π^l gives an action $\boldsymbol{a}^l \in \mathbb{R}^{12}$ representing target joint positions of the lower body according to the proprioception, commands, and phase variables at each step, which are fed into the PD controller to calculate the joint torques. For the upper body, policy π^u takes proprioception and reference joint position as input, and the action $\boldsymbol{a}^u \in \mathbb{R}^9$ includes target positions of the 3 shoulder joints and 1 elbow joint per arm, along with the waist yaw joint. As a result, the two policies (π^l and π^u) control distinct action spaces of the robot, and share the same state space by masking the irrelevant commands for the different policy.

3 Methods

In this section, we first present the theoretical foundation for the adversarial learning framework, and then give practical algorithms for implementing such a framework for humanoid robots.

3.1 The Adversarial Learning Framework

Problem Formulation for learning π^l. We consider a two-player zero-sum Markov game [9, 10] to learn a robust lower-body policy π^l. At each time step, the two players (i.e., π^l and π^u) choose different actions (a^l and a^u), and the humanoid robot executes both actions to obtain the reward and the next state. In learning π^l, we consider the lower body as *agent*, and the upper body is *adversary* that causes adversarial disturbances to the locomotion policy. Thus, the lower-body policy receives the command-following reward as $r^l(s, a^l, a^u)$, and the upper-body policy obtains a negative reward $-r^l(s, a^l, a^u)$. Formally, the value function $V^l(\pi^l, \pi^u)$ is defined as

$$V^l(s, \pi^l, \pi^u) := \mathbb{E}_{\pi^l,\pi^u}\left[\sum\nolimits_{t=0}^{T} r^l(s_t, a^l, a^u)\big|s_0 = s\right], \tag{1}$$

where $\mathbb{E}_{\pi^l,\pi^u}$ is the under the trajectory distribution induced by π^l and π^u. Then we have the value function as $V^l_\rho(\pi^l, \pi^u) = \mathbb{E}_{s\sim\rho}[V^l(s, \pi^l, \pi^u)]$, which is defined by the expectation of accumulated locomotion reward r^l. Then the locomotion policy π^l is learned to *maximize* the value function to better follow commands in locomotion, while the upper body policy tries to *minimize* the value function, aiming to provide an effective disturbance to help learn a robust locomotion policy. Formally, the two players form a Markov game and there exists a Nash equilibrium such have

$$V^l_\rho(\pi^{l_*}, \pi^{u_*}) = \max_{\pi^l}\min_{\pi^u} V^l_\rho(\pi^l, \pi^u). \tag{2}$$

To solve this max-min problem, we adopt an independent RL optimization process for both players. Specifically, the *agent* obtains $[(s_0, a^l_0, r^l_0), \dots, (s_T, a^l_T, r^l_T)]$, and *adversarial* obtains $[(s_0, a^u_0, r^u_0), \dots, (s_T, a^u_T, r^u_T)]$ by executing each policy in the game to sample a trajectory, where each player is oblivious to the actions of the other player. The two players optimize their policies independently with policy gradients using their own experiences. Then the following theorem holds.

Theorem 3.1. *Given $\epsilon > 0$, suppose each policy has ε-greedy exploration scheme with factors of $\varepsilon_x \asymp \epsilon$ and $\varepsilon_x \asymp \epsilon^2$, under a specific two-timescale rule of the two players' learning-rate following the independent policy gradient, we have*

$$\max_{\pi^l}\min_{\pi^u} V_\rho(\pi^l, \pi^u) - \mathbb{E}\left[\frac{1}{N}\sum\nolimits_{i=1}^{N}\min_{\pi^u} V_\rho(\pi^u, \pi^{l(i)})\right] \le \epsilon \tag{3}$$

after N episodes, which results in a ϵ-approximate Nash equilibrium.

Intuitively, the max-min game for the lower and upper bodies leads to a robust locomotion policy (by maximizing V^l_ρ in the outer loop) even when the upper-body movements significantly disrupt the whole-body balance (by minimizing V^l_ρ in the inner loop). This adversarial learning process is guaranteed to converge to a ϵ-approximate Nash equilibrium according to Theorem 3.1.

Problem Formulation for learning π^u. Learning a precise motion imitation policy π^u follows a similar process by considering π^u as the *agent* and π^l as the *adversary*. In the Markov game, the upper body receives a motion tracking reward $r^u(s, a^l, a^u)$ and the lower body receives $-r^u(s, a^l, a^u)$, which aims to give adversarial disturbances for motion tracking. Then we define $V^u(s, \pi^l, \pi^u) := \mathbb{E}_{\pi^l,\pi^u}\left[\sum\nolimits_{t=0}^{T} r^u(s_t, a^l, a^u)\big|s_0 = s\right]$ and $V^u_\rho(\pi^l, \pi^u) = \mathbb{E}_{s\sim\rho}[V^u(s, \pi^l, \pi^u)]$. Then the Markov game is formulated as

$$V^u_\rho(\pi^{l_*}, \pi^{u_*}) = \max_{\pi^u}\min_{\pi^l} V^u_\rho(\pi^l, \pi^u). \tag{4}$$

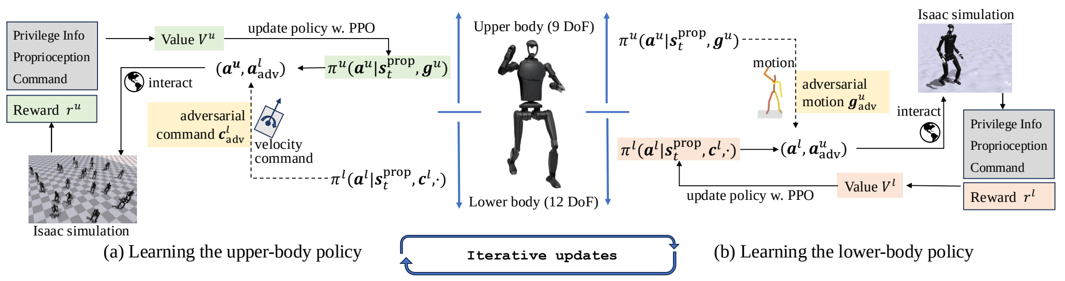

Privilege Info
Proprioception
Command
Value V^u
update policy w. PPO
Reward r^u
interact
$(\boldsymbol{a}^u, \boldsymbol{a}^l_{\text{adv}})$
$\pi^u(\boldsymbol{a}^u|\boldsymbol{s}_t^{\text{prop}}, \boldsymbol{g}^u)$
adversarial command $\boldsymbol{c}^l_{\text{adv}}$
velocity command
$\pi^l(\boldsymbol{a}^l|\boldsymbol{s}_t^{\text{prop}}, \boldsymbol{c}^l, \cdot)$
Isaac simulation
Upper body (9 DoF)
Lower body (12 DoF)
$\pi^u(\boldsymbol{a}^u|\boldsymbol{s}_t^{\text{prop}}, \boldsymbol{g}^u)$
motion
adversarial motion $\boldsymbol{g}^u_{\text{adv}}$
Isaac simulation
interact
$\pi^l(\boldsymbol{a}^l|\boldsymbol{s}_t^{\text{prop}}, \boldsymbol{c}^l, \cdot)$
$(\boldsymbol{a}^l, \boldsymbol{a}^u_{\text{adv}})$
Privilege Info
Proprioception
Command
update policy w. PPO
Value V^l
Reward r^l
(a) Learning the upper-body policy
Iterative updates
(b) Learning the lower-body policy

Figure 1: The overview of ALMI. (a) In updating the upper-body policy π^u, we sample adversarial velocity command $\boldsymbol{c}^l_{\text{adv}}$ and obtains $\boldsymbol{a}^l_{\text{adv}}$. Then we use $(\boldsymbol{a}^u, \boldsymbol{a}^l_{\text{adv}})$ to interact with the environment to collect experiences, which are used to update π^u via PPO algorithm [11]. (b) Similarly, in updating the lower-body policy, we sample adversarial motion $\boldsymbol{g}^u_{\text{adv}}$ and obtains $\boldsymbol{a}^u_{\text{adv}}$. Then we use $(\boldsymbol{a}^l, \boldsymbol{a}^u_{\text{adv}})$ to interact and update π^l. The two policies (π^l, π^u) finally converge via multiple mutual iterations.

By performing independent policy gradient optimization for the two players, we can prove that such a process also results in a ϵ-approximate Nash equilibrium.

Simplified Formulation. According to Eq. (2) and (4), we require two pairs of (π^l, π^u) to optimize V^l and V^u since they are defined by the opposite max-min problem, which could be computationally expensive. To address this, we propose a simplified framework by learning a single paired policy (π^l, π^u). Specifically, (i) in learning the locomotion policy π^l, we optimize the parameters of π^l while keeping the upper-body policy π^u fixed. However, we sample an *adversarial motions* with a designed curriculum from the motion dataset, and set the corresponding reference joint position $\boldsymbol{g}^u_{\text{adv}}$ as the condition of $\pi^u(\cdot|\boldsymbol{s}_t^{\text{prop}}, \boldsymbol{g}^u_{\text{adv}})$ to generate adversarial actions $\boldsymbol{a}^u_{\text{adv}}$. Similarly, (ii) in learning the motion imitation policy π^u, we only update π^u and keep the lower-body policy π^l fixed. Then, we sample an *adversarial command* $\boldsymbol{c}^l$ with the curriculum for the locomotion policy $\pi^l(\cdot|\boldsymbol{s}_t^{\text{prop}}, \boldsymbol{c}^l, \cdot)$ to generate $\boldsymbol{a}^l_{\text{adv}}$. To conclude, we change the inner-loop optimization of the max-min problem (in Eq. (2) and (4)) from the parameter space to the command space (i.e., by sampling $\boldsymbol{g}^u_{\text{adv}}$ and $\boldsymbol{c}^l_{\text{adv}}$), which leads to a practical efficient algorithm. Fig. 1 gives an overview of our method.

3.2 Robust Locomotion for the Lower Body

Achieving robust locomotion in humanoid robots remains a significant challenge, primarily due to the dynamic nature of environments and the inherent complexity of multi-joint floating base systems. This instability is particularly pronounced in full-sized humanoid robots, which often require increased load capacity to execute whole-body control or loco-manipulation tasks. Such requirements typically lead to higher arm mass and inertia, exacerbating system instability. Consequently, during coordinated upper and lower body movements or mobile manipulation, disturbances generated by the upper body exert significant impacts on the lower body, necessitating robust policies to mitigate these effects.

Table 1: Reward terms and weights for lower policy.

Term	Expression	Weight
	Penalty	
DoF position limits	$\mathbb{I}(\boldsymbol{q}^{\text{l}} \notin [\boldsymbol{q}^{\text{l}}_{\text{min}}, \boldsymbol{q}^{\text{l}}_{\text{max}}])$	-5.0
Alive	$\mathbb{I}(\text{robot stays alive})$	0.15
	Regularization	
Linear velocity of Z axis	$\Vert\boldsymbol{v}_{\text{z}}\Vert_2^2$	-2
Angular velocity of X&Y axis	$\Vert\boldsymbol{w}_{\text{xy}}\Vert_2^2$	-0.5
Orientation	$\Vert\boldsymbol{g}^{\text{root}}_{\text{xy}}\Vert_2^2$	-1
Torque	$\Vert\boldsymbol{\tau}^{\text{l}}\Vert_2^2$	-1e-5
Base Height	$\Vert h - h^{\text{target}}\Vert_2^2$	-10.0
DoF acceleration	$\Vert\ddot{\boldsymbol{q}}^{\text{l}}\Vert_2^2$	-2.5e-7
DoF velocity	$\Vert\dot{\boldsymbol{q}}^{\text{l}}\Vert_2^2$	-1e-3
Lower body action rate	$\Vert\boldsymbol{a}^{\text{l}}_t - \boldsymbol{a}^{\text{l}}_{t-1}\Vert_2^2$	-0.01
Hip DoF position	$\Vert\boldsymbol{q}^{\text{hip roll\&yaw}}\Vert_2^2$	-1
Slippage	$\Vert\boldsymbol{v}^{\text{feet}}_{\text{xy}}\Vert_2^2 \times \mathbb{I}(\Vert F^{\text{feet}}\Vert_2 \geq 1)$	-0.2
Feet swing height	$\Vert\boldsymbol{q}^{\text{feet}}_{\text{z}} - 0.08\Vert_2^2 \times \mathbb{I}(\Vert F^{\text{feet}}\Vert_2 < 1)$	-20
Feet Contact	$\sum_{i=1}^{N_{\text{feet}}} \neg(\mathbb{I}(\Vert\boldsymbol{F}^{\text{feet}}_{\text{z,i}}\Vert_2 > 1) \oplus \mathbb{I}(\phi_{\text{i}} < 0.55))$	0.18
Feet distance	$\exp(-100 \times d^{\text{out of range}}_{\text{feet}})$	0.5
Knee distance	$\exp(-100 \times d^{\text{out of range}}_{\text{knee}})$	0.4
Stand still	$\Vert\boldsymbol{q}^{\text{l}}_{\text{t}} - \boldsymbol{q}^{\text{l}}_{\text{t}-1}\Vert_2^2 \times \mathbb{I}(\Vert\boldsymbol{c}^{\text{l}}\Vert_2 < 0.1)$	-2
Ankle torque	$\Vert\boldsymbol{\tau}^{\text{ankle}}\Vert_2^2$	-5e-5
Ankle action rate	$\Vert\boldsymbol{a}^{\text{ankle}}_t - \boldsymbol{a}^{\text{ankle}}_{t-1}\Vert_2^2$	-0.02
Stance base velocity	$\Vert\boldsymbol{v}\Vert_2^2 \times \mathbb{I}(\Vert\boldsymbol{c}^{\text{l}}\Vert_2 < 0.1)$	-1
Feet contact forces	$\min(\Vert F^{\text{feet}} - 100\Vert_2^2, 0)$	-0.01
	Task	
Linear velocity	$\exp(-4\Vert\boldsymbol{c}_{\text{xy}} - \boldsymbol{v}_{\text{xy}}\Vert_2^2)$	2
Angular velocity	$\exp(-4\Vert\boldsymbol{c}_{\text{yaw}} - \boldsymbol{v}_{\text{yaw}}\Vert_2^2)$	1

Motivation of Curriculum. In learning a robust locomotion policy, an adversarial upper body component is essential to generate perturbations. However, directly training such an adversary is challenging. Empirical results show that without constraints, the upper body can terminate episodes early or collide with the lower body to minimize its reward. Manually designing constraints is labor intensive, and the lower body can exploit flaws in the reward function, leading to poor real-world performance. Thus, as we discussed in §3.1, we simplify the original problem by sampling adversarial motions without updating the upper-body policy. Although aggressive sampling of highly dynamic motions can reduce the locomotion policy's value function (i.e. V^l) as in Eq. (1), this strategy cannot adequately teach the locomotion policy how to resist disturbances when the policy is relatively weak, resulting in slow convergence. To overcome this, we introduce a novel curriculum mechanism that starts with moderate disturbances and gradually increases their intensity as the lower body's robustness improves, ensuring efficient learning and robustness improvement throughout training.

