Robot Trains Robot: Automatic Real-World Policy Adaptation and Learning for Humanoids

Kaizhe Hu[*] **Haochen Shi**[*] **Yao He** **Weizhuo Wang** **C. Karen Liu**[†] **Shuran Song**[†]

[*]Equal contribution [†]Equal advising

Stanford University

robot-trains-robot.github.io

Abstract: Simulation-based reinforcement learning (RL) has significantly advanced humanoid locomotion tasks, yet direct real-world RL from scratch or adapting from pretrained policies remains rare, limiting the full potential of humanoid robots. Real-world learning, despite being crucial for overcoming the sim-to-real gap, faces substantial challenges related to safety, reward design, and learning efficiency. To address these limitations, we propose Robot-Trains-Robot (RTR), a novel framework where a robotic arm teacher actively supports and guides a humanoid robot student. The RTR system provides protection, learning schedule, reward, perturbation, failure detection, and automatic resets. It enables efficient long-term real-world humanoid training with minimal human intervention. Furthermore, we propose a novel RL pipeline that facilitates and stabilizes sim-to-real transfer by optimizing a single dynamics-encoded latent variable in the real world. We validate our method through two challenging real-world humanoid tasks: fine-tuning a walking policy for precise speed tracking and learning a humanoid swing-up task from scratch, illustrating the promising capabilities of real-world humanoid learning realized by RTR-style systems.

Keywords: Humanoid Robots, Sim-to-Real Adaptation, Real-World RL

1 Introduction

Recent advances in training reinforcement learning (RL) policies using massive parallel simulation environments have yielded remarkable results in humanoid locomotion tasks [1, 2, 3, 4, 5]. These methods demonstrate the ability to deploy on physical humanoid robots in a zero-shot manner or via real-world adaptation.

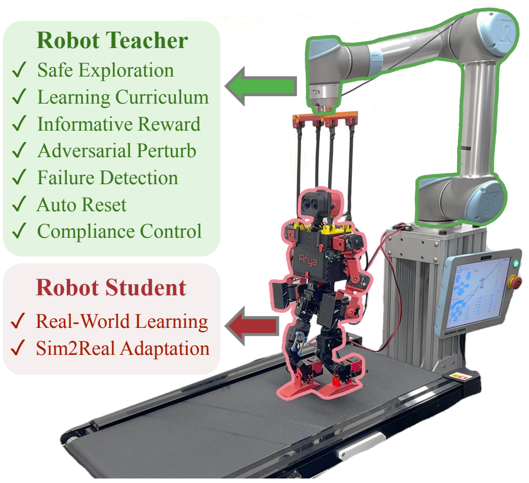


Figure 1: **Robot Trains Robot (RTR).** We propose RTR for automatic real-world policy adaptation and learning with a robot arm as the teacher and a humanoid robot as the student.

Nevertheless, learning directly in the real world remains arguably the most effective way to pursue optimal performance, as it bypasses the inevitable sim-to-real gap [6]. This is especially critical for complex systems like humanoid robots, where discrepancies between simulation and real-world dynamics can significantly hinder policy capability. Despite these benefits, real-world learning on humanoids, either from scratch or fine-tuning, remains highly challenging due to several key factors:

- **Safety**: Real-world RL can produce unexpected and dangerous actions during exploration, easily breaking the fragile balance of the humanoid. Current approaches typically suspend humanoid robots from fixed gantries during real-world rollout, inevitably constraining the range of data collection. Such passive protection mechanisms also influence learning efficiency, as the robot will be frequently dragged by the gantry and fall before the policy stabilizes.

- **Reward Design**: Many useful reward signals easily available in simulation, such as global velocity or external forces, are difficult or impossible to measure directly on physical robots themselves, complicating real-world reward design.

- **Learning Efficiency**: Real-world training requires frequent resets, which incur high human labor costs. Poor sample efficiency of the learning algorithm further compounds these issues, as real-world data collection is inherently costly and constrained.

These considerations highlight the need for a comprehensive and practical system tailored explicitly for real-world humanoid learning. To address these challenges, we propose a novel real-world policy adaptation and learning paradigm: **Robot-Trains-Robot (RTR)**, where a teacher robot (a robot arm with force feedback) trains another student robot (a humanoid). Specifically:

- To address safety concerns, RTR utilizes a robot arm with force-torque sensing to actively support and deliver interactive force feedback to the humanoid robot via compliance control, allowing extensive yet safe exploration across diverse behaviors.

- To gather crucial reward information, RTR uses real-time measurements from the teacher to derive proxy reward signals that would otherwise be difficult to obtain in the real world.

- To improve learning efficiency, RTR uses an automatic curriculum to provide procedural guidance, dynamically adjusting training difficulty, and deliberately perturbing the robot to enhance robustness. The RTR system also enables failure detection, automatic resets, and an asynchronous data collection and policy update pipeline, significantly reducing manual intervention and enabling high-throughput data collection.

Additionally, to further improve learning efficiency, we propose a novel **sim-to-real fine-tuning algorithm**. Our approach consists of a three-stage process: First, we train a dynamics-aware policy in simulation via domain randomization, embedding environment physics information into the policy via a latent encoder and FiLM [7] layers. Next, we optimize a universal latent vector across diverse simulated environments, providing a robust initialization for subsequent real-world training. Finally, leveraging the RTR hardware, we efficiently refine the dynamics latent in the real world using PPO [8], substantially improving the policy performance and robustness in the real world.

We evaluate our system and method through two challenging real-world humanoid tasks: fine-tuning a walking policy for precise speed tracking, and learning a humanoid swing-up behavior from scratch. Given a desired velocity command, RTR effectively doubles the zero-shot walking speed with only 20 minutes of real-world training. In the swing-up task, the humanoid successfully learns to achieve a periodical swing-up motion from scratch within 15 minutes of real-world interaction. While we implement the RTR system with an open-source small-sized humanoid ToddlerBot [9], the proposed paradigm is readily generalizable to full-scale humanoids. Contemporary industrial robotics arms are capable of lifting and interacting with payloads up to 600 kg [10], suggesting the broad applicability of RTR across a wide range of humanoid platforms.

In summary, our contributions include: **(1)** A comprehensive real-world learning system that provides crucial protection, guidance, automation, and informative feedback tailored for real-world humanoid learning. **(2)** An efficient RL paradigm leveraging dynamics-aware latent optimization for rapid and stable real-world policy adaptation. **(3)** Empirical validation of our approach through two challenging tasks highlighting RTR's efficiency and capability to support and generalize across diverse real-world learning scenarios.

2 Related Works

To address the sim-to-real gap [6] and improve policy performance in the real world, there are three mainstream approaches: zero-shot sim-to-real transfer, policy pretraining in simulation followed by real-world adaptation, and direct real-world RL from a random initialization.