Table 2: Reward terms and weights for upper policy.

Term	Expression	Weight
	Penalty	
DoF position limits	$\mathbb{I}(\boldsymbol{q}^{\mathrm{u}} \notin [\boldsymbol{q}^{\mathrm{u}}_{\min}, \boldsymbol{q}^{\mathrm{u}}_{\max}])$	-5.0
Alive	$\mathbb{I}(\mathrm{robot\ stays\ alive})$	0.15
	Regularization	
Orientation	$\lVert \boldsymbol{g}^{\mathrm{root}}_{\mathrm{xy}} \rVert_2^2$	-1
Torque	$\lVert \boldsymbol{\tau}^{\mathrm{u}} \rVert_2^2$	-1e-5
Upper DoF acceleration	$\lVert \ddot{\boldsymbol{q}}^{\mathrm{u}} \rVert_2^2$	-2.5e-7
Upper DoF velocity	$\lVert \dot{\boldsymbol{q}}^{\mathrm{u}} \rVert_2^2$	-1e-3
Upper body action rate	$\lVert \boldsymbol{a}^{\mathrm{u}}_t - \boldsymbol{a}^{\mathrm{u}}_{t-1} \rVert_2^2$	-0.01
	Task	
Upper DoF position	$\exp(-0.5\lVert \hat{\boldsymbol{q}^{\mathrm{u}}} - \boldsymbol{q}^{\mathrm{u}} \rVert_2^2)$	10

Dual Curriculum Mechanism. We propose a novel dual curriculum mechanism for adversarial motion sampling by scoring upper-body motions based on their impact on the lower-body's stability. Specifically, we initially train a basic locomotion policy π^l_0 without any upper body intervention. Then we use a PD controller (in the first round) or π^u to control the upper-body to track motions from the re-targeted AMASS [8] dataset in simulation, while the lower body follows the basic velocity commands. The robot inevitably falls initially due to upper body perturbations, and we record *survival length* (denoted as $l^{\mathrm{sl}} \in [0, l^{\mathrm{sl}}_{\max}]$) as the primary metric for motion difficulty. Then, motions are sorted into a list $\mathbf{M} = [M_1, \ldots, M_{|\mathbf{M}|}]$ by increasing difficulty based on survival length. Further, we introduce a factor, *motion scale* $\alpha_s \in [0, 1]$, to scale the target joint positions of the upper body as

$$\boldsymbol{q}^i_{\text{target}} = \boldsymbol{q}^i_0 + \alpha_s(\boldsymbol{q}^i_{\text{ref}} - \boldsymbol{q}^i_0). \tag{5}$$

For each iteration, the robot is assigned a window of motions $[M_{\alpha_d}, M_{\alpha_d+w}]$ from $\mathbf{M}$, where w is the window size and $\alpha_d = 1$ at the start. These motions, scaled by α_s, are used to control the upper body to provide disturbances. The lower-body policy π^l_0 is updated to $\pi^l_1 \leftarrow \pi^l_0$ through an RL optimization process with locomotion-related rewards detailed in Table 1. To adaptively adjust the difficulty of sampled motions, we calculate *mean survival length* (denoted as l^{msl}) of the sampled motions as a metric to update α_d and α_s, which progressively increases motion difficulty based on the current policy's anti-disturbance capability. Formally,

$$\alpha_d \leftarrow \begin{cases} \min(\alpha_d + w, |\mathbf{M}| - w), & \text{if} \quad l^{\mathrm{msl}} \geq 0.8 * l^{\mathrm{sl}}_{\max} \\ \max(\alpha_d - 2w, 0), & \text{otherwise} \end{cases}, \tag{6}$$

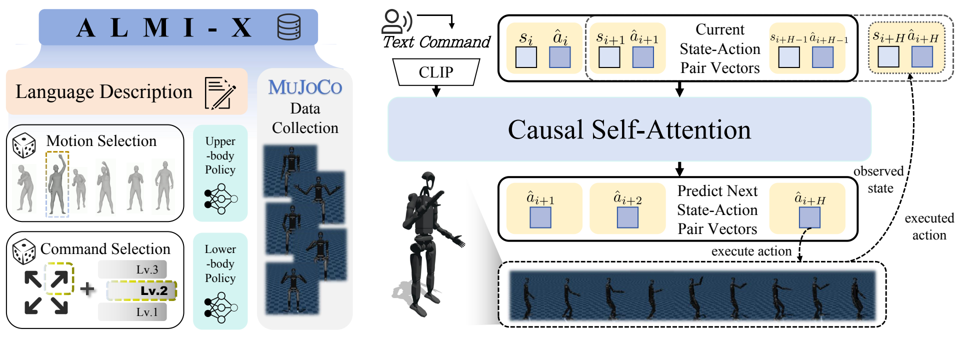
which increases the difficulty if π^l can effectively resist the interference caused by sampled motions, and decreases the difficulty otherwise. The update of motion scale follows a similar process as

$$\alpha_s \leftarrow \begin{cases} \min(\alpha_s + 0.05, 1), & \alpha_d = |\mathbf{M}| - w \\ \max(\alpha_s - 0.01, 0), & \alpha_d = 0 \end{cases}, \tag{7}$$

which signifies that the motion scale α_s increases after all motions at the current scale are successfully managed. However, increasing α_s typically decreases l^{msl}, potentially causing α_d to decrease and resulting in motion sampling that reverts to previous entries in $\mathbf{M}$. This triggers a new cycle of policy updates with adjusted motions and scales. To maintain relevance, the motion list $\mathbf{M}$ is periodically re-sorted to ensure alignment with the evolving robustness of the locomotion policy. The Algorithm 1 in Appendix B.2 gives a training process for the locomotion policy.

3.3 Motion Tracking for the Upper Body

The upper-body policy π^u aims to track various motions under disturbances from lower-body movements with adversarial commands. Before motion tracking, we follow the process proposed by



(a) ALMI-X Collection

(b) Transformer-based Foundation Policy

Figure 2: ALMI-X dataset and foundation model training. (a) ALMI-X features motion target and velocity command for the learned policies, combining with language description. (b) The foundation model learns $P(\hat{a}_{i+1}|s_{\leq i}, a_{\leq i}, \mathcal{T})$ from a segment of state-action pairs via causal attention. In inference, the last action is executed based on the history and obtains the true state for next steps.

PHC [12] to re-target the human motion from AMASS [8] dataset to humanoid robot. To enhance motion smoothness, we applied low-pass filtering to the re-targeted motion data.

In the adversarial training process, we use the locomotion policy π^l in the current round to control the lower body, and the upper-body policy is learned by maximizing the motion tracking reward. Different from §3.2 that requires delicate curriculum mechanism, the curriculum for adversarial π^l can be relatively simple since the command $\mathbf{c}^l$ only contains velocity commands. Specifically, $\mathbf{c}^l$ is sampled from a range of $[\boldsymbol{c}^l_{\text{min}}, \boldsymbol{c}^l_{\text{max}}]$, and we adjust the value of $\boldsymbol{c}^l_{\text{min}}$ and $\boldsymbol{c}^l_{\text{max}}$ according to the tracking error of motions in the upper body, which is affected by the movement of the lower body. In training π^u, the lower-body command gradually increases its difficulty based on the tracking error of the upper body. The reward design for the upper-body policy is described in Table 2, and the detailed curriculum mechanism is provided in Appendix B.3.

4 ALMI-X Datasets and Foundation Model

The adversarial training processes outlined in §3.2 and §3.3 involve an iterative procedure over multiple epochs to reach convergence. The final policies are used to gather a large-scale high-quality dataset, which is used to train a foundation model dedicated to end-to-end whole-body control.

Dataset Construction. We collect ALMI-X dataset in MuJoCo simulations by running the trained ALMI policy, which consists of the final policy π^l and π^u. In this simulation, we combine a diverse range of upper-body motions with omnidirectional lower-body commands, and employ a pre-defined paradigm to generate corresponding linguistic descriptions for each combination. (i) For the upper-body, we collect data using our upper-body policy to track various motions from a subset of the AMASS dataset [8], where we remove entries with indistinct movements or those that could not be matched with the lower-body commands, such as *push from behind*. (ii) For the lower-body, we first categorize command directions into several types according to different combination of linear and angular velocity command and define 3 difficulty levels for command magnitudes, then the lower-body command is set by combining direction types and difficulty levels. Overall, each upper-body motion from the AMASS subset is paired with a specific direction type and a difficulty level serving as the inputs of π^u and π^l to control the robot. In addition, trajectories in which the lower body *keep standing* while the upper body tracks motions are also incorporated into the dataset. Each language description $\mathcal{T}$ in ALMI-X is organized as *"${movement mode} ${direction} ${velocity level} and ${motion}"*, each of which corresponds to the data collected from a trajectory lasting on average 4 seconds with 200 steps. For each $\mathcal{T}$, we run two policies (i.e., π^u, π^l) based on the commands obtained from the aforementioned combinations to achieve humanoid whole-body control. We record the trajectory information such as the robot states $\tau_s = (s_0, s_1, ..., s_T)$ and actions $\tau_a = (a_0, a_1, ..., a_T)$. Our data exhibits high executability, and the details are discussed in Appendix C.

Foundation Model for Humanoid Control. Using the ALMI-X dataset, we give preliminary attempts to train a whole-body foundation model with supervised learning that can execute various motions in response to text commands. We adopt a Transformer decoder architecture that implements causal self-attention modules over a text command and historical state-action pairs. Unlike UH-1 [13] that predicts the entire action sequence based on the text command, our policy autoregressively models the next action by incorporating real-time interaction history between the robot and the environment. This architecture avoids executing the entire action sequence without considering the random noise and interference of the environment, instead adjusting actions based on intermediate states, ensuring robust and adaptive control. Moreover, compared to methods focused solely on walking [14], ALMI-X contains more diverse data from various motions and velocity commands, enabling the learning of the foundation model to achieve complex language-guided whole body control. The architecture is shown in Fig. 2 and the details are given in Appendix D.

5 Related Works

Humanoid Locomotion. Traditional humanoid locomotion primarily focuses on trajectory planning [15, 16] and Zero-Moment Point (ZMP)-based gait synthesis [17, 18]. Other methods like whole-body control [19, 20] and model-predictive control [21, 22] gain significant progress with dynamics modeling, while it can be difficult to model rich contact in complex terrains [23]. Recently, learning-based algorithms have shown promising results in humanoid locomotion tasks [24, 25]. RL has become a powerful tool for learning locomotion policies by enabling agents to interact with the environment through parallelized simulations [26]. Based on this, humanoids can acquire a wide range of skills, including walking on complex terrains [27], [28], gait control [29], standing up [30, 31], narrow-terrain navigation [32], jumping [33], and even parkour [34, 35]. However, most methods focus solely on controlling the lower-body joints, treating the upper body as a fixed load [36, 37]. Although some studies have explored whole-body control policies [32, 29], their primary objective is to learn an upper-body swinging policy to coordinate with the locomotion task. In contrast, our method enables the lower-body policy to resist disturbances caused by various upper-body motions.

Humanoid Motion Imitation. Recent research has proposed expressive motion imitation algorithms that learn human-like behaviors from motion capture datasets [38–40]. These methods employ motion re-targeting and RL-based optimization to achieve fine-grained motion imitation and postural stability [41–43]. However, balancing these two objectives can be challenging, as the execution of motions may disturb stability, while poor stability, in turn, affects the precision of motion execution. Several methods address this issue through careful reward design [44, 45], decomposed tracking strategies [6, 4, 46, 47], and residual learning [48]. Unlike these methods, our approach focuses solely on tracking human motions in the upper body. Although this may result in less expressiveness, it leads to robust locomotion in the lower body and precise motion tracking in the upper body, leading to an effective whole-body policy for real-world deployment. Other methods [49, 50, 5] also adopt a separate control paradigm for training the two parts with different rewards, while we employ an adversarial training process to promote coordinated behaviors between locomotion and motion imitation policies.

Foundation Model for Robotics. Our work is also related to the learning of foundation models for robotics. Previous works mainly focus on building vision-language-action models for end-to-end manipulation, such as MT-Diff [51], RT-1 [52], Octo [53], OpenVLA [54], RDT [55], π_0 [56], GO-1 [57], and BRS [58]. They rely on large-scale datasets collected by humans, such as RH20T [59], Bridge Data [60], RoboMIND [61], Open-X [62], AgiBot Data [57] and HGen-Bench[63]. However, these datasets focus mainly on arm manipulation with a fixed or wheeled platform. In contrast, our dataset (i.e., ALMI-X) targets whole-body control in humanoid robots, which involves controlling more DoFs than manipulation tasks. UH-1 [13] is closely related to ALMI, but learns a mapping from text descriptions to keypoints, and low-level actions are found to be unexecutable in real robots.

6 Experiments

In this section, we evaluate the performance of ALMI using the Unitree H1-2 robot in both simulated and real-world environments. Our experiments aim to address the following research questions: **Q1.** How does ALMI perform in terms of **tracking precision** for lower-body velocity commands and upper-body motions? **Q2.** How does ALMI perform regarding **stability** and **robustness** in complex scenarios? **Q3.** What benefits does **adversarial iterative training** and the **arm curriculum** mechanism provide for policy optimization? **Q4.** How does ALMI perform in the **real world**?

Training Details. Policy training is conducted within the Isaac Gym simulator utilizing 4,096 parallel environments. We perform three rounds of adversarial iterations and evaluate the policies of the final round. Notably, during the adversarial iterative training process, the initial lower-body policy converges after approximately 10^4 steps. As iterations progress, the number of steps required for convergence decreases significantly. The total duration of the three training iterations is approximately 17 hours. For each baseline method, we use the checkpoint obtained after training convergence for testing. To enhance generalization, we implement domain randomization for sim-to-sim and sim-to-real transfers, with specific details provided in Appendix B.5.

Evaluation Metrics. For simulated evaluation, we adopt the high-quality CMU MoCap dataset [64] with 1122 motion clips (denoted as $\mathcal{D}_{\text{cmu}}$) to evaluate different metrics in IsaacGym [26]. The metrics include (i) the *mean linear velocity error* $E_{\text{vel}}(\text{m/s})$ and the mean angular velocity error $E_{\text{ang}}(\text{rad/s})$ that evaluate the command tracking accuracy; (ii) the upper body *joints position error* $E_{\text{jpe}}^{\text{upper}}(\text{m})$ and *key points error* $E_{\text{kpe}}^{\text{upper}}(\text{m})$ that evaluates the motion tracking accuracy; (iii) the *joint difference* for both parts $E_{\text{action}}^{\text{upper}}(\text{rad})$, $E_{\text{action}}^{\text{lower}}(\text{rad})$; (iv) and the *projected gravity* E_{g} evaluate the stability of policy. (v) We also statistically analyze *survival rate* to assess the robustness of the policy.

Baselines. Our baselines are as follows: (i) **Exbody** [6], which employs a unified policy to control whole-body joints, tracking upper-body movements from motion data and lower-body root motion via command sampling. (ii) **ALMI (whole)**, which uses a single policy to control the upper and lower bodies, with identical reward functions for training each part as in ALMI. (iii) **ALMI (w/o curriculum)**. An ablation study that omits the arm curriculum, where the policy loads the motion randomly and the motion scale α_s is set to 1. (iv) **ALMI (w/o adv. learning)**. This study evaluates the impact of our iterative adversarial training by testing policies from the first and second rounds. We also compare our method with approaches that simultaneously imitate both upper and lower body movements. We consider two state-of-the-art methods: (v) **ExBody2** [4], which leverages a privileged teacher policy to distill precise mimicry skills into a student policy, enabling whole-body imitation, and (vi) **OmniH2O** [45], an imitation-based whole-body control method that supports real-time input from various input devices.

6.1 Main Result of ALMI

To evaluate locomotion and motion tracking capabilities, we calculate metrics in $\mathcal{D}_{\text{cmu}}$ using commands in the lower body categorized into difficulty levels *easy*, *medium*, and *hard*, as described in Table 3. These levels encompass linear velocities in the x and y directions, angular velocity in the yaw direction, terrain level in the Isaac Gym and the presence of external forces. To control variables and facilitate analysis, we assess linear and angular velocities separately (setting the other to zero during testing) and report the average metrics.

To answer **Q1** and **Q2**, we give the comparative results in Table 4. (i) **Tracking accuracy.** ALMI demonstrates superior tracking precision across all difficulty levels for both

Table 3: Locomotion difficulty level setting.

	Command & Environment				
Level	$\hat{v}_{\text{x,t}}$	$\hat{v}_{\text{y,t}}$	$\hat{\omega}_{\text{yaw,t}}$	terrain level	push robot
easy	0.7	0.0	0.2	0	✗
medium	1.0	0.3	0.4	3	✓
hard	1.3	0.6	0.6	6	✓

Table 4: Simulated evaluation of ALMI, ALMI (whole body) and Exbody on CMU dataset.

	Metrics							
Method	$E_{\text{vel}}\downarrow$	$E_{\text{ang}}\downarrow$	$E_{\text{jpe}}^{\text{upper}}\downarrow$	$E_{\text{kpe}}^{\text{upper}}\downarrow$	$E_{\text{action}}^{\text{upper}}\downarrow$	$E_{\text{action}}^{\text{lower}}\downarrow$	$E_{\text{g}}\downarrow$	Survival $\uparrow$
Easy								
ALMI	**0.1135**	**0.2647**	**0.1931**	**0.0460**	**0.0462**	**0.0170**	**0.6919**	**1.0000**
ALMI(whole)	0.1386	0.5433	0.5756	0.0704	0.0800	3.0356	0.9675	0.9991
Exbody	0.2383	0.4056	0.3559	0.0995	1.7813	1.8152	0.9693	0.8912
Medium								
ALMI	**0.2192**	**0.3520**	**0.2007**	**0.0450**	**0.0598**	**0.0172**	**0.7604**	**0.9852**
ALMI(whole)	0.2380	0.5563	0.6734	0.0637	0.0409	2.9225	1.0750	0.9763
Exbody	0.3063	0.5087	0.3658	0.1233	1.7683	1.8019	1.0166	0.8845
Hard								
ALMI	**0.2202**	**0.4812**	**0.2116**	**0.0458**	**0.0600**	**0.0175**	**0.8551**	**0.9723**
ALMI(whole)	0.3178	0.7224	0.7022	0.0635	0.0519	2.9317	1.1656	0.9491
Exbody	0.4838	0.5753	0.3758	0.1269	1.7352	1.7689	1.0243	0.8778

Table 5: Comparison with Exbody2 and OmniH2O on G1 in simulation.

Method	Metrics							
	$E_{\text{vel}} \downarrow$	$E_{\text{ang}} \downarrow$	$E_{\text{jpe}}^{\text{upper}} \downarrow$	$E_{\text{kpe}}^{\text{upper}} \downarrow$	$E_{\text{action}}^{\text{upper}} \downarrow$	$E_{\text{action}}^{\text{lower}} \downarrow$	$E_{\text{g}} \downarrow$	Survival $\uparrow$
ALMI	**0.1396**	**0.2776**	**0.2367**	**0.0411**	**0.0198**	**0.7411**	**0.0977**	**0.9484**
OmniH2O	0.1615	0.4166	1.0826	0.0598	1.2773	2.3219	0.1696	0.3882
Exbody2	0.4015	0.6066	0.3821	0.0719	1.1797	1.3547	0.3367	0.8848

upper-body motions and lower-body velocity commands, consistently outperforming other baselines. The results highlight that a single policy struggles with upper-lower body coordination, limiting tracking accuracy. In contrast, our adversarial training framework simultaneously enhances the accuracy of both policies for their respective objectives. (ii) **Stability and robustness.** According to the survival rate, it can be found that with an improvement in difficulty levels, ALMI can track motions and velocity commands robustly and effectively prevent falls. This stability is attributed to the lower-body policy, which effectively learns to execute stable movements despite adversarial upper-body interference, underscoring the efficacy of adversarial training strategy.

As OmniH2O and Exbody2 are imitation-based whole-body control methods without velocity tracking, for ALMI, we use linear and angular velocity provided by the reference motion as velocity tracking command; for OmniH2O and Exbody2, they directly track the reference motions. This experiment is conducted without terrain or push. To indicate that our method can be extended to other robot platforms, we use Unitree G1 robot in this setting. The results are shown in 5. It can be observed that our method generalizes well across different robotic platforms and demonstrates superior robustness compared to imitation-based whole-body control methods.

To answer **Q3**, we evaluate the impact of our arm curriculum and adversarial learning techniques, with results presented in Table 6. We use the trained lower-body policies across three iterations, labeled *lower-1*, *lower-2*, and *lower-3*. Under easy velocity commands without environmental disturbances, performance differences among the iterations are minimal. However, as task complexity increased, *lower-3* demonstrated superior performance across all metrics compared to earlier iterations, while *lower-1* showed notably inferior results. During adversarial training, the lower and upper policies iteratively refined their motion and command tracking capabilities while generating progressively larger disturbances for each other, guided by the designed curriculum. For instance, to achieve high-velocity commands, the lower body might execute large-amplitude actions, inducing oscillations that disrupt upper-body movements, and vice versa. Through this iterative process, both policies gradually adapt to adversarial perturbations, eventually reaching a stable condition. In addition, Table 6 underscores the critical role of the arm curriculum. By systematically introducing upper-body motions from easy to hard in training, the lower-body policy effectively learned to handle increasingly difficult perturbations and motion scales.

Table 6: Ablation studies of adversarial training technique and arm curriculum in ALMI.

Method	Metrics							
	$E_{\text{vel}} \downarrow$	$E_{\text{ang}} \downarrow$	$E_{\text{jpe}}^{\text{upper}} \downarrow$	$E_{\text{kpe}}^{upper} \downarrow$	$E_{\text{action}}^{\text{upper}} \downarrow$	$E_{\text{action}}^{\text{lower}} \downarrow$	$E_{\text{g}} \downarrow$	Survival $\uparrow$
Easy								
lower-3 + upper-2	**0.1135**	**0.2647**	0.1931	**0.0460**	**0.0462**	**0.0170**	**0.6919**	**1.0000**
lower-2 + upper-2	0.1164	0.2669	0.1955	0.0452	0.0475	**0.0171**	0.7121	**1.0000**
lower-1 + upper-2	0.1271	0.2738	0.1928	0.0526	0.0642	**0.0171**	0.7052	**1.0000**
w/o arm curriculum	0.1411	0.2726	**0.1924**	0.0504	0.0618	**0.0172**	0.7472	0.9995
Medium								
lower-3 + upper-2	**0.2192**	**0.3520**	**0.2007**	**0.0450**	**0.0598**	**0.0172**	**0.7604**	**0.9852**
lower-2 + upper-2	0.2213	0.3571	0.2032	0.0458	0.0607	**0.0172**	0.7748	0.9772
lower-1 + upper-2	0.2262	0.3872	0.2173	0.0492	0.0604	0.0175	0.7730	0.9273
w/o arm curriculum	0.2571	0.4348	0.2068	0.0476	0.0601	0.0173	1.0587	0.9652
Hard								
lower-3 + upper-2	**0.2202**	**0.4812**	**0.2116**	**0.0458**	**0.0600**	**0.0175**	**0.8551**	**0.9723**
lower-2 + upper-2	0.2892	0.5395	0.2231	0.0482	0.0645	0.0178	0.9479	0.9233
lower-1 + upper-2	0.2566	0.5172	0.2451	0.0537	0.0777	0.0179	0.9462	0.8743
w/o arm curriculum	0.3658	0.6398	0.2394	0.0461	0.0726	0.0180	1.2042	0.8480

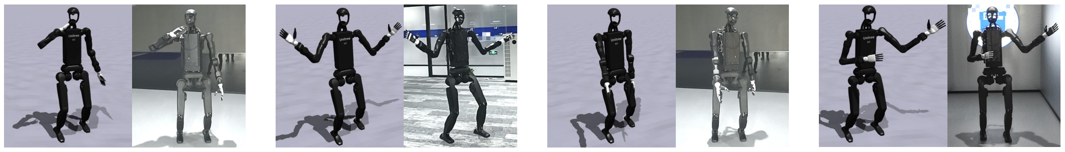

(a) Salute (b) Wave (c) Shake hand (d) Punch

Figure 3: The sim-to-real comparison of humanoid robot in tracking various motions.

6.2 Result of Foundation Model

The Transformer-based foundation model leverages textual descriptions of the whole-body motion along with historical information to predict future action. In experiments, we find that when training on a subset of the ALMI-X dataset (e.g., waving-related data), the model converges to a robust policy, enabling execution of various waving and moving motions in the MuJoCo simulations. We highlight that this is particularly challenging, as no prior works show a purely offline policy can control full-sized humanoid robots to generate language-guided behaviors. However, when we extend the training data to the complete ALMI-X, the performance degraded, indicating limitations of the current model in handling highly diverse behaviors. ALMI-X dataset and our finding provide a promising starting point for future exploration, with detailed results provided in Appendix D.

6.3 Real-World Experiments

To answer **Q4**, we deploy ALMI on the Unitree H1-2 robot to evaluate real-world performance. For lower-body control, we use a joystick-mounted remote controller to send velocity commands to π^l, which controls the robot to achieve omnidirectional movement. For upper body control, we can adopt open-loop controller or ALMI policy to track various motions, as well as using VR devices to support dexterous control in loco-manipulation tasks. The details are given in Appendix F.