Zero-shot Sim-to-real Transfer. Directly deploying a simulation-pretrained policy into the real world is prone to failures due to the sim-to-real gap. Pretraining in simulation with extensive domain randomization has emerged as an effective strategy across both manipulation [11, 6, 12] and locomotion [13, 14, 15, 16, 17] tasks. This approach has notably succeeded even in challenging contexts such as dexterous manipulation involving rich contact interactions [18, 19, 20, 21, 22] and complex humanoid locomotion characterized by inherently unstable dynamics [23, 24, 25, 3, 5]. However, inappropriate domain randomization may produce policies that either fail to adapt effectively to real conditions or become overly conservative, thus limiting performance.

Real-world Adaptation. A popular approach to bridging the sim-to-real gap is to leverage real-world data. These methods typically rely on a robust pretrained policy to safely generate on-policy data, facilitating targeted exploration within task-specific distributions. Some adaptation methods use real-world data to adjust simulation parameter distributions, aligning simulated policy behaviors more closely with real-world experiences [26, 27, 28]. Other approaches employ online adaptation techniques to fine-tune latent representations [29, 30, 31, 32], a residual policy [33, 1], or the original policy directly [34, 35, 36]. Additionally, some studies incorporate online human corrections for policy refinement [37, 38]. Notably, due to humanoids' inherent instability, it is challenging to maintain safety during real-world adaptation. Consequently, fewer studies focus on humanoids [1, 32], highlighting the need for a specialized system to support safe and effective humanoid adaptation.

Our proposed dynamics latent tuning pipeline is the most similar to context-based meta-RL methods [39, 40], where a latent vector is extracted from past observations or physical parameters and used to pivot the policy to adapt to different tasks or environments during meta-pretraining. The latent could then be optimized in the real world for rapid policy adaptation. While these methods have shown success in quadruped locomotion tasks [41, 42], they implicitly rely on the inherent stability of quadrupeds to safely initiate real-world interaction. For humanoids, any initial performance gap risks falls or hardware damage. Although early attempts have deployed such methods on humanoids [43], they rely on a simple platform and a scripted policy for initial data collection, which is impractical for more complex humanoid systems. In contrast to previous works, RTR combines a carefully designed hardware setup with a simulation-optimized initial latent, enabling the real-world application of this category of algorithms on humanoid robots.

Real-world RL. For tasks that are difficult to simulate or have a large sim-to-real gap, directly training in the real world is often more desirable. Some studies have successfully adopted this approach across various setups, including tabletop manipulation [44], dexterous manipulation [45], mobile manipulation [46], and quadruped locomotion [47, 48, 49, 50]. However, few studies have explored direct real-world RL with humanoids due to several unresolved challenges: (1) maintaining humanoid stability and preventing falls during early stages of training, (2) ensuring safe and efficient reset mechanisms, and (3) providing sufficient feedback and informative reward signals. One notable work of this kind by Bloesch et al. [51] use a much simpler humanoid platform [52] with 20 DoFs to move forward via end-to-end reinforcement learning. We propose RTR to address these challenges and enable real-world reinforcement learning on a much more complex humanoid platform.

3 Method

3.1 Teacher-Student Hardware System

Teacher Setup. We use a 6-DoF UR5 arm to support the humanoid robot. An ATI mini45 force-torque (F/T) sensor is mounted on the arm's end-effector to measure interaction forces. Four elastic ropes connect the arm's end effector to the humanoid's shoulders. The elasticity of the rope is cru-

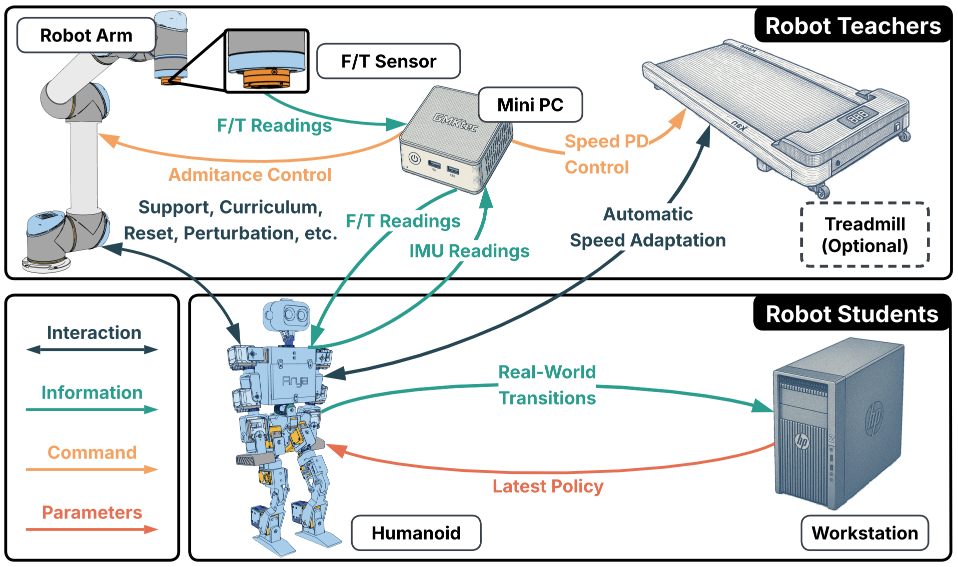


Figure 2: **System Setup.** We illustrate the system architecture and component interactions. The system consists of two groups: robot teachers and robot students. The teachers include a robot arm with an F/T sensor, a mini PC, and an optional treadmill for locomotion tasks; the students include a humanoid robot and a workstation for policy training. The four types of lines represent physical interaction, data transmission, control commands, and neural network parameters, respectively.

cial, as it enables smoother force transmission and avoids abrupt force changes commonly seen in rigid connections or non-elastic ropes. For walking experiments, we additionally provide a programmable treadmill to ensure the robot stays within the reach of the robot arm. This treadmill is equipped with a position encoder and a microcontroller to provide closed-loop control of the moving speed. As shown in Figure 2, a mini PC connects to the arm, F/T sensor, and treadmill via Ethernet cables, while communicating with the student robot via WiFi. This mini PC bears several functionalities: (1) sending control signals to the arm and the treadmill to ensure they assist the learning of the humanoid properly, and (2) collecting data from the F/T sensor and the treadmill to tailor the curriculum, reset timing, and gather reward information for the robot.

Student Setup. We use the open-source humanoid ToddlerBot [9] for its compact size (0.56 m, 3.4 kg), dexterity (30 degrees of freedom), availability (costs less than $6,000$ USD), and robustness. The UR5 arm, with a 5 kg payload, provides adequate capacity to support the robot. As safety is a major concern in real-world learning, verifying our algorithm on a lightweight yet versatile platform like ToddlerBot enables unattended operation without risk of damaging itself or the surrounding environment. Moreover, hardware reliability for the humanoid is equally important—ToddlerBot's motors are sufficiently resistant to overheating and capable of continuous operation over extended periods (> 1 hour), making the robot well-suited for our real-world learning tasks.

3.2 Real-world Adaptation

Real-world adaptation, which fine-tunes a simulation-pretrained policy in the real world, is critical for improving performance on demanding locomotion tasks. In this section, we apply our real-world adaptation pipeline to fine-tune a walking policy for precise speed tracking, while our setup is suitable for adapting a range of locomotion tasks like running and whole-body trajectory following.