To evaluate the capabilities in maintaining stability in movements while tracking motions precisely, we control the robot to execute various upper body motions while standing or walking in omnidirection. Fig. 3 illustrates the motion tracking behavior, emphasizing the alignment between simulation and real-world performance. To further demonstrate the robot's capability to perform complex motions during locomotion, Fig. 8 in the appendix provides a sim-to-real comparison of the robot completing a full motion sequence while moving forward and recovering to a standing position. These experiments confirm that ALMI enables the real robot to achieve robust locomotion and accurate motion tracking.

7 Conclusion

This paper presented ALMI, an adversarial training framework for humanoid whole-body control. ALMI leverages the designed curriculum for upper and lower body training, achieving stable locomotion and precise motion imitation through iterative policy updates. Our theoretical analysis demonstrates ALMI's effectiveness within a two-player Markov game framework, supported by practical algorithms for implementation. The learned policy is used to collect the ALMI-X dataset with language annotations, facilitating foundation model training for the research community. Empirical results highlight ALMI's superior performance in both simulated environments and real-world deployments. However, our approach still has limitations. The adversarial framework of ALMI, albeit practical and robust for mobile manipulation, performs suboptimally in highly dynamic whole-body tasks like dancing; furthermore, our foundation model remains exploratory, with significant headroom for enhancing data efficiency. Future works include enhancing whole-body coordination through unified task-oriented rewards and improving the foundation model using advanced model architectures.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (Grant No.2024YFE0210900), the National Natural Science Foundation of China (Grant No.62306242), the Young Elite Scientists Sponsorship Program by CAST (Grant No. 2024QNRC001), and the Yangfan Project of the Shanghai (Grant No.23YF11462200).

References

[1] Zan Wang, Yixin Chen, Baoxiong Jia, Puhao Li, Jinlu Zhang, Jingze Zhang, Tengyu Liu, Yixin Zhu, Wei Liang, and Siyuan Huang. Move as you say interact as you can: Language-guided human motion generation with scene affordance. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 433–444, 2024.

[2] Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. Motiongpt: Human motion as a foreign language. *Advances in Neural Information Processing Systems*, 36:20067–20079, 2023.

[3] Zipeng Fu, Qingqing Zhao, Qi Wu, Gordon Wetzstein, and Chelsea Finn. Humanplus: Humanoid shadowing and imitation from humans. In *8th Annual Conference on Robot Learning*, 2024.

[4] 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.

[5] 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. In *IEEE International Conference on Robotics and Automation*, 2025.

[6] Xuxin Cheng, Yandong Ji, Junming Chen, Ruihan Yang, Ge Yang, and Xiaolong Wang. Expressive whole-body control for humanoid robots. In *Robotics: Science and Systems*, 2024.

[7] Qingwei Ben, Feiyu Jia, Jia Zeng, Junting Dong, Dahua Lin, and Jiangmiao Pang. HOMIE: Humanoid Loco-Manipulation with Isomorphic Exoskeleton Cockpit. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.070.

[8] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019.

[9] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[10] Julien Perolat, Bruno Scherrer, Bilal Piot, and Olivier Pietquin. Approximate dynamic programming for two-player zero-sum markov games. In *International Conference on Machine Learning*, pages 1321–1329. PMLR, 2015.

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

[12] 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.

[13] Jiageng Mao, Siheng Zhao, Siqi Song, Tianheng Shi, Junjie Ye, Mingtong Zhang, Haoran Geng, Jitendra Malik, Vitor Guizilini, and Yue Wang. Learning from massive human videos for universal humanoid pose control. *arXiv preprint arXiv:2412.14172*, 2024.

[14] Ilija Radosavovic, Bike Zhang, Baifeng Shi, Jathushan Rajasegaran, Sarthak Kamat, Trevor Darrell, Koushil Sreenath, and Jitendra Malik. Humanoid locomotion as next token prediction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[15] Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. Perceptive locomotion through nonlinear model-predictive control. *IEEE Transactions on Robotics*, 39(5):3402–3421, 2023. doi: 10.1109/TRO.2023.3275384.

[16] Avadesh Meduri, Paarth Shah, Julian Viereck, Majid Khadiv, Ioannis Havoutis, and Ludovic Righetti. Biconmp: A nonlinear model predictive control framework for whole body motion planning. *IEEE Transactions on Robotics*, 39(2):905–922, 2023. doi: 10.1109/TRO.2023.3268888.

[17] M. Vukobratovic and J. Stepanenko. On the stability of anthropomorphic systems. *Mathematical Biosciences*, 15:1–37, 1972.

[18] M. Vukobratović, B. Borovac, D. Surla, and D. Stokić. *Biped Locomotion: Dynamics, Stability, Control and Application*. Springer-Verlag, Berlin, 1990.

[19] Luis Sentis and Oussama Khatib. A whole-body control framework for humanoids operating in human environments. In *IEEE International Conference on Robotics and Automation*, pages 2641–2648, Orlando, FL, USA, 2006. IEEE.

[20] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard. Whole-body model-predictive control applied to the HRP-2 humanoid. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3346–3351, 2015.

[21] Junheng Li and Quan Nguyen. Multi-contact mpc for dynamic loco-manipulation on humanoid robots. In *American Control Conference (ACC)*, pages 1215–1220. IEEE, 2023.

[22] Jean-Pierre Sleiman, Farbod Farshidian, Maria Vittoria Minniti, and Marco Hutter. A unified mpc framework for whole-body dynamic locomotion and manipulation. *IEEE Robotics and Automation Letters*, 6(3):4688–4695, 2021. doi: 10.1109/LRA.2021.3068908.

[23] Gerrit Schultz and Katja Mombaur. Modeling and optimal control of human-like running. *IEEE/ASME Transactions on Mechatronics*, 15(5):783–792, 2009. doi: 10.1109/TMECH.2009.2032865.

[24] 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.

[25] 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.

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

[27] Huayi Wang, Zirui Wang, Junli Ren, Qingwei Ben, Tao Huang, Weinan Zhang, and Jiangmiao Pang. BeamDojo: Learning Agile Humanoid Locomotion on Sparse Footholds. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.068.

[28] Dewei Wang, Xinmiao Wang, Xinzhe Liu, Jiyuan Shi, Yingnan Zhao, Chenjia Bai, and Xuelong Li. More: Mixture of residual experts for humanoid lifelike gaits learning on complex terrains, 2025. URL `https://arxiv.org/abs/2506.08840`.

[29] Yufei Xue, Wentao Dong, Minghuan Liu, Weinan Zhang, and Jiangmiao Pang. A Unified and General Humanoid Whole-Body Controller for Fine-Grained Locomotion. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.067.

[30] Xialin He, Runpei Dong, Zixuan Chen, and Saurabh Gupta. Learning Getting-Up Policies for Real-World Humanoid Robots. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.063.

[31] Tao Huang, Junli Ren, Huayi Wang, Zirui Wang, Qingwei Ben, Muning Wen, Xiao Chen, Jianan Li, and Jiangmiao Pang. Learning Humanoid Standing-up Control across Diverse Postures. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.064.

[32] Weiji Xie, Chenjia Bai, Jiyuan Shi, Junkai Yang, Yunfei Ge, Weinan Zhang, and Xuelong Li. Humanoid whole-body locomotion on narrow terrain via dynamic balance and reinforcement learning. *arXiv preprint arXiv:2502.17219*, 2025.

[33] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Robust and versatile bipedal jumping control through reinforcement learning. In *Robotics science and systems*. RSS, 2023.

[34] Junfeng Long, Junli Ren, Moji Shi, Zirui Wang, Tao Huang, Ping Luo, and Jiangmiao Pang. Learning humanoid locomotion with perceptive internal model. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9997–10003. IEEE, 2025.

[35] Ziwen Zhuang, Shenzhe Yao, and Hang Zhao. Humanoid parkour learning. In *8th Annual Conference on Robot Learning*.

[36] Xinyang Gu, Yen-Jen Wang, and Jianyu Chen. Humanoid-gym: Reinforcement learning for humanoid robot with zero-shot sim2real transfer. *arXiv preprint arXiv:2404.05695*, 2024.

[37] 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. In *Robotics: Science and Systems*, 2024.

[38] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (ToG)*, 40(4), 2021.

[39] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Trans. Graph.*, 41(4), July 2022.

[40] Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. Calm: Conditional adversarial latent models for directable virtual characters. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701597. doi: 10.1145/3588432.3591541.

[41] Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris M. Kitani, and Weipeng Xu. Universal humanoid motion representations for physics-based control. In *The Twelfth International Conference on Learning Representations*, 2024.

[42] 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)*, 2024.

[43] 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*, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701597.

[44] Tairan He, Zhengyi Luo, Wenli Xiao, Chong Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Learning human-to-humanoid real-time whole-body teleoperation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024.

[45] Tairan He, Zhengyi Luo, Xialin He, Wenli Xiao, Chong Zhang, Weinan Zhang, Kris M Kitani, Changliu Liu, and Guanya Shi. Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning. In *8th Annual Conference on Robot Learning*, 2024.

[46] Weiji Xie, Jinrui Han, Jiakun Zheng, Huanyu Li, Xinzhe Liu, Jiyuan Shi, Weinan Zhang, Chenjia Bai, and Xuelong Li. Kungfubot: Physics-based humanoid whole-body control for learning highly-dynamic skills. In *Neural Information Processing Systems (NeurIPS)*, 2025. URL `https://arxiv.org/abs/2506.12851`.

[47] Weiji Xie, Jinrui Han, Jiyuan Shi, Jiakun Zheng, Huanyu Li, Xinzhe Liu, Weinan Zhang, Chenjia Bai, and Xuelong Li. KungfuBot2: Learning versatile motion skills for humanoid whole-body control. Sep 2025. Under review.

[48] Tairan He, Jiawei Gao, Wenli Xiao, Yuanhang Zhang, Zi Wang, Jiashun Wang, Zhengyi Luo, Guanqi He, Nikhil Sobanbabu, Chaoyi Pan, Zeji Yi, Guannan Qu, Kris Kitani, Jessica K. Hodgins, Linxi Fan, Yuke Zhu, Changliu Liu, and Guanya Shi. ASAP: Aligning Simulation and Real-World Physics for Learning Agile Humanoid Whole-Body Skills. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.066.

[49] Minghuan Liu, Zixuan Chen, Xuxin Cheng, Yandong Ji, Rizhao Qiu, Ruihan Yang, and Xiaolong Wang. Visual whole-body control for legged loco-manipulation. *CoRL*, 2024.

[50] Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: Teleoperation with immersive active visual feedback. In *8th Annual Conference on Robot Learning*, 2024.

[51] Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *Advances in Neural Information Processing Systems*, 36:64896–64917, 2023.

[52] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *Robotics: Science and Systems XIX*, 2023.

[53] Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024.

[54] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*.

[55] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *The Thirteenth International Conference on Learning Representations*.

[56] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. : A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.010.

[57] AgiBot-World-Contributors. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems, 2025.

[58] Yunfan Jiang, Ruohan Zhang, Josiah Wong, Chen Wang, Yanjie Ze, Hang Yin, Cem Gokmen, Shuran Song, Jiajun Wu, and Li Fei-Fei. Behavior robot suite: Streamlining real-world whole-body manipulation for everyday household activities. *arXiv preprint arXiv:2503.05652*, 2025.

[59] Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 653–660. IEEE, 2024.

[60] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[61] Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, Shichao Fan, Xinhua Wang, Fei Liao, Zhen Zhao, Guangyu Li, Zhao Jin, Lecheng Wang, Jilei Mao, Ning Liu, Pei Ren, Qiang Zhang, Yaoxu Lyu, Mengzhen Liu, He Jingyang, Yulin Luo, Zeyu Gao, Chenxuan Li, Chenyang Gu, Yankai Fu, Di Wu, Xingyu Wang, Sixiang Chen, Zhenyu Wang, Pengju An, Siyuan Qian, Shanghang Zhang, and Jian Tang. RoboMIND: Benchmark on Multi-embodiment Intelligence Normative Data for Robot Manipulation. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.152.

[62] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[63] Zhi Jing, Siyuan Yang, Jicong Ao, Ting Xiao, Yugang Jiang, and Chenjia Bai. Humanoidgen: Data generation for bimanual dexterous manipulation via llm reasoning. In *Neural Information Processing Systems (NeurIPS)*, 2025. URL `https://arxiv.org/abs/2507.00833`.

[64] Carnegie Mellon University. CMU MoCap Dataset, 2024. URL `http://mocap.cs.cmu.edu`.

[65] Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

[66] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

[67] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. In *International Conference on Learning Representations*, 2018.

[68] Panayotis Mertikopoulos, Christos Papadimitriou, and Georgios Piliouras. Cycles in adversarial regularized learning. In *Proceedings of the twenty-ninth annual ACM-SIAM symposium on discrete algorithms*, pages 2703–2717. SIAM, 2018.

[69] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems*, 30, 2017.

[70] Tianyi Lin, Chi Jin, and Michael Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In *International Conference on Machine Learning*, pages 6083–6093. PMLR, 2020.

[71] Constantinos Daskalakis, Dylan J Foster, and Noah Golowich. Independent policy gradient methods for competitive reinforcement learning. *Advances in Neural Information Processing Systems*, 33:5527–5540, 2020.

[72] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

[73] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J. Pal, and Liam Paull. Active domain randomization. In *Proceedings of the Conference on Robot Learning*, volume 100, pages 1162–1176. PMLR, 30 Oct–01 Nov 2020.

[74] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763, 2021.

[75] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[76] Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Yong Zhang, Hongwei Zhao, Hongtao Lu, Xi Shen, and Ying Shan. Generating human motion from textual descriptions with discrete representations. In *IEEE/CVF conference on computer vision and pattern recognition*, pages 14730–14740, 2023.

[77] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. *Communications of the ACM*, 59(2):103–112, 2016.

[78] Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dieter Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. In *Robotics: Science and Systems*, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper's contributions and scope are specifically claimed in the abstract and introduction. Three of our contributions are summarized in the last paragraph of Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We explain why we simplify the original problem formulation in Section 3.1. We also discuss our further works in Section 7 .

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We propose and proof our theory in Section 3.1, and provide more theoretical analysis in Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.

- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We fully describe our settings in simulation and real world experiments in Section 6, Appendix D and Appendix E, and the information provided is enough to the reproducibility of our main experimental results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Please see: (1) `https://drive.google.com/file/d/12hK8wajdeDG3wN0_WWCt0NY0N9p1HVlA/view?usp=sharing` for our code; (2) `https://drive.google.com/file/d/1eU4203jaktImQehK0AUaG8uv7s0SAkrZ/view?usp=sharing` for foundation model checkpoints; (3) `https://drive.google.com/file/d/13msVuYWsC0TeQ4yTgPcrNBDFM9UqzNKQ/view?usp=sharing` for ALMI-X dataset.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.

- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Please see Section 6, Appendix B and Appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We define the evaluation metrics in Section 6 and Appendix D. And then we compare our method with other baselines, showing statistical results in Table 2. The statistical results are shown in Table 3, Table 11 and Table 12.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Please see our experimental setup details, and Appendix E for our real robot deployment setup.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We read the NeurIPS Code of Ethics, and make sure that our research conducted in the paper conform with the NeurIPS Code of Ethics in every respect.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss the societal impacts in Appendix F.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: No such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite works of related assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets are introduced.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: NA

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: NA

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: NA

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Theoretical Analysis

In this section, we provide more theoretical analysis for the Markov game in our method. We give the analysis of the zero-sum game to learn the locomotion policy π^l, and a similar analysis can be derived for learning π^u.

Recall that in learning π^l, we consider the lower body as *agent*, and the upper body is *adversary* that causes adversarial disturbances to the locomotion policy. Thus, the lower-body policy receives the command-following reward as $r^l(s, a^l, a^u)$, and the upper-body policy obtains a negative reward $-r^l(s, a^l, a^u)$. Formally, the value function $V^l(\pi^l, \pi^u)$ is defined as

$$V^l(s, \pi^l, \pi^u) := \mathbb{E}_{\pi^l, \pi^u}\left[\sum\nolimits_{t=0}^{T} r^l(s_t, a^l, a^u)\big|s_0 = s\right], \tag{8}$$

Then we have the value function as $V^l_\rho(\pi^l, \pi^u) = \mathbb{E}_{s\sim\rho}[V^l(s, \pi^l, \pi^u)]$, which is defined by the expectation of accumulated locomotion reward r^l. According to the theory of stochastic game [65], for any game $\mathcal{G}$, there exists a Nash equilibrium $(\pi_1^\star, \pi_2^\star)$ such that $V_\rho(\pi_1^\star, \pi_2) \le V_\rho(\pi_1^\star, \pi_2^\star) \le V_\rho(\pi_1, \pi_2^\star)$. Then we have

$$V_\rho(\pi^l, \pi^{u\star}) \le V_\rho(\pi^{l\star}, \pi^{u\star}) \le V_\rho(\pi^{l\star}, \pi^u), \quad \text{for all } \pi^l, \pi^u, \tag{9}$$

and in particular

$$V^{l\star}_\rho := V^l_\rho(\pi^{l*}, \pi^{u*}) = \max_{\pi^l}\min_{\pi^u} V^l_\rho(\pi^l, \pi^u) = \min_{\pi^u}\max_{\pi^l} V^l_\rho(\pi^l, \pi^u), \tag{10}$$

which signifies that π^l is learned to *maximize* the value function to better follow commands in locomotion, while π^u tries to *minimize* the value function, aiming to provide an effective disturbance to help learn a robust locomotion policy. Our goal in this setting is to develop algorithms to find ε-approximate Nash equilibrium, i.e. to find π^l such that

$$\big|\min_{\pi^u} V_\rho(\pi^l, \pi^u) - V^l_\rho(\pi^{l\star}, \pi^{u\star})\big| \le \varepsilon, \tag{11}$$

To solve this min-max problem in Eq. (10), we adopt an independent RL optimization process for both players. Specifically, the *agent* obtains $[(s_0, a^l_0, r^l_0), \dots, (s_T, a^l_T, r^l_T)]$, and *adversarial* obtains $[(s_0, a^u_0, r^u_0), \dots, (s_T, a^u_T, r^u_T)]$ by executing each policy in the game to sample a trajectory, where each player is oblivious to the actions of the other player. In our analysis, we adopt continuous policy parameterizations $x \mapsto \pi^u$, and $y \mapsto \pi^l$, where $x \in \mathcal{X} \subseteq \mathbb{R}^{d_1}$, $y \in \mathcal{Y} \subseteq \mathbb{R}^{d_2}$ are parameter vectors. Each player simply treats $V^l_\rho(x, y) \mapsto V^l_\rho(\pi^u, \pi^l)$ as a continuous optimization objective, and updates their policy using REINFORCE gradient estimator [66]. For episode i, the two players update their policies with stochastic gradient descent, as

$$x^{(i+1)} \leftarrow \mathcal{P}_\mathcal{X}(x^{(i)} - \eta_x \widehat{\nabla}^{(i)}_x), \quad y^{(i+1)} \leftarrow \mathcal{P}_\mathcal{Y}(y^{(i)} + \eta_y \widehat{\nabla}^{(i)}_y), \tag{12}$$

where $\mathcal{P}_\mathcal{X}$ denotes euclidean projection onto the convex set $\mathcal{X}$, with

$$\widehat{\nabla}^{(i)}_x \mapsto R^{(i)}_T \sum_{t=0}^{T} \nabla \log \pi^u(a^{u(i)}_t \mid s^{(i)}_t), \quad \widehat{\nabla}^{(i)}_y \mapsto R^{(i)}_T \sum_{t=0}^{T} \nabla \log \pi^l(a^{l(i)}_t \mid s^{(i)}_t), \tag{13}$$

where $R^{(i)}_T \mapsto \sum_{t=0}^{T} r^{l(i)}_t$. This process is independent for two players, as they optimize policies independently with policy gradients using their own experiences. Here, it is well-known that if players update their policies independently using online gradient descent/ascent with the same learning rate, the resulting dynamics may cycle, leading to poor guarantees [67, 68]. However, previous studies show that two-timescale rules help policy converge in simple minimax optimization settings [69, 70]. As a result, we follow recent studies in the Min-Max game [71] and use *two-timescale updates* for the two players, which is a simple modification of the usual gradient descent-ascent scheme for Min-Max optimization, in which the Min player uses a much smaller step size than the Max player. Next, to ensure that the variance of the REINFORCE estimator remains bounded, we require that both players use ε-greedy exploration in conjunction with the basic policy gradient updates.

Assumption A.1. Both players follow direct parameterization with ε-greedy exploration, as $\pi^u(a^u \mid s) = (1 - \varepsilon_x)\mathbb{1}_{s,a^u} + \varepsilon_x/|\mathcal{A}^u|$ and $\pi^l(a^l \mid s) = (1 - \varepsilon_y)\mathbb{1}_{s,a^l} + \varepsilon_y/|\mathcal{A}^l|$, where $\varepsilon_x, \varepsilon_y \in [0, 1]$ are the *exploration factors*.

Then the following theorem holds by following Assumption A.1 and the two-timescale rule of update.

Theorem A.2 (Restate of Theorem 3.1). *Given $\epsilon > 0$, suppose each policy has ε-greedy exploration scheme with factors of $\varepsilon_x \asymp \epsilon$ and $\varepsilon_x \asymp \epsilon^2$, under a specific two-timescale rule of the two players' learning-rate following the independent policy gradient, we have*

$$\max_{\pi^l}\min_{\pi^u} V_\rho(\pi^l, \pi^u) - \mathbb{E}\left[\frac{1}{N}\sum\nolimits_{i=1}^{N} \min_{\pi^u} V_\rho(\pi^u, \pi^{l(i)})\right] \le \epsilon \tag{14}$$

after N episodes, which results in a ϵ-approximate Nash equilibrium.

Proof. This proof basically follows Appendix A.2 of Daskalakis et al. [71], which proves that the Max player leads to the ϵ-approximate Nash equilibrium. We can follow a similar process to prove that the Min player also has the same property. According to [71], by following the REINFORCE gradient estimator, we have

$$\mathbb{E}_{\pi^u,\pi^l}\|\widehat{\nabla}_x - \nabla_x V_\rho^l(x,y)\|^2 \le 24\frac{|\mathcal{A}^u|^2}{\varepsilon_x \zeta^4}, \quad \text{and} \quad \mathbb{E}_{\pi_x,\pi_y}\|\widehat{\nabla}_y - \nabla_y V_\rho^l(x,y)\|^2 \le 24\frac{|\mathcal{A}^l|^2}{\varepsilon_y \zeta^4}, \tag{15}$$

where ζ is the stopping probability of the Markov game. Then according to the performance difference lemma [72], we have

$$V_\rho(\pi^u,\pi^l) - V_\rho(\pi^{u'},\pi^l) = \sum_{s\in\mathcal{S}} \tilde{d}_\rho^{\pi^u,\pi^l}(s)\mathbb{E}_{a\sim\pi^u(\cdot|s)}\mathbb{E}_{b\sim\pi^l(\cdot|s)}\left[A^{\pi^{u'},\pi^l}(s,a,b)\right] \tag{16}$$

Then, for a policy π, let $\pi_1^\star(\pi^l) \in \Pi_1^\star(\pi^l)$ denote a policy minimizing $\|\frac{d_\rho^{\pi_1,\pi^l}}{\rho}\|_\infty$, then we have

$$\begin{aligned} V_\rho^l(x,y) - \min_{x'} V_\rho(x',y) &\le V_\rho(\pi^u,\pi^l) - V_\rho(\pi_1^*(\pi^l),\pi^l) && (17)\\ &= \sum_{s,a} \tilde{d}_\rho^{\pi_1^\star(\pi^l),\pi^l}(s)\pi_1^\star(\pi^l)(a\mid s)\mathbb{E}_{b\sim\pi^l(\cdot|s)}[-A^{\pi^u,\pi^l}(s,a,b)] && (18)\\ &\le \sum_s \tilde{d}_\rho^{\pi_1^\star(\pi^l),\pi^l}(s)\max_a \mathbb{E}_{b\sim\pi^l(\cdot|s)}[-A^{\pi^u,\pi^l}(s,a,b)] && (19)\\ &\le \left\|\frac{\tilde{d}_\rho^{\pi_1^\star(\pi^l),\pi^l}}{\tilde{d_\rho}^{\pi^u,\pi^l}(s)}\right\|_\infty \sum_s \tilde{d}_\rho^{\pi^u,\pi^l}(s)\max_a \mathbb{E}_{b\sim\pi^l(\cdot|s)}[-A^{\pi^u,\pi^l}(s,a,b)]. && (20)\end{aligned}$$

We observe that $\left\|\frac{\tilde{d}_\rho^{\pi_1^\star(\pi^l),\pi^l}}{\tilde{d_\rho}^{\pi^u,\pi^l}}\right\|_\infty \le \frac{1}{\zeta}\left\|\frac{d_\rho^{\pi_1^\star(\pi^l),\pi^l}}{\rho}\right\|_\infty \le \frac{1}{\zeta}C_{\mathcal{G}}$, where $C_{\mathcal{G}}$ is the minimax mismatch coefficient for the game $\mathcal{G}$. Then we have

$$\begin{aligned} &\sum_{s,a} \tilde{d}_\rho^{\pi^u,\pi^l}(s)\max_a \mathbb{E}_{b\sim\pi^l(\cdot|s)}[-A^{\pi^u,\pi^l}(s,a,b)]\\ &= \max_{\bar{x}\in\Delta(\mathcal{A}^u)^{|\mathcal{S}|}} \sum_{s,a} \tilde{d}_\rho^{\pi^u,\pi^l}(s)\bar{x}_{s,a}\mathbb{E}_{b\sim\pi^l(\cdot|s)}[-A^{\pi^u,\pi^l}(s,a,b)]\\ &= \max_{\bar{x}\in\Delta(\mathcal{A}^u)^{|\mathcal{S}}} |\sum_{s,a} \tilde{d}_\rho^{\pi^u,\pi^l}(s)(\pi^u(a\mid s) - \bar{x}_{s,a})\mathbb{E}_{b\sim\pi^l(\cdot|s)}[Q^{\pi^u,\pi^l}(s,a,b)]\\ &= \max_{\bar{x}\in\Delta(\mathcal{A}^u)^{|\mathcal{S}|}} \sum_{s,a} \tilde{d}_\rho^{\pi^u,\pi^l}(s)((1-\varepsilon_x)x_{s,a} + \varepsilon_x A^{-1} - \bar{x}_{s,a})\mathbb{E}_{b\sim\pi^l(\cdot|s)}[Q^{\pi^u,\pi^l}(s,a,b)],\\ &\le \max_{\bar{x}\in\Delta(\mathcal{A}^u)^{|\mathcal{S}|}} \sum_{s,a} \tilde{d}_\rho^{\pi^u,\pi^l}(s)((1-\varepsilon_x)x_{s,a} + \varepsilon_x A^{-1} - \varepsilon_x\bar{x}_{s,a} - (1-\varepsilon_x)\bar{x}_{s,a})\mathbb{E}_{b\sim\pi^l(\cdot|s)}[Q^{\pi^u,\pi^l}(s,a,b)]\\ &\le (1-\varepsilon_x)\max_{\bar{x}\in\Delta(\mathcal{A}^u)^{|\mathcal{S}|}} \sum_{s,a} \tilde{d}_\rho^{\pi^u,\pi^l}(s)(x_{s,a} - \bar{x}_{s,a})\mathbb{E}_{b\sim\pi^l(\cdot|s)}[Q^{\pi^u,\pi^l}(s,a,b)] + \frac{2\varepsilon_x}{\zeta^2}\\ &= \max_{\bar{x}\in\Delta(\mathcal{A}^u)^{|\mathcal{S}|}} \langle \nabla_x V_\rho(x,y), x - \bar{x}\rangle + \frac{2\varepsilon_x}{\zeta^2}, \qquad (21)\end{aligned}$$

where the last equation holds since $Pr[T \ge t] \le (1-\zeta)^t$ for any $t \ge 0$ that for any $\rho \in \Delta(\mathcal{S})$,

$$\begin{aligned} \nabla_x V_\rho(x,y) &= \mathbb{E}_{\tau\sim Pr^{\pi_x,\pi_y}(\cdot|s_0)}\left[\sum_{t=0}^{T}(\nabla_x \log \pi_x(a_t|s_t))Q^{\pi_x,\pi_y}(s_t,a_t,b_t)\right] && (22)\\ &= \sum_{s\in\mathcal{S}} \mathbb{E}_{a\sim\pi_x(\cdot|s)}\mathbb{E}_{b\sim\pi_y(\cdot|s)}\left[\tilde{d}_\rho^{\pi_x,\pi_y}(s)(\nabla_x \log \pi_x(a|s))Q^{\pi_x,\pi_y}(s,a,b)\right]. && (23)\end{aligned}$$