On the teachers' side, we introduce an automatic curriculum that dynamically adjusts the supporting force and enables rapid horizontal compliant arm following. On the students' side, we propose a general-purpose sim-to-real adaptation algorithm based on domain randomization and dynamics latent optimization. We first present our three-stage student learning algorithm, as illustrated in Figure 3, and then describe the teacher's policy during real-world adaptation.

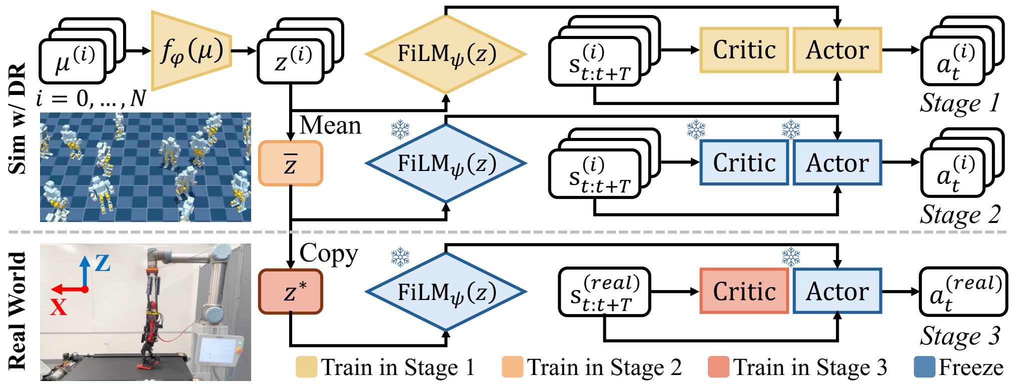


Figure 3: **Sim-to-real Fine-tuning Algorithm.** We illustrate our sim-to-real finetuning process. First, we train a dynamics-aware policy in simulation via domain randomization (DR), encoding environment physics into a latent vector. Next, we optimize a universal latent across diverse simulation environments to initialize real-world training. Finally, we refine the latent and train a new critic in the real world. Orange denotes trainable components in three stages; blue indicates frozen ones.

Dynamics-conditioned Policy Training. In the first stage, we train a dynamics-aware policy $\pi(s, z)$ that conditions on both the current observation s and a dynamics latent z in $N = 1000$ domain randomized simulation environments (more details in Appendix A). Specifically, for the i-th environment, we encode environment-specific physical parameters $\mu^{(i)}$ into a latent representation $z^{(i)}$ using a multi-layer perceptron (MLP) encoder f_ϕ: $z^{(i)} = f_\phi(\mu^{(i)})$. This dynamics latent is then incorporated into the standard PPO actor network through Feature-wise Linear Modulation (FiLM) layers [7]. Concretely, let $h_j^{(i)}$ denote the hidden latents of the j-th layer within the actor network after activation function, we modulate these latents using FiLM as follows:

$$\gamma_j^{(i)}, \beta_j^{(i)} = \text{FiLM}_j(z^{(i)}), \quad h_j^{(i)} \leftarrow \gamma_j^{(i)} \odot h_j^{(i)} + \beta_j^{(i)}, \tag{1}$$

where $\gamma_j^{(i)}$ and $\beta_j^{(i)}$ are scaling and shifting parameters generated by the FiLM layers from the latent vector $z^{(i)}$, and $\odot$ represents element-wise multiplication. The encoder network f_ϕ and the FiLM-modulated policy network $\pi(s, z)$ are jointly trained with PPO in randomized domains with parameters $\mu^{(i)}$, allowing the policy to leverage latent dynamics information effectively and adapt to various simulation environments. Empirically, we find that the learning rate of the FiLM layers is critical: a rate that is too small causes the policy to ignore the dynamics latent, while a rate that is too large leads to instability. An ablation study of this effect is provided in Appendix B.

Universal Latent Optimization. A practical challenge arises when deploying the dynamics-aware policy in the real world: the initial latent representation z for the real world is unknown, as the environment-specific parameters μ are unavailable. Therefore, we propose the second stage to optimize a universal latent $\tilde{z}$ starting from $\bar{z}$, the average of all the latent vectors of the training environments. Formally, we freeze the policy network and FiLM layer parameters and optimize $\tilde{z}$ using PPO, aiming for robust performance across all domain-randomized simulation environments:

$$\tilde{z} = \arg\max_z \sum_i \mathbb{E}_{\tau \sim \pi(\cdot|z),\, \mathcal{T}_i}[J(\tau)] \tag{2}$$

where $\mathcal{T}_i$ denotes the transition distribution for the i-th environment, J denotes the optimization objective of the PPO algorithm, i.e., the expected cumulative reward under policy π, and the expectation is taken over the joint distribution induced by the policy and the environment dynamics. The resulting latent $\tilde{z}$ provides a robust initial condition suitable for real-world deployment.

Real-world Finetuning. In this stage, we freeze the actor network and FiLM layer parameters and fine-tune the latent in the real world using $\tilde{z}$ as the initial solution. Meanwhile, we train the critic from scratch since some privileged observations are unavailable in the real world, resulting in a different observation space from the simulation. For the walking task, the reward is designed to encourage tracking of the target velocity and is defined as follows:

$$r = \exp\left(-\sigma \cdot (v - v^{\text{target}})^2\right), \tag{3}$$

where v and v^{target} are the current and target robot velocity, respectively, and $\sigma = 100$ is a reward-shaping hyperparameter. Since the humanoid is walking on a treadmill, its torso remains relatively stationary in the global frame - state estimation or motion capture yields a near-zero velocity. Therefore, we approximate v with the speed of the treadmill.

Real-world Teachers. The robot teachers include the robot arm with an F/T sensor, a mini PC, and a treadmill. They together provide guidance, schedule, reward, failure detection, and automatic resets. **(1) Guidance:** Leveraging feedback from the F/T sensor, we employ admittance control [53, 54] for the robot arm. The arm remains compliant along the XY axes to accommodate the relative movement generated by the humanoid (Figure 3). This configuration allows the humanoid to move freely in the XY direction while maintaining an upright posture. **(2) Schedule:** To gradually reduce assistance, we implement a scheduling strategy where the arm's height linearly decreases by 0.02m in 5×10^4 environment steps, diminishing the supporting force to near zero at the end of training. **(3) Reward:** We also implement a PD feedback control loop on the treadmill's velocity, based on force along the X axis by the F/T sensor and the humanoid's torso pitch angle. This feedback loop helps the robot maintain an upright walking posture and ensures that the treadmill speed reflects the humanoid's walking speed. We further use this tracking speed to provide reward signal as in Equation (3). **(4) Failure Detection and Automatic Resets:** Moreover, RTR automatically detects failure if the humanoid's torso pitch exceeds a threshold or the F/T sensor reads a large force along the X or Y axis, prompting the system to step and the arm to lift the humanoid to reset the training.