Thus, for any $s \in \mathcal{S}, a \in \mathcal{A}$, we have

$$\frac{\partial V_\rho(x,y)}{\partial x_{s,a}} = (1-\varepsilon_x)\tilde{d}_\rho^{\pi_x,\pi_y}(s)\mathbb{E}_{b\sim\pi_y(\cdot|s)}\left[Q^{\pi_x,\pi_y}(s,a,b)\right],$$

and so it follows that
$$\left|\frac{\partial V_\rho(x,y)}{\partial x_{s,a}}\right| \le \frac{d_\rho^{\pi_x,\pi_y}(s)}{\zeta}\left|\mathbb{E}_{b\sim\pi_y(\cdot|s)}\left[Q^{\pi_x,\pi_y}(s,a,b)\right]\right|.$$

According to Eq. (17) and (21), we have

$$V_\rho(\pi^u, \pi^l) - \min_{x'} V_\rho(x', \pi^l) \le \min_{\pi_1 \in \Pi_1^*(\pi^l)} \left\| \frac{d_\rho^{\pi_1, \pi^l}}{\rho} \right\|_\infty \left(\frac{1}{\zeta} \max_{\bar{x} \in \Delta(\mathcal{A}^u)^{|\mathcal{S}|}} \left\langle \nabla_x V_\rho(\pi^u, \pi^l), \pi^u - \bar{x} \right\rangle + \frac{2\varepsilon_x}{\zeta^3} \right), \tag{24}$$

The term in the right side can be bounded by $O\left(\frac{\epsilon C_g}{\zeta}\right)$ according to the gradient dominance condition in Lemma 1a of [71]. Then, according to Theorem 2a of [71], suppose each policy has ε-greedy exploration scheme with factors of $\varepsilon_x \asymp \epsilon$ and $\varepsilon_x \asymp \epsilon^2$, the average performance difference can be bounded. Specifically, under a specific two-timescale rule of the two players' learning-rate, we have

$$\max_{\pi^l} \min_{\pi^u} V_\rho(\pi^l, \pi^u) - \mathbb{E}\left[\frac{1}{N} \sum_{i=1}^{N} \min_{\pi^u} V_\rho(\pi^u, \pi^{l(i)}) \right] \le O\left(\frac{\epsilon C_g}{\zeta} \right), \tag{25}$$

which concludes our proof. □

B Implementation Details

B.1 State and Action Space

In this section, we introduce the detailed observation and action-space information of policies used in experiments. The adversarial iterations use the same state space setting. We use 21 DoF of the H1-2 robot without its wrist joints. The details are in Table 7.

Table 7: State and action space information in ALMI setting.

State Term	Lower dim.	Upper dim.	Whole dim.
Base angular velocity	3	3	3
Base gravity	3	3	3
Commands	3 (velocity)	9 (motion)	12 (velocity+motion)
DoF position	21	21	21
DoF velocity	21	21	21
Actions	12 (lower)	9 (upper)	21 (whole)
Periodic phase	2	2	2
Total dim	65	68	83

The policies use the PPO [11] algorithm for training, where the observation of the critic policy has 3 additional dimensions of base linear velocity compared to the state space of the actor policy, which is often called *privileged information* in robotics. For action space, the lower body policy uses 12 DoF of two leg joints, the upper body uses 9 DoF of one waist joint and two arms, and the whole body policy uses all 21 DoF joints.

B.2 Training Detail of the Locomotion Policy

Algorithm Description We employ PPO to train the locomotion policy, which consists of two key components: the policy and the environment. Trajectories are generated by deploying the policy on the robot within the environment, and the collected data is then used to update the policy. Additionally, the curriculum factors are used to adjust at the end of each episode. The algorithmic description is given in Algorithm 1.

Implementation Details To enable the robot to exhibit a regular gait during movement and remain stationary when the velocity command is zero, we design a gait phase parameter $\boldsymbol{\phi}_t = (\phi_{t,\text{left}}, \phi_{t,\text{right}})$ as an observation term, which is updated as:

$$\phi_{t+1,\text{left}} = \phi_{t,\text{left}} + f \times \mathrm{d}t, \tag{26}$$

$$\phi_{t+1,\text{right}} = \phi_{t+1,\text{left}} + \psi, \tag{27}$$

where $f, \mathrm{d}t, \psi$ are gait frequency, time step and gait offset, respectively. We use PPO to train the policy and employ an asymmetric actor-critic architecture, where the critic is granted access to the base linear velocity $\boldsymbol{v}_t$ as privileged information. For the first iteration, we apply an open-loop controller to control the upper-body to track the reference motion trajectory. In subsequent iterations, we employ the trained upper-body policy as the upper-body controller.

Reward Design We divide the reward terms into three parts according to their role: *penalty* rewards to prevent unwanted behaviors, *regularization* to refine motion, and *task* reward to achieve goal tracking (velocity commands or upper body DoF position). The details are in Table 1. The three iterations use the same reward terms and weights.

Algorithm 1 Training process of the locomotion policy

```
Require: max iterations $N$, max episode length $l^{\text{sl}}_{\text{max}}$
 1: Initialize policy $\pi^l_\theta$, value function $V^l_\phi$
 2: for iteration $= 1, \dots, N$ do
 3:     Collect trajectories by running policy $\pi^l_\theta$ in the environment
 4:     // Update curriculum at episode termination
 5:     if $l^{\text{msl}} > 0.8 \times l^{\text{sl}}_{\text{max}}$ then
 6:         $\alpha_d \leftarrow \min(\alpha_d + w, \|\mathcal{M}\| - w)$
 7:     else
 8:         $\alpha_d \leftarrow \max(\alpha_d - 2w, 0)$
 9:     end if
10:     if $\alpha_d == \|\mathcal{M}\| - w$ then
11:         $\alpha_s \leftarrow \min(\alpha_s + 0.05, 1), \alpha_d \leftarrow 0$
12:     else if $\alpha_d == 0$ then
13:         $\alpha_s \leftarrow \max(\alpha_s - 0.01, 0)$
14:     end if
15:     Perform PPO update with GAE advantage update
16: end for
17: return Trained policy $\pi_\theta$
```

B.3 Training Detail of the Motion-Tracking Policy

Algorithm Description During the training of the upper-body motion tracking policy, we use the pre-trained lower-body policy π^l to execute the locomotion command and simultaneously introduce disturbance to the upper-body. We begin by sampling the locomotion velocity commands from a narrow range and compute the tracking error of the motion-tracking task. At each episode termination, if the tracking error is smaller than a threshold, the command sampling range will be extended to generate more intensive disturbance to the upper-body. We use PPO to train the motion tracking policy to simply follow the reference DoF position sampled from the dataset. We did not use the 6D pose of keypoints as an observation because the computation is based on the robot's root pose, which will differ from that of motion dataset under omnidirectional commands. In contrast, joint angles avoid such dependency and provide sufficient information for motion tracking.

Curriculum Setting In learning the motion tracking policy in the upper body, the lower-body command follows a curriculum by adjusting the range of the velocity commands. Formally,

$$\boldsymbol{c}^l_{\text{min}} \leftarrow \begin{cases} \max(\boldsymbol{c}^l_{\text{min}} - 0.1, \boldsymbol{C}^l_{\text{min}}), & \exp(-0.5\|\hat{\boldsymbol{q}}^{\text{u}} - \boldsymbol{q}^{\text{u}}\|^2_2) > d^u \\ \max(\boldsymbol{c}^l_{\text{min}} + 0.1, \boldsymbol{C}^l_{\text{min}}), & \text{otherwise} \end{cases}, \tag{28}$$

$$\boldsymbol{c}^l_{\text{max}} \leftarrow \begin{cases} \min(\boldsymbol{c}^l_{\text{max}} + 0.1, \boldsymbol{C}^l_{\text{max}}), & \exp(-0.5\|\hat{\boldsymbol{q}}^{\text{u}} - \boldsymbol{q}^{\text{u}}\|^2_2) > d^u \\ \min(\boldsymbol{c}^l_{\text{max}} - 0.1, \boldsymbol{C}^l_{\text{max}}), & \text{otherwise} \end{cases}, \tag{29}$$

where $\boldsymbol{c}^l_{\text{min}}, \boldsymbol{c}^l_{\text{max}}$ are the lower and upper bound of the current command range, $\boldsymbol{c}^l$ is sampled from$[\boldsymbol{c}^l_{\text{min}}, \boldsymbol{c}^l_{\text{max}}]$ during training. $[\boldsymbol{C}^l_{\text{min}}, \boldsymbol{C}^l_{\text{max}}]$ is the range limit of pre-defined commands. $\hat{\boldsymbol{q}}^{\text{u}}$, $\boldsymbol{q}^{\text{u}}$ and d^u are the reference upper body joint position, upper body joint position and the threshold of motion tracking error, respectively. The related terms and values are given in Table 8.

Table 8: Upper body curriculum terms and values.

Term	Value
$\boldsymbol{C}^l_{\text{min}}$	[-0.7, -0.5, -0.5]
$\boldsymbol{C}^l_{\text{max}}$	[0.7, 0.5, 0.5]
d^u	0.9

Reward Design Table 2 gives the reward terms for the motion tracking policy in the upper body. The upper-body policies of adversarial iterations all use these reward terms and weights.

B.4 PPO Hyperparameters

We use GAE to estimate the advantage and CLIP-PPO to train our policy. The total loss is given by:

$$L_{\text{total}} = L_{\text{policy}} + w_v L_{\text{value}} - w_s S, \tag{30}$$

where

$$L_{\text{policy}} = -\mathbb{E}[\min(r_i(\theta)\hat{A}_i, \text{clip}(r_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i)], \tag{31}$$

$$L_{\text{value}} = \mathbb{E}[(V(s_i) - R_i)^2], \tag{32}$$

$$S = \mathbb{E}[-\pi_\theta(a|s)\log \pi_\theta(a|s)], \tag{33}$$

where $r_i(\theta) = \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{old}}(a_i|s_i)}$ and $\hat{A}_i$ is the estimated advantage. w_v and w_s are value loss coefficient and entropy term coefficient, respectively. The hyperparameters of the PPO algorithm and the information of the network backbone are listed in Table 10.

B.5 Domain Randomization

Domain randomization is a popular technique for improving domain transfer, often used in a zero-shot setting when the target domain is unknown or cannot easily be used for training [73]. In order to adapt the trained policy to the real world, we employ the domain randomization technique during training to facilitate robust sim-to-sim and sim-to-real transfer [36, 37]. We give the domain randomization terms and ranges used during the training process, and the details are in Table 9.

Table 9: Domain randomization terms and ranges.

Term	Value
Dynamics Randomization	
Friction	$\mathcal{U}(0.1, 1.25)$
Base mass	$\mathcal{U}(-3, 5)$ kg
Link mass	$\mathcal{U}(0.9, 1.1) \times$ default kg
Base CoM	$\mathcal{U}(-0.1, 0.1)$ m
Control delay	$\mathcal{U}(0, 40)$ ms
External Perturbation	
Push robot	interval = 10s, v_{xy} = 1m/s
Randomized Terrain	
Terrain type	trimesh, level from 0 to 10
Velocity Command	
Linear x velocity	$\mathcal{U}(-1.0, 1.0)$
Linear y velocity	$\mathcal{U}(-0.3, 0.3)$
Angular yaw velocity	$\mathcal{U}(-0.5, 0.5)$

Table 10: Hyperparameters related to PPO

Hyperparameter	Default Value
Actor lstm size	[64]
Actor MLP size	[64, 32]
Critic MLP size	[64, 32]
Optimizer	Adam
Batch size	4096
Mini Batches	4
Learning epoches	8
Activation	elu
Entropy coef(w_s)	0.01
Value loss coef(w_v)	1.0
Clip param	0.2
Max grad norm	1.0
Init noise std	0.8
Learning rate	1e-3
Desired KL	0.01
GAE decay factor(λ)	0.95
GAE discount factor(γ)	0.998
Curriculum window size(w)	40

C Data Collection and Model Details

Dataset Overview The ALMI-X dataset contains 1989 motions combined with 41 commands, resulting 81,549 trajectories totally. Each trajectory is organized as $\tau = \{\tau_s^t, \tau_a^t, \tau_{\text{dof\_pos}}^t, \tau_{\text{trans}}^t, \tau_{\text{rot}}^t\}_{t=1}^T$, respectively representing robot states, actions, joint angles, global position and global orientation, where the global position and global orientation are usually considered as privilege information that is missing in the state space. The statistical results shown in Fig. 4, Fig. 5 and Fig. 6 demonstrate the advantages of ALMI-X, and the experimental results of Appendix D further indicate that our dataset is higher quality and more suitable for foundation model training. The dataset is available at `https://almi-humanoid.github.io/`.

Data Collection Details (i) For the lower-body, we first categorize command directions into 11 types (8 translation directions: front, back, left, right, front-left, front-right, back-left, and back-right; 2 rotation directions: left and right; and keep standing) according to different combination of linear and angular velocity command, and define 4 difficulty levels for command magnitudes, as shown in Table 11. We sample velocity value for the first 3 levels and task a fix value for the 'fix' level, which corresponds to a text that does not contain speed-related modifiers, e.g. just 'go forward' without 'slowly' or 'fast'. We take the lower-body commands according to the difficulty levels with corresponding ranges, combining with different direction types. In addition, we set a special category for standing, resulting in 41 categories totally. (ii) For the upper body, we select motions according to §4 and remove those that are overly large in amplitude or difficult

Table 11: Command difficulty level setting.

	Command		
Level	$\hat{v}_{\text{x},\text{t}}$	$\hat{v}_{\text{y},\text{t}}$	$\hat{\omega}_{\text{yaw},\text{t}}$
easy	[0.2,0.4]	[0.2,0.3]	[0.2,0.3]
medium	[0.4,0.5]	[0.3,0.4]	[0.3,0.4]
hard	[0.5,0.7]	[0.4,0.5]	[0.4,0.5]
fix	0.4	0.4	0.4

to distinguish, resulting in 680 different motions. However, these motions have an uneven distribution, with each motion category associated with a different number of trajectories. As shown in the left plot of Fig. 4, for instance, the number of steps for "wave" greatly exceeds that of "salute". Through empirical evaluation, we find that imbalanced data distribution negatively impacts the training performance of the foundation model.

To address this problem, we classify the motions into 30 categories based on their corresponding texts. Then we count the number of steps in each category and augment those with fewer steps by repeating their motion sequences multiple times during data collection. This is not equivalent to mere data duplication, since environmental interactions cause the robot to produce slightly different trajectories each time, despite following the same reference motion. After that, the data volume across all categories is approximately the same, as shown in Fig. 4.

After data augmentation, we obtain a total of 1,989 motions. When combined with different direction and difficulty commands, these motions generate a total of 81,549 trajectory sequences. Fig. 5 shows the planar position of all steps in space. The visualization result shows that despite the accumulated errors from the policy and the limitations of the simulator, the robot is still able to execute velocity commands in a reasonably correct manner. For instance, under the ‘go forward’ command, the steps are distributed along the positive x-axis, enabling the foundation model to learn representations across the space.

Fig. 6 illustrates the distribution of the upper body-hand positions of dataset. The visualization shows that the upper body motions in the dataset basically allow the robot arms to cover the entire activity space. When we train the lower body policy, we set the arm curriculum which is detailed in §3.2 to gradually expand the activity range of the two arms from the default position to the whole activity space.

Comparison to Humanoid-X [13] data Compared to the existing text-to-action dataset, Humanoid-X [13], our dataset exhibits much higher quality, which can be utilized for relatively stable open-loop control of the robot in simulations without causing the robot to lose control. This is because the RL policy that UH-1 employs to track motions have complex sources and can introduce dangerous or unreasonable behavior. However, due to the robust lower-body policy and expressive upper-body policy obtained through adversarial and curriculum learning, our collected ALMI-X dataset exhibits high executability. This attribute benefits the learning of the robot foundation model via a supervised learning paradigm, since the learned model would be better for deployment and also ensures safety in the real robot.

D Experiment Results of Foundation Model

In this section, we introduce the architecture details and inference process of the Transformer-based foundation model. Then, we give experimental results and analysis of the trained model.

Architecture Details During the training process, we first extract a segment of states and actions with fixed length H and its corresponding language description $\mathcal{T}$, then concatenate the state and action for the same frame resulting in a new sequence $\{(s_i, a_i), (s_{i+1}, a_{i+1}), ..., (s_{i+H-1}, a_{i+H-1})\}$, each of which is projected onto an embedding vector $x_i = \texttt{Linear}(o_i, a_i)$. We use CLIP [74] and a linear layer process the language description $\mathcal{T}$ into a vector of the same dimension $x_{\text{text}} = \texttt{Linear}(\texttt{CLIP}(\mathcal{T}))$ These vectors serve as input to the transformer decoder, which predicts the subsequent actions corresponding to each input. The computation process of the l-th transformer decoder layer and the output layer can be represented as follows:

$$x^l_{\text{text}}, x^l_i, ..., x^l_{i+H-1} = \texttt{CausalSA}(x^{l-1}_{\text{text}}, x^{l-1}_i, ..., x^{l-1}_{i+H-1}), \tag{34}$$

$$\{\hat{a}_{i+1}, ..., \hat{a}_{i+H}\} = \texttt{Linear}(x^N_{\text{text}}, x^N_i, ..., x^N_{i+H-1}) \tag{35}$$

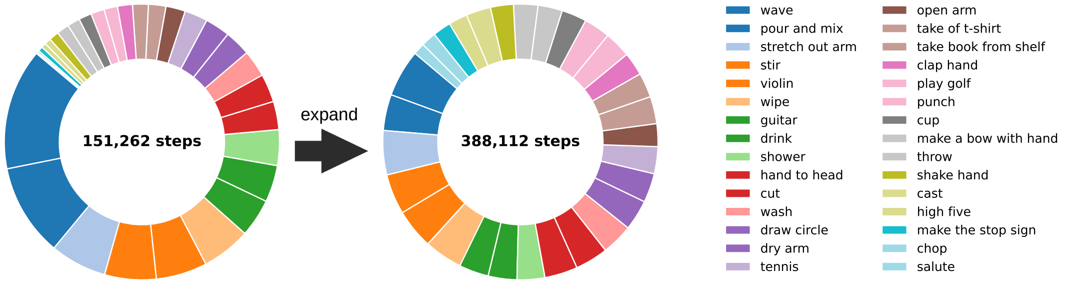


Figure 4: Percentage of steps for different categories of motions before and after data augmentation.

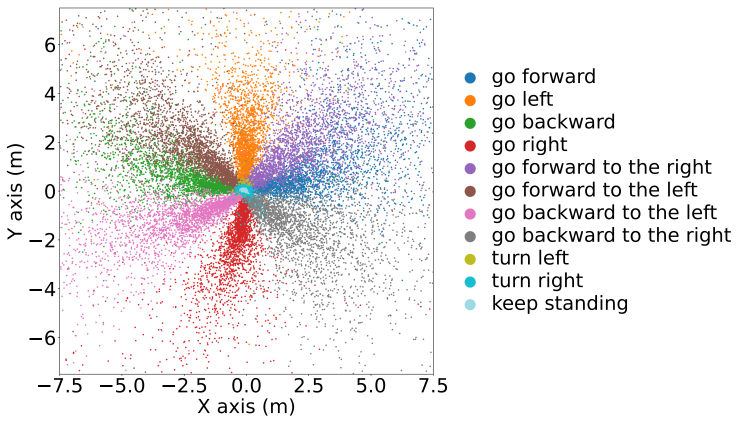


Figure 5: The visualization of $x - y$ coordinates of the robot for each step in the dataset. We down-sample the data for visualization.

where N is the number of transformer decoder layers. $\hat{a}_j$, where $i + 1 \leq j \leq i + H + 1$, represent the predict action integrating text command information and historical state-action pairs. Mean square errors between predict and ground truth actions are used for policy update.

Model Inference In inference, the input text vector t_{text} remains constant throughout the execution of the task. The robot executes the action decoded from the last vectors of the policy output. It is then concatenated with the newly acquired state, which is then computed into an embedding vector and serve as the last vector of the input for the policy to predict the next action. In the initial stage, when the length h of the historical state-action pairs is less than H, the robot will execute the action decoded using the vector corresponding to the latest state-action pair rather than the last one.

Implementation Details We investigated multiple designs for language-guided whole-body humanoid control through supervised learning using the ALMI-X dataset. We evaluate the Transformer architecture with different input sequence lengths for the *closed-loop* robot control approach, while also exploring the *open-loop* control approach of prediction and execution of the entire sequence with a tokenizer like UH-1 [13]. Specifically, the Transformer is trained with different state-action sequence lengths. We selected 20 and 400 as the input sequence lengths for the *closed-loop* control, while the *open-loop* control approach predicts the entire motion sequence with a maximum sequence length of 700 (175 after tokenization). For a shorter sequence, we expect the model to focus more on short-term historical information, allowing more robust locomotion performance. As for the longer sequence, we aim to integrate information from complete motion sequences in modeling, thereby better responding to text commands. For tokenization, we employ the same VQ-VAE [75] architecture as in

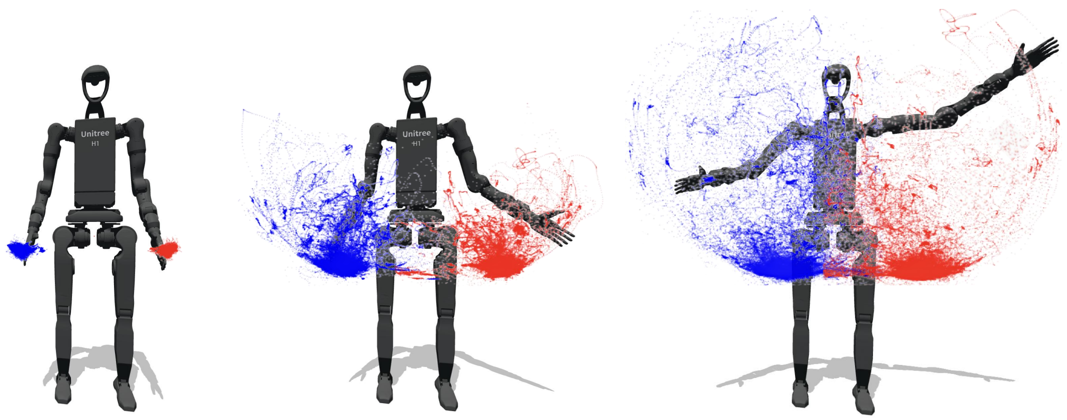


Figure 6: We illustrate the hand positions in the dataset relative to the robot's hand joints. From left to right, scaling factors of 0.05, 0.5, and 1.0 are applied to the arm joint angle offsets from the default pose, effectively modulating the motion amplitude. Larger scaling results in more pronounced arm movements.

Table 12: Average velocity of Foundation models with different text commands.

Text commands w.r.t. lower body	Average Velocity $\bar{v}_x$	$\bar{v}_y$	$\bar{\omega}_{\text{yaw}}$
CL-20sl			
forward fast	0.48	0.18	0.08
left fast	0.07	0.28	0.00
backward to left slowly	-0.19	0.21	0.00
turn left slowly	0.02	0.02	0.08
go right fast	-0.11	-0.37	-0.07
keep standing	0.00	0.00	0.01
CL-400sl			
forward fast	0.87	-0.05	-0.10
left fast	0.53	0.42	-0.07
backward to left slowly	0.35	0.30	-0.01
turn left slowly	0.53	0.16	0.2
go right fast	0.47	-0.50	-0.10
keep standing	0.22	0.08	0.00

Table 13: Survival duration (SD) and success rates of upper-body(SR_{up})/lower-body(SR_{low}) with different text commands.

Text commands	Metrics SR_{low}	SR_{up}	SD
CL-20sl			
go forward slowly and wave left.	1.00	0.20	8.0
go backward moderately and wave right.	1.00	0.20	8.0
go right fast and wave both.	1.00	0.40	8.0
CL-400sl			
go forward slowly and wave left.	1.00	1.00	2.14
go backward moderately and wave right.	1.00	1.00	2.54
go right fast and wave both.	0.00	1.00	3.57
OL			
go forward slowly and wave left.	0.00	1.00	0.31
go backward moderately and wave right.	0.00	1.00	0.29
go right fast and wave both.	0.00	1.00	0.32

T2M-GPT [76] to tokenize four state-action pairs into one token; while for the non-tokenized design, we utilize a linear layer to project both the text command feature and state-action pairs into a shared latent space.

Evaluation Metrics Since training a foundation model to follow all text commands in ALMI-X is quite challenging, we conduct preliminary investigations using only wave-relative motion. We evaluate the models using different control approaches and input sequence lengths with different text commands in MuJoCo simulations, repeating each command 5 times with each test lasting 8 seconds. we adopt the following notation: $\mathrm{CL}-x\mathrm{sl}$ denotes the closed-loop control approach without tokenization, where x represents the input sequence length (e.g., $\mathrm{CL}-20\mathrm{sl}$). OL indicates the open-loop control approach with tokenization, which predicts and executes the entire motion sequence in a single forward pass. We evaluated several metrics, including robot survival time (up to 8 seconds), velocity during test time, and success rate of lower-body and upper-body. We consider the lower-body successful if the robot's movement direction matches the text command, and the upper-body successful if the robot waves the correct hand.