3.3 Real-world Learning from Scratch

While sim-to-real learning is effective for humanoid locomotion, it struggles with tasks involving hard-to-simulate objects like deformable ones. RTR is also suitable for direct real-world training in such cases. To demonstrate this flexibility, we introduce a challenging real-world RL task—learning a swing-up behavior (Figure 5)—which is difficult to simulate due to complex cable dynamics.

Three-stage Training. Inspired by Lei et al. [35], we use a three-stage training pipeline: (1) train the actor and critic from scratch in the real-world using PPO, and collect $50{,}000$ steps of suboptimal transition data; (2) pre-train the critic using offline RL on these data; and (3) initialize a new actor while loading the pretrained critic, and continue to train both networks jointly. The reward is designed to maximize the amplitude of the dominant periodic force measured by the force sensor during the swinging. Specifically, we applied the Fast Fourier Transform (FFT) on the most recent $1,000$ force readings along the x-axis to obtain the force spectrum in the frequency domain and retrieve the dominant frequency, denoted as ν_x. We then extract the force amplitude at this frequency, $\hat{A}_{\nu_x}$, and define the reward as:

$$r = \exp\left(-\alpha \cdot (\hat{A}_{\nu_x} - A^{\text{target}})^2\right), \tag{4}$$

where $\alpha = 0.005$ and $A^{\text{target}} \approx mg\theta_0$ is derived under the small-angle approximation of a pendulum swing. $m = 3.5$ kg is the mass and $\theta_0 = 30°$ is the maximum angular displacement expected.

Real-world Teachers. The robot teacher guides the humanoid's swing-up motion by either amplifying the swing or damping the swing. Both strategies are phase-aligned with the humanoid's current swing angle θ_t at the dominant force frequency ν_x. **(1) Guidance:** To amplify the swing, i.e., increase the force amplitude at the dominant frequency, the arm is given a position target $x_t = x_0 + A_{\text{arm}} \cos(\theta_t)$. **(2) Perturbation:** To dampen the swing, i.e., decrease the force amplitude, the arm is given a phase-inverted position target $x_t = x_0 - A_{\text{arm}} \cos(\theta_t)$, where $A_{\text{arm}} = 0.05$ m, and x_0 denotes the initial x-axis displacement of the arm. **(3) Schedule:** We evenly divide each data batch into several bins and randomly select a certain number of bins to apply either guidance or perturbation. The arm maintains a fixed position in the rest bins. The final training schedule includes a mixture of helping, perturbing, and remaining static periods.

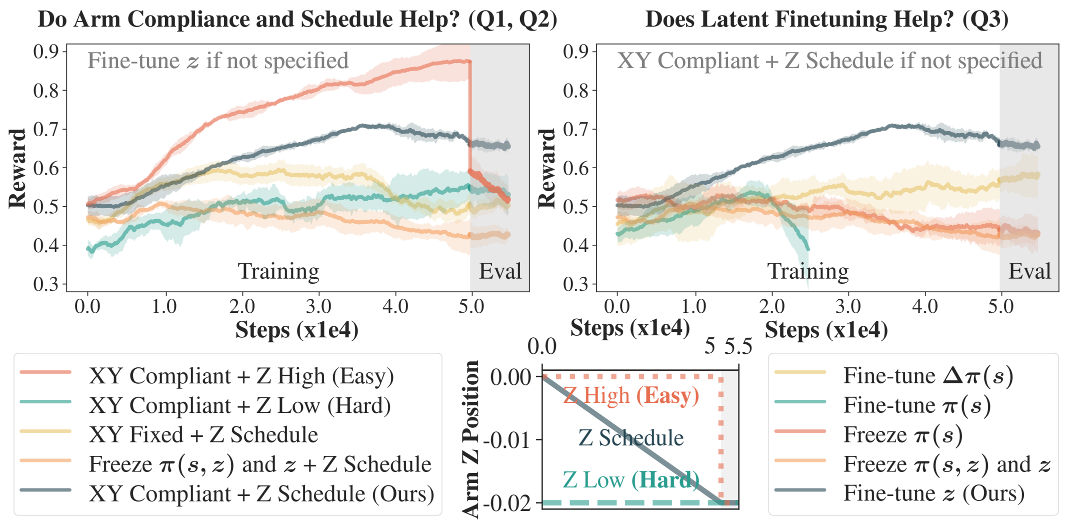


Figure 4: **Walking Ablation.** This experiment aims to evaluate the effectiveness of arm feedback control and latent vector finetuning. We present the linear velocity tracking rewards during training and evaluation, with the arm schedule shown at the bottom center. All variants are tested under the same condition: the arm uses an XY Compliant policy with Z fixed at a Δ position of -0.02 m. We conduct each experiment with three random seeds and show the mean and standard deviation in the plot. Unless otherwise specified, *Finetune z*, *XY Compliant*, and *Z Schedule* are assumed.

Table 1: We evaluate the walking baselines in this table. The humanoid's task is to track the treadmill speed (0.15 m/s). To assess walking stability, torso pitch and roll (radian) are measured by the humanoid's IMU, and end-effector (EE) force (N) by the F/T sensor. Unless otherwise specified, *Finetune z*, *XY Compliant*, and *Z Schedule* are assumed if not otherwise specified.

Method	Torso Roll ↓	Torso Pitch ↓	EE Force X ↓	EE Force Y ↓	EE Force Z ↓
Freeze $\pi(s)$	0.124 ± 0.012	0.102 ± 0.034	1.217 ± 0.134	1.235 ± 0.065	3.431 ± 0.606
Finetune $\Delta\pi(s)$	0.102 ± 0.032	0.096 ± 0.031	1.007 ± 0.025	1.078 ± 0.136	2.316 ± 0.486
Freeze $\pi(s, z)$	0.110 ± 0.021	0.064 ± 0.014	1.672 ± 0.174	1.088 ± 0.186	3.654 ± 0.988
Z Fixed (High)	0.151 ± 0.030	0.126 ± 0.011	1.064 ± 0.143	0.993 ± 0.279	2.402 ± 0.732
Z Fixed (Low)	0.199 ± 0.113	0.112 ± 0.053	1.254 ± 0.133	0.882 ± 0.104	3.237 ± 1.196
XY Fixed	0.170 ± 0.030	0.156 ± 0.027	1.175 ± 0.130	0.864 ± 0.100	2.299 ± 0.820
RTR (ours)	$\mathbf{0.093} \pm \mathbf{0.020}$	$\mathbf{0.053} \pm \mathbf{0.044}$	$\mathbf{0.943} \pm \mathbf{0.202}$	$\mathbf{0.754} \pm \mathbf{0.122}$	$\mathbf{0.954} \pm \mathbf{0.445}$

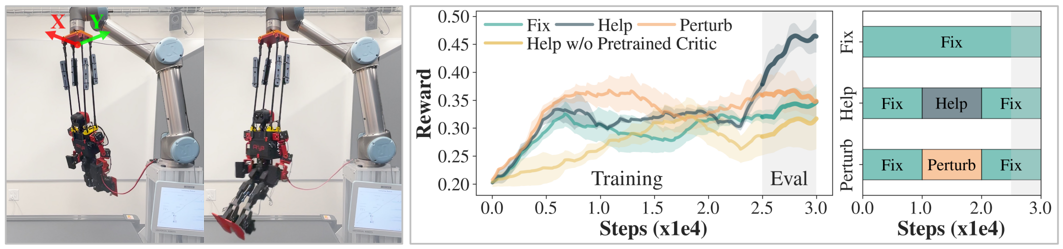
4 Experiments

4.1 Real-world Adaptation with Simulation Pretraining: Walk

Overview. We consider the task of walking on a treadmill while accurately tracking the treadmill's speed to demonstrate RTR 's real-world adaptation capability. During training and evaluation, the reward is defined as the treadmill's speed, with the treadmill speed controlled via feedback from IMU and force readings. During testing, performance is measured by walking stability at a fixed treadmill speed of 0.15 m/s (Table 1). We conduct ablation studies to answer three key questions: **(Q1)** Does arm compliance control help? **(Q2)** Does arm schedule help? **(Q3)** Is fine-tuning z^* more data efficient in real-world adaptation?

Arm Compliance. As shown on the left of Figure 4, we compare RTR with the baseline *XY Fixed + Z Schedule*, where the arm remains stationary and cannot adapt to the humanoid's XY movement during walking. We observe that the arm often drags the humanoid back and impairs policy adaptation. During evaluation, compliance control is re-enabled to ensure a fair comparison.

Arm Schedule. We compare our linearly decreasing arm height schedule with two alternative scheduling strategies: *XY Compliant + Z High (Easy)* and *XY Compliant + Z Low (Hard)*. The different arm schedules are illustrated in the lower center of Figure 4. *Z High (Easy)* trains the



(a) Swing-up Task (b) Swing-up Ablations (c) Arm Schedules

Figure 5: **Swing-up Ablation.** We illustrate the swing-up setup and experiment results. (a) The humanoid is suspended from a robot arm and uses its legs to build momentum and maximize rope angle. (b) We compare helping and perturbing arm schedules against a fixed-arm baseline and also evaluate helping without a pretrained critic. Each experiment is run with three random seeds, and plots show the mean and standard deviation. (c) We show three arm schedules, where helping and perturbing occur during the middle phase, with the arm fixed at the beginning and end.

humanoid at a relatively high delta arm height (0 m) and evaluates it at a low delta arm height (-0.02 m). As shown on the left of Figure 4, the training curve goes up very quickly during training because the robot takes advantage of the arm support, and then it collapses during evaluation when the arm height is lower due to the large training and evaluation gap. For *Z Low (Hard)*, the training and evaluation arm heights are the same, so the policy should ideally overfit to the arm height. However, due to the high initial task difficulty, the policy frequently falls in the early stage, leading to poor data quality for finetuning and slower reward improvement compared to our method.

Fine-tuning Latent. We compare RTR with two fine-tuning strategies: fine-tuning $\pi(s)$ and fine-tuning $\Delta\pi(s)$ (Figure 4). Directly fine-tuning $\pi(s)$ initially progresses well but quickly collapses. In fine-tuning $\Delta\pi(s)$, the base policy $\pi(s)$ is frozen, and a residual policy is initialized and trained from $\pi(s)$ with the last layer weights reset to zero. We find that RTR achieves better data efficiency than both baselines. All methods use the same arm policy, *XY Compliant + Z Schedule*. For reference, we also show that without fine-tuning $\pi(s)$ or $\pi(s, z)$, the reward gradually decreases as the arm lowers and the task difficulty increases. We additionally compare RTR with RMA [30] in Appendix D.

4.2 Real-world Learning from Scratch: Swing-up

Overview. To demonstrate the effectiveness of RTR for real-world learning from scratch, we consider training a swing-up behavior with RL in the real world. The objective is to achieve the maximum swing height within a 20-second time window. To simplify the task, we constrain the action space to hip pitch, knee pitch, and ankle pitch and enforce symmetry between the left and right legs. We perform ablation experiments to answer two key questions: **(Q1)** Does active arm involvement help in the training schedule? **(Q2)** Does critic pretraining help?

Analysis. As shown in Figure 5, both the helping and perturbing arms outperform the fixed arm, with the helping achieving the best performance. We conclude that helping allows the critic to learn what good states look like during the helping interval (1e4 – 2e4 steps), enabling the policy to quickly reach those states afterward. Pretraining the value function with offline data also accelerates early-stage learning - the model without a pretrained critic exhibits the slowest improvement.

5 Conclusion

We present a comprehensive real-world learning system, RTR, which enables protection, guidance, schedule, reward, perturbation, failure detection, and automatic resets for humanoid learning. We introduce an efficient RL paradigm based on dynamics-aware latent optimization, enabling rapid and stable policy adaptation in the real world. Looking ahead, we aim to extend RTR to larger humanoid robots, handle increasingly complex tasks, and further improve data efficiency, allowing more effective collection and utilization of real-world data for humanoid learning.

6 Limitation

Although RTR can autonomously execute the training curriculum in the real world, the curriculum itself remains task-specific and requires manual design and tuning. Future work should explore more generalizable approaches to real-world curriculum generation. Additionally, our reward design is constrained by the availability of hardware and sensors—for example, RTR lacks access to ground reaction force measurements, which are readily available in simulation but absent in our real-world setup, while instrumenting a force plate beneath the treadmill is a potential workaround. Developing comprehensive real-world sensing methodologies represents a promising direction for advancing continuous robot learning in real-world environments.

One promising future direction of RTR is to extend to full-scale humanoid robots. Although the current setup cannot handle such robots due to payload constraints, the principle of RTR–a dynamic, force-aware teacher-student system that provides crucial learning curriculum, safety guarantee, and auto resets during real-world learning–is generally applicable. For larger humanoid robots, the robot teacher can be an industrial robot arm or a bridge crane with force sensing, and the latter would also eliminate the need for a treadmill since the crane can move in a large horizontal area.

Acknowledgment

The authors would like to express their great gratitude to Yifan Hou for providing valuable help and feedback on the robot arm compliance controller. We thank Sirui Chen, Pei Xu, Kun Lei, Albert Wu, Ruocheng Wang, and Ziang Cao for their input on humanoid reinforcement learning. Finally, we appreciate the helpful discussions from all members of TML and REALab.

This work was supported in part by the NSF Award #2143601, #2037101, #2132519, #2153854, Sloan Fellowship, and Stanford Institute for Human-Centered AI. We would like to thank Google for the UR5 robot hardware. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

References

[1] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan, Z. Yi, G. Qu, K. Kitani, J. Hodgins, L. J. Fan, Y. Zhu, C. Liu, and G. Shi. ASAP: Aligning Simulation and Real-World Physics for Learning Agile Humanoid Whole-Body Skills, Feb. 2025.

[2] T. He, W. Xiao, T. Lin, Z. Luo, Z. Xu, Z. Jiang, J. Kautz, C. Liu, G. Shi, X. Wang, L. Fan, and Y. Zhu. HOVER: Versatile Neural Whole-Body Controller for Humanoid Robots, Mar. 2025.

[3] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, Apr. 2024. doi:10.1126/scirobotics.adi9579.

[4] I. Radosavovic, S. Kamat, T. Darrell, and J. Malik. Learning Humanoid Locomotion over Challenging Terrain, Oct. 2024.

[5] Z. Zhuang, S. Yao, and H. Zhao. Humanoid Parkour Learning. In *8th Annual Conference on Robot Learning*, Sept. 2024.

[6] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, Sept. 2017. doi:10.1109/IROS.2017.8202133.

[7] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer, 2017. URL https://arxiv.org/abs/1709.07871.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[9] H. Shi, W. Wang, S. Song, and C. K. Liu. ToddlerBot: Open-Source ML-Compatible Humanoid Platform for Loco-Manipulation, Feb. 2025.

[10] Research Nester. Industrial robotic arm market size, share, growth trends, regional share, competitive intelligence, forecast report 2025–2037, March 2025. URL https://www.researchnester.com/reports/industrial-robotic-arm-market/6763. Accessed: 2025-04-28.

[11] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, May 2018. doi:10.1109/ICRA.2018.8460528.

[12] Z. Yuan, T. Wei, S. Cheng, G. Zhang, Y. Chen, and H. Xu. Learning to Manipulate Anywhere: A Visual Generalizable Framework For Reinforcement Learning, Oct. 2024.

[13] H. Ha, Y. Gao, Z. Fu, J. Tan, and S. Song. UMI-on-Legs: Making Manipulation Policies Mobile with Manipulation-Centric Whole-body Controllers. In *8th Annual Conference on Robot Learning*, Sept. 2024.

[14] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning. In *Proceedings of the 5th Conference on Robot Learning*, pages 91–100. PMLR, Jan. 2022.

[15] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme Parkour with Legged Robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450, May 2024. doi:10.1109/ICRA57147.2024.10610200.

[16] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. In *Robotics: Science and Systems XIV*, volume 14, June 2018. ISBN 978-0-9923747-4-7.

[17] F. Shi, Y. Kojio, T. Makabe, T. Anzai, K. Kojima, K. Okada, and M. Inaba. Reference-free learning bipedal motor skills via assistive force curricula. In A. Billard, T. Asfour, and O. Khatib, editors, *Robotics Research*, pages 304–320, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-25555-7.

[18] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving Rubik's Cube with a Robot Hand, Oct. 2019.

[19] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, Nov. 2023. doi:10.1126/scirobotics.adc9244.

[20] Y. Chen, C. Wang, L. Fei-Fei, and K. Liu. Sequential Dexterity: Chaining Dexterous Policies for Long-Horizon Manipulation. In *7th Annual Conference on Robot Learning*, Aug. 2023.

[21] Y. Chen, C. Wang, Y. Yang, and K. Liu. Object-Centric Dexterous Manipulation from Human Motion Data. In *8th Annual Conference on Robot Learning*, Sept. 2024.

[22] T. Lin, K. Sachdev, L. Fan, J. Malik, and Y. Zhu. Sim-to-Real Reinforcement Learning for Vision-Based Dexterous Manipulation on Humanoids, Feb. 2025.

[23] Z. Fu, Q. Zhao, Q. Wu, G. Wetzstein, and C. Finn. HumanPlus: Humanoid Shadowing and Imitation from Humans. In *8th Annual Conference on Robot Learning*, Sept. 2024.

[24] X. Gu, Y.-J. Wang, X. Zhu, C. Shi, Y. Guo, Y. Liu, and J. Chen. Advancing Humanoid Locomotion: Mastering Challenging Terrains with Denoising World Model Learning. In *Robotics: Science and Systems XX*. Robotics: Science and Systems Foundation, July 2024. ISBN 9798990284807. doi:10.15607/RSS.2024.XX.058.

[25] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humplik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, M. Bloesch, K. Hartikainen, A. Byravan, L. Hasenclever, Y. Tassa, F. Sadeghi, N. Batchelor, F. Casarini, S. Saliceti, C. Game, N. Sreendra, K. Patel, M. Gwira, A. Huber, N. Hurley, F. Nori, R. Hadsell, and N. Heess. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89): eadi8022, Apr. 2024. doi:10.1126/scirobotics.adi8022.

[26] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979, May 2019. doi:10.1109/ICRA.2019.8793789.

[27] A. Z. Ren, H. Dai, B. Burchfiel, and A. Majumdar. AdaptSim: Task-Driven Simulation Adaptation for Sim-to-Real Transfer. In *Proceedings of The 7th Conference on Robot Learning*, pages 3434–3452. PMLR, Dec. 2023.

[28] P. Huang, X. Zhang, Z. Cao, S. Liu, M. Xu, W. Ding, J. Francis, B. Chen, and D. Zhao. What Went Wrong? Closing the Sim-to-Real Gap via Differentiable Causal Discovery. In *Proceedings of The 7th Conference on Robot Learning*, pages 734–760. PMLR, Dec. 2023.

[29] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow. Meta-Reinforcement Learning for Robotic Industrial Insertion Tasks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9728–9735, Oct. 2020. doi:10.1109/IROS45743.2020.9340848.

[30] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid Motor Adaptation for Legged Robots, July 2021.

[31] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-Hand Object Rotation via Rapid Motor Adaptation. In *6th Annual Conference on Robot Learning*, Aug. 2022.

[32] A. Kumar, Z. Li, J. Zeng, D. Pathak, K. Sreenath, and J. Malik. Adapting Rapid Motor Adaptation for Bipedal Robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1161–1168, Oct. 2022. doi:10.1109/IROS47612.2022.9981091.

[33] Y. Sun, W. L. Ubellacker, W.-L. Ma, X. Zhang, C. Wang, N. V. Csomay-Shanklin, M. Tomizuka, K. Sreenath, and A. D. Ames. Online Learning of Unknown Dynamics for Model-Based Controllers in Legged Locomotion. *IEEE Robotics and Automation Letters*, 6 (4):8442–8449, Oct. 2021. ISSN 2377-3766, 2377-3774. doi:10.1109/LRA.2021.3108510.

[34] Y. Zhang, L. Ke, A. Deshpande, A. Gupta, and S. Srinivasa. Cherry-Picking with Reinforcement Learning. In *Robotics: Science and Systems XIX*, volume 19, July 2023. ISBN 978-0-9923747-9-2.

[35] K. Lei, Z. He, C. Lu, K. Hu, Y. Gao, and H. Xu. Uni-O4: Unifying Online and Offline Deep Reinforcement Learning with Multi-Step On-Policy Optimization. In *The Twelfth International Conference on Learning Representations*, Oct. 2023.

[36] H. Xiong, R. Mendonca, K. Shaw, and D. Pathak. Adaptive Mobile Manipulation for Articulated Objects In the Open World, Jan. 2024.

[37] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei. TRANSIC: Sim-to-Real Policy Transfer by Learning from Online Correction. In *8th Annual Conference on Robot Learning*, Sept. 2024.

[38] J. Luo, C. Xu, J. Wu, and S. Levine. Precise and Dexterous Robotic Manipulation via Human-in-the-Loop Reinforcement Learning, Mar. 2025.

[39] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine. Meta-reinforcement learning of structured exploration strategies, 2018. URL https://arxiv.org/abs/1802.07245.

[40] K. Rakelly, A. Zhou, D. Quillen, C. Finn, and S. Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables, 2019. URL https://arxiv.org/abs/1903.08254.

[41] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha. Learning fast adaptation with meta strategy optimization, 2020. URL https://arxiv.org/abs/1909.12995.

[42] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals, 2020. URL https://arxiv.org/abs/2004.00784.

[43] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu. Sim-to-real transfer for biped locomotion, 2019. URL https://arxiv.org/abs/1903.01390.

[44] S. Huang, Z. Zhang, T. Liang, Y. Xu, Z. Kou, C. Lu, G. Xu, Z. Xue, and H. Xu. MENTOR: Mixture-of-Experts Network with Task-Oriented Perturbation for Visual Reinforcement Learning, Oct. 2024.

[45] K. Xu, Z. Hu, R. Doshi, A. Rovinsky, V. Kumar, A. Gupta, and S. Levine. Dexterous Manipulation from Images: Autonomous Real-World RL via Substep Guidance. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5938–5945, May 2023. doi:10.1109/ICRA48891.2023.10161493.

[46] R. Mendonca, E. Panov, B. Bucher, J. Wang, and D. Pathak. Continuously Improving Mobile Manipulation with Autonomous Real-World RL. In *Proceedings of The 8th Conference on Robot Learning*, pages 5204–5219. PMLR, Jan. 2025.

[47] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan. Learning to Walk in the Real World with Minimal Human Effort. In *Proceedings of the 2020 Conference on Robot Learning*, pages 1110–1120. PMLR, Oct. 2021.

[48] L. Smith, I. Kostrikov, and S. Levine. A Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning, Aug. 2022.

[49] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. DayDreamer: World Models for Physical Robot Learning. In *Proceedings of The 6th Conference on Robot Learning*, pages 2226–2240. PMLR, Mar. 2023.

[50] L. Smith, Y. Cao, and S. Levine. Grow your limits: Continuous improvement with real-world rl for robotic locomotion, 2023. URL https://arxiv.org/abs/2310.17634.

[51] M. Bloesch, J. Humplik, V. Patraucean, R. Hafner, T. Haarnoja, A. Byravan, N. Y. Siegel, S. Tunyasuvunakool, F. Casarini, N. Batchelor, F. Romano, S. Saliceti, M. Riedmiller, S. M. A. Eslami, and N. Heess. Towards real robot learning in the wild: A case study in bipedal locomotion. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1502–1511. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/bloesch22a.html.

[52] ROBOTIS. ROBOTIS OP3 e-Manual. https://emanual.robotis.com/docs/en/platform/op3/introduction/, 2024. Accessed: 2025-07-29.

[53] J. Maples and J. Becker. Experiments in force control of robotic manipulators. In *1986 IEEE International Conference on Robotics and Automation Proceedings*, volume 3, pages 695–702, Apr. 1986. doi:10.1109/ROBOT.1986.1087590.

[54] Y. Hou, Z. Liu, C. Chi, E. Cousineau, N. Kuppuswamy, S. Feng, B. Burchfiel, and S. Song. Adaptive Compliance Policy: Learning Approximate Compliance for Diffusion Guided Control, Mar. 2025.

[55] Y. Liang, T. Xu, K. Hu, G. Jiang, F. Huang, and H. Xu. Make-An-Agent: A Generalizable Policy Network Generator with Behavior-Prompted Diffusion. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, Nov. 2024.

A State and Action Representation

A.1 Walking

Following the RL training setup in Shi et al. [9], both the base policy $\pi(\mathbf{s}_t)$ and the dynamics-aware policy $\pi(\boldsymbol{s}_t, z)$ output $\mathbf{a}_t$ as joint position setpoints for proportional-derivative (PD) controllers based on the observable state $\mathbf{s}_t$:

$$\mathbf{s}_t = (\boldsymbol{\phi}_t, \mathbf{c}_t, \Delta\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{a}_{t-1}, \boldsymbol{\omega}_t, \boldsymbol{\theta}_t)\,, \tag{5}$$

where $\boldsymbol{\phi}_t$ is a phase signal, $\boldsymbol{c}_t$ represents velocity commands, $\Delta\boldsymbol{q}_t$ denotes the position offset relative to the neutral pose $\boldsymbol{q}_0$, $\boldsymbol{a}_{t-1}$ is the action from the previous time step, $\boldsymbol{\omega}_t$ represents the torso's angular velocity, and $\boldsymbol{\theta}_t$ is the torso orientation in euler angles.

In simulation, the critic observation space includes more privileged information to provide better accuracy for the value function, we have:

$$\mathbf{s}_t^+ = \Big(\boldsymbol{\phi}_t, \mathbf{c}_t, \Delta\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{a}_{t-1}, \mathbf{e}_{\mathbf{q}_t}, \mathbf{v}_t, \boldsymbol{\omega}_t, \boldsymbol{\theta}_t, \mathbf{m}_t, \mathbf{ref}_t, \tilde{\mathbf{v}}_t, \tilde{\boldsymbol{\theta}}_t\Big)\,, \tag{6}$$

where $\mathbf{e}_{\mathbf{q}_t}$ is the error of the current motor position to a reference motor position, $\mathbf{v}_t$ is the linear velocity of the robot in the world frame, $\mathbf{ref}_t$ is the reference motor pose, while $\tilde{\mathbf{v}}_t$ and $\tilde{\boldsymbol{\theta}}_t$ are the linear and angular velocity of the random perturbation applied to the robot.

In the real world, as some of the privileged observations in the simulation are hard to acquire, we leverage the RTR system's force information, and augment the real-world privileged observation as:

$$\mathbf{s}_t^{r+} = (\boldsymbol{\phi}_t, \mathbf{c}_t, \Delta\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{a}_{t-1}, \mathbf{q}_t, \mathbf{v}_t, \boldsymbol{\omega}_t, \boldsymbol{\theta}_t, \mathbf{F}_t, \boldsymbol{\tau}_t)\,, \tag{7}$$

where $\mathbf{q}_t$ is the joint position of the humanoid, $\mathbf{v}_t$ is the linear velocity of the torso measured by a motion tracking system, and $\mathbf{F}_t$ and $\boldsymbol{\tau}_t$ are the force and torque measured by the force-torque sensor mounted on the robot arm teacher.

A.2 Swing-up

For the swing-up task, the observation space is defined the same as Equation 5. We also include more information to form the task's privileged observation space:

$$\mathbf{s}_t^+ = (\boldsymbol{\phi}_t, \mathbf{c}_t, \Delta\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{a}_{t-1}, \boldsymbol{\omega}_t, \boldsymbol{\theta}_t, \boldsymbol{F}_t, \boldsymbol{\tau}_t, \boldsymbol{x}_t) \tag{8}$$

where $\boldsymbol{F}_t, \boldsymbol{\tau}_t, \boldsymbol{x}_t$ are the force reading, torque reading, and arm end effector position, respectively.

B FiLM Ablation

In this section, we discuss details regarding the FiLM layers in our sim-to-real adaptation pipeline. As shown in Equation (1), the FiLM layer introduces a scaling and shifting effect to each layer of the hidden network, altering the behavior of the policy network inherently.

One key consideration for the FiLM layer during training is its learning rate, as it decides how fast the FiLM layer will change compared to the policy network. In our implementation, we train the FiLM layers along with the policy network from scratch, as we find that initializing the policy network from a pretrained model hinders the effect of the FiLM layer over the policy. Recall that:

$$\gamma_j^{(i)}, \beta_j^{(i)} = \text{FiLM}_j(z^{(i)}), \quad h_j^{(i)} \leftarrow \gamma_j^{(i)} \odot h_j^{(i)} + \beta_j^{(i)}$$

We initialize all the FiLM layers with all-zero weights and all-zero or all-one bias, such that at the start of the training, we have: $\gamma_j^{(i)} = 1.0, \beta_j^{(i)} = 0.0$, which means the FiLM layers do not affect the network output. As the FiLM layers are trained simultaneously with the policy, they are in a

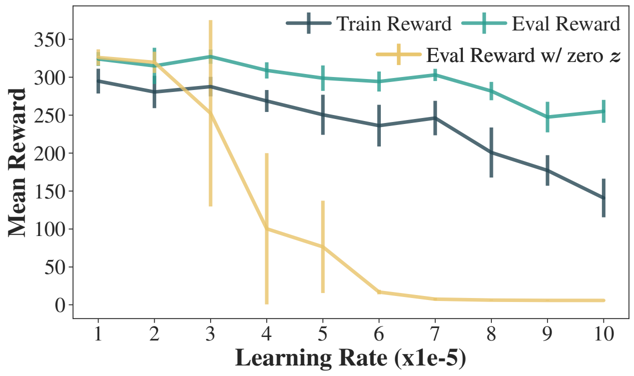


Figure 6: **FiLM learning rate ablation.** This experiment aims to evaluate the effect of FiLM layer learning rates. All experiments are run under seven seeds. Vertical bars indicate standard deviation.

Table 2: We randomize the following environment parameters for the walking task.

Parameter Names	Friction	Damping	Armature	Friction Loss	Body Mass	EE Mass
Randomize Range	$[0.5, 2.0]$	$[0.5, 2.0]$	$[0.5, 2.0]$	$[0.5, 2.0]$	$[-0.3, 0.3]$	$[0.0, 0.1]$

competing condition: the policy network is learning to gain better overall performance given the current FiLM distribution, while the FiLM layers are trained to make use of the environment's latent and maximize the performance in each of the different environments.

We find that a small learning rate for the FiLM layers causes the policy to behave similarly to training without the modulation, limiting its ability to generalize across latent conditions and adapt to different environments. On the other hand, if the learning rate is too large, it will lead to unstable training and lower performance.

We propose a metric to measure the effectiveness for the dynamics latent conditioned $\pi(s, z)$ to utilize the environment information embedded in the latent z. Specifically, we compare the performance of the policy given the true latent z computed from the dynamics encoder $z^{(i)} = f_\phi(\mu^{(i)})$ and give an all-zero latent. We find that the performance of an all-zero latent could be even comparable to that given the real z when the FiLM learning rate is small, but it will gradually decrease to near zero when the learning rate increases and the policy starts to learn to use the latent information. On the other hand, the overall performance of the policy will decrease nearly monotonously when the FiLM learning rate increases. We plot the performance curves at different learning rates in Figure 6. For each learning rate, we run seven trials and plot the mean and standard deviation of the performance. We find that the learning rate of $5e-5$ is a good trade-off between performance and the ability to utilize latent information, while $1e-5$ is too small, leading to nearly identical zero latent and true latent performance, and $1e-4$ is too large and leads to poor performance.

C Experiment Details

C.1 Domain Randomization

Following the domain randomization settings in [9], we slightly increase the domain randomization range to encourage the policy network to better use the dynamics information from the latent. We list the randomized parameters and their range in Table 2. We use the encoder-decoder architecture from [55] to encode the physics parameters to a 1024-dimensional latent before sending it to the FiLM layers, though we only use the encoder in our implementation.

C.2 Arm Control Policy

We implement our appliance arm controller based on the open-sourced code of Hou et al. [54].

For the walking task, we enable appliance control along the XY axis while setting the Z-height directly. We use a stiffness of $[100, 50]$ along the two axis, while setting the damping to $[0.5, 0.5]$ and inertia to $[0.03, 0.03]$, making the arm slightly more compliant on the Y-axis.

For the swing task, we disable the appliance control for the arm since it makes the phase tracking lag behind. We use position control to let the arm follow the helping or perturbing movement described in Section 3.3. During both helping and perturbing modes, each real-world data collection period is evenly divided into 5 bins. In helping mode, 3 bins are randomly selected from the last 4 to apply assistance. In perturbing mode, 1 bin is randomly selected from the first 4 to apply the perturbation.

C.3 Treadmill Control Policy

During the walking task, the robot is walking on the treadmill while the treadmill adjusts its speed dynamically to keep the robot in a good position. We use a PD controller for the treadmill speed v:

$$v = v_{base} + k_p^1 F_x + k_p^2 \psi \tag{9}$$

where $v_{base} = 0.1\ \mathrm{m/s}$ is the default treadmill speed, F_x is the force reading along the x-axis and ψ is the robot's torso pitch. $k_p^1 = 0.2$ and $k_p^2 = -5$ are the proportional gain. We set a treadmill speed limit of $0.24\ \mathrm{m/s}$ to assure the safety of the humanoid platform.

C.4 Online Learning Hyperparameters

During real-world adaptation for walking, we collect a batch of 1024 steps of data before updating the policy on them over 20 epochs using the PPO algorithm with a clipping ratio of 0.2. Both the actor and critic networks are optimized with a learning rate of 1×10^{-4}. To encourage exploration, we apply an entropy coefficient of 0.005.

For training the swing-up task from scratch, we also collect 1024 samples before starting to update the policy. The actor is optimized with a learning rate of 2×10^{-3} and the critic is optimized with a learning rate of 2×10^{-5}. To encourage exploration, we apply an larger entropy coefficient of 0.04. Each PPO update is performed over 20 epochs with a clipping ratio of 0.2.

C.5 Real-world Learning Details

D Comparison with RMA

Rapad Motor Adaptation (RMA) [30] is a sim-to-real adaptation method that is similar to our approach. The training of RMA consists of two stages: **(1)** A pretraining stage, where the policy is trained in a simulated environment with domain randomization. In this stage, the latent information is encoded from the physics parameters by a projection layer, and then concatenated