Experiment Results Results in Table 13 demonstrate that the open-loop control approach, which predicts and executes the entire action sequence based on a single text command, fails to maintain the robot's balance

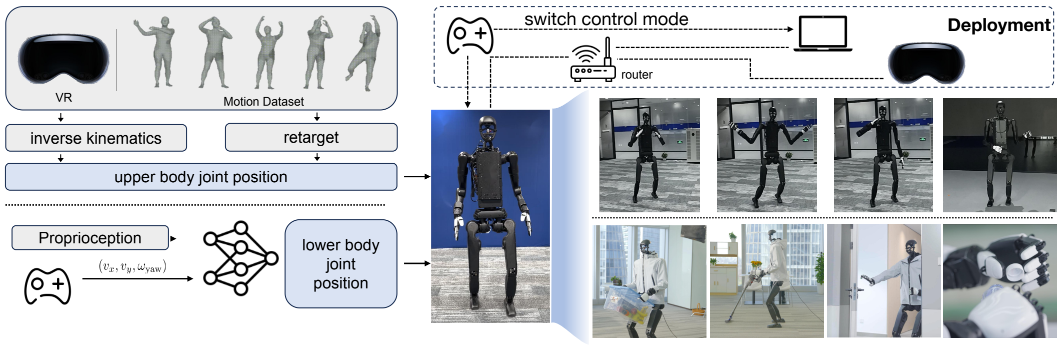


Figure 7: Real word deployment overview. The deployment framework contains an Apple Vision Pro, a router, a Unitree H1-2 robot and a PC which runs our policy and teleoperation server. The upper-body target DoF pos is given by teleoperation or motions from our motion dataset. We can switch the control mode via the remote controller. Through this framework, the robot can perform upper-body motion during movement, and diverse loco-manipulation tasks.

and leads to an extremely low survival time. In contrast, the closed-loop control method exhibits more robust locomotion and superior lower-body text-following performance. Table 12 demonstrates that the closed-loop approach with a short input sequence length enables robot's movement direction adapts to commands such as $forward$ and $right$ and its velocity adjusts to modifiers like $fast$ and $slowly$, aligning more accurately with the sampled velocity commands as shown in Table 11. However, the approach with a longer sequence length results in poorer lower-body motion performance in both movement direction and velocity. It can be inferred that the long input sequence length plays a crucial role in the upper-body success rate, since OL and $\mathrm{CL}-400\mathrm{sl}$ achieve much better upper-body compliance with text commands, and focusing the model exclusively on short-term historical information brings locomotion stability and better command performance of the lower body. These experimental results indicate that training a humanoid foundation control model using supervised learning and a large-scale high-quality dataset represents a feasible approach with substantial exploratory potential. However, we find that the current model faces challenges when extending the training data to the entire ALMI-X, indicating limitations of the current model in handling highly diverse behaviors.

E More Experiments Regarding Adversarial Training

We also conducted more experiments to verify the rationality and advantages of the simplification to the original method.

Without simplifying the max-min problem, four separate policies would be required in each round of policy learning. Specifically, in round 1, when updating, an additional adversary is required; when updating, an additional adversary is required. It would be computationally expensive to train an additional adversary in each round. As a result, we change the inner-loop optimization of the adversary from the parameter space to the command space, which allows us to sample adversarial command to approximate the effects of training adversarial policies. We add additional experiments to show the necessity of the simplification:

Train an Adversary. If we do not adopt the simplified approach, we would instead directly train an adversary to attack the policy. However, it is very challenging to properly control the strength of the adversary. Take the training of lower body policy as example, we jointly train an upper-body adversary to minimize the lower-body reward. We vary $action\_clip\_scale$ to control adversary strength, where $action\_clip = 100 \times action\_clip\_scale$. The results are shown in the Table 14. It can be seen that large adversary collapse training, while weak adversaries yield poor robustness. In contrast, our curriculum-based method, guided by motion difficulty, provides effective and stable adversarial training.

Table 14: Comparison before and after simplification of the adversarial training.

	Metrics							
Method	$E_{\mathrm{vel}} \downarrow$	$E_{\mathrm{ang}} \downarrow$	$E^{\mathrm{upper}}_{\mathrm{jpe}} \downarrow$	$E^{\mathrm{upper}}_{\mathrm{kpe}} \downarrow$	$E^{\mathrm{upper}}_{\mathrm{action}} \downarrow$	$E^{\mathrm{lower}}_{\mathrm{action}} \downarrow$	$E_{\mathrm{g}} \downarrow$	Survival $\uparrow$
ALMI (Ours)	**0.2202**	**0.4812**	**0.2116**	**0.0458**	**0.0600**	**0.0175**	**0.8551**	**0.9723**
$action\_clip\_scale = 0.1$	1.2610	5.3445	/	/	/	6.4459	1.7446	0
$action\_clip\_scale = 0.01$	1.3707	6.8632	/	/	/	6.1924	2.3056	0

Policy with adversarial goals. Another way to implement adversarial training is to assign each policy two objectives. For this experiment, we train upper and lower policies jointly, each maximizing its own reward while minimizing the other's. We vary adversarial strength via reward weights (e.g., -1 for strong adversary and -0.1 for weak adversary). Table 15 show adversarial weight greatly affects training, and tuning it is nontrivial. The training of weak π^u_a+ strong π^l_a and strong π^u_a+ strong π^l_a are failed. Our simplified method ensures stable and effective adversarial interaction. Among converged settings, our method outperforms others in all metrics.

Table 15: The lower and upper policies are updated simultaneously, and both have two sets of goals.

	Metrics							
Method	$E_{\mathrm{vel}} \downarrow$	$E_{\mathrm{ang}} \downarrow$	$E^{\mathrm{upper}}_{\mathrm{jpe}} \downarrow$	$E^{\mathrm{upper}}_{\mathrm{kpe}} \downarrow$	$E^{\mathrm{upper}}_{\mathrm{action}} \downarrow$	$E^{\mathrm{lower}}_{\mathrm{action}} \downarrow$	$E_{\mathrm{g}} \downarrow$	Survival $\uparrow$
ALMI (Ours)	**0.2202**	**0.4812**	**0.2116**	**0.0458**	**0.0600**	**0.0175**	**0.8551**	**0.9723**
weak π^u_a+ weak π^l_a	0.3415	1.4413	0.29469	0.0526	0.0970	0.0183	1.0250	0.9612
strong π^u_a+ weak π^l_a	1.1220	4.6715	2.7251	0.1840	0.2450	4.6937	1.1702	0.3524

F Real-Robot Deployment

Hardware and Deployment We conduct experiments using the Unitree H1-2 robot equipped with the ROBOTERA XHAND robotic hand, applying ALMI-trained policies to the lower body control and various interfaces (i.e., open-loop controller, ALMI policy, and VR device) for the upper body control, as shown in

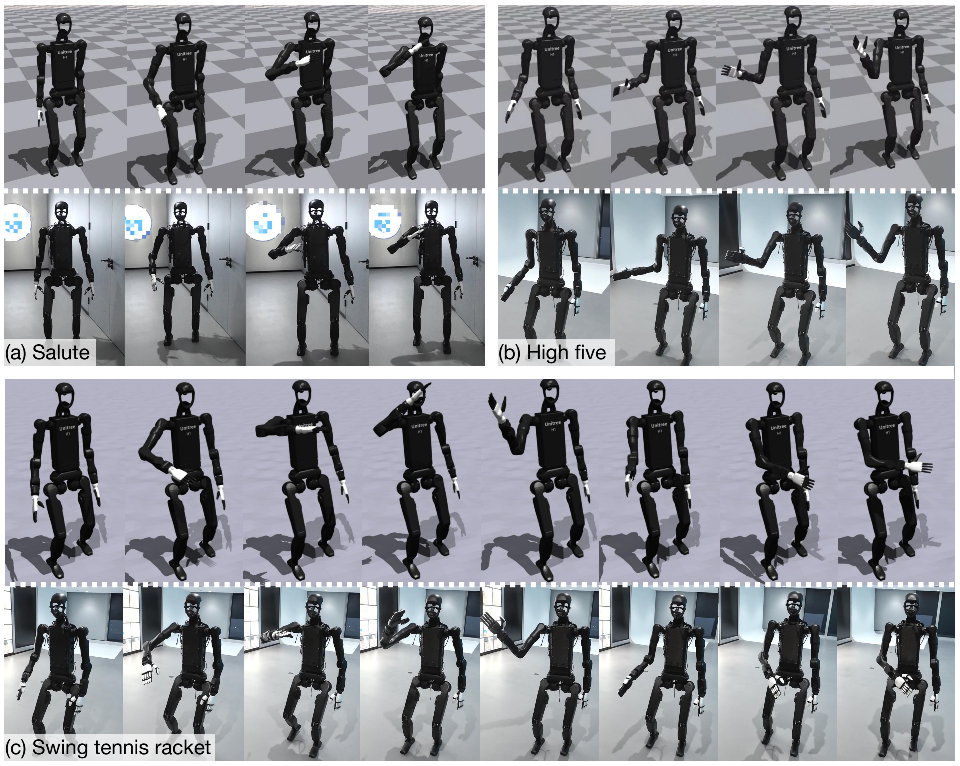


Figure 8: Upper body behavior of the robot during locomotion compared between simulator and real world. In simulator, robot keeps standing to show the original upper body motion; In real world, robot goes forward while doing motion.

Fig. 7. Specifically, (i) we can use an open-loop controller that sends the target joint positions of the pre-loaded motions; (ii) we can obtain actions of the upper body via the policy π^u, which uses the reference joint positions of the preloaded motion as input. In real-world experiments, we find that open-loop control has higher tracking accuracy when the robot remains standing, but the learned policy shows better stability during movement. (iii) The other one is combining with teleoperation systems, which enable the robot to perform loco-manipulation tasks. Specifically, the upper-body follows wrist poses of humans and the relative positions between the robot end-effectors and the robot head are expected to match those between the human wrists and head, then the arm actions are obtained from inverse kinematics. The finger movements are also captured by VR simultaneously to control the dexterous hands for loco-manipulation tasks. Benefiting from the superiority of the training method and the use of domain randomization, our policy does not require additional sim-to-real techniques and can be deployed directly on the NVIDIA Jetson Orin NX onboard the robot for inference. The action output frequency of the policy is 50Hz.

Teleoperation for Loco-Manipulation. Due to the absence of hand keypoints in the motion dataset, we use the VR device to capture human hand poses. This data was then used to real-time control the robot's dexterous hand, enabling it to perform various loco-manipulation tasks. In this setup, the robot's camera provides real-time, first-person 3D visual feedback to the VR device, allowing the operator to see through the robot's perspective. The system translates human wrist poses into the robot's coordinate frame, ensuring that the relative positions between the robot's end-effectors and head mirror those of the human wrists and head. The robot wrist orientations are calibrated to match the absolute orientations of the human wrists, as determined during the initialization of the hand-tracking system. We employ closed-loop inverse kinematics, implemented with Pinocchio [77], to calculate the robot arm's joint angles. The human hand keypoints are translated into robot joint angle commands through dex-retargeting, utilizing a flexible and efficient motion retargeting library [78]. Our method further leverages vector optimizers to enhance the performance of the dexterous hand control. The teleoperation system basically follows Open-Television [50].

Real-robot Experiments on Motion Tracking In the real world, we use the remote controller to control the robot to complete a series of motion tracking tasks. Fig. 8 provides some illustrative examples of our ALMI's motion tracking quality compared between simulation and real world. These results demonstrate that our policy achieves comparable performance in tracking speed commands and upper-body motions in the real world as in the simulator, without relying on any additional sim-to-real techniques.

Real-robot Experiments on Loco-Manipulation By leveraging the ALMI policy and teleoperation deployment, we enable the robot to accomplish a variety of real-world loco-manipulation tasks. Fig. 9 illustrates that the robot can hit the tennis ball with a precise swing. Our robot can also complete household tasks, which requires a combination of stable movement with manipulation capabilities. In Fig. 10(a), the robot can carry a 5kg box and walk stably; while in Fig. 10(b), the robot can use a vacuum to clean the floor.

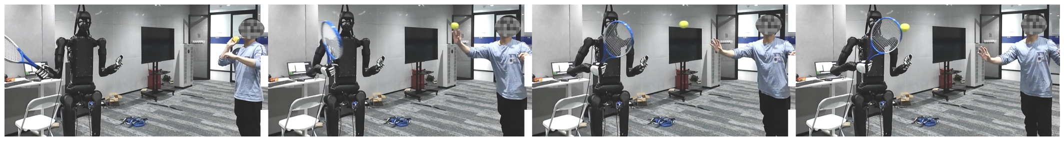


Figure 9: Robot swings the tennis racket with precision.

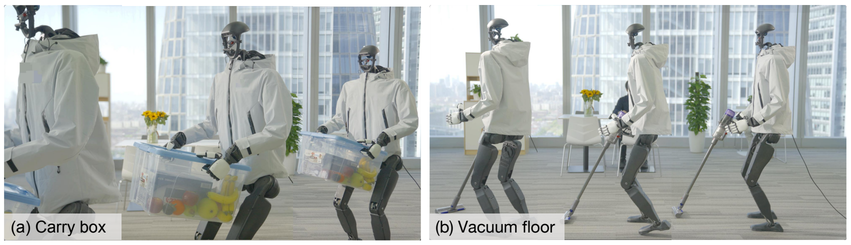


Figure 10: Robot performs loco-manipulation tasks with Teleoperation. (a) Carry a 5kg box. (b) Vaccum floor.

G Broader Social Impact

Our humanoid locomotion method with teleoperation enables robots to perform diverse tasks in human environments, with potential applications in disaster response, healthcare, and industrial automation. However, challenges such as job displacement and safety risks must be addressed to ensure responsible deployment. Ethical guidelines and policy discussions will be crucial to maximize societal benefits while minimizing unintended consequences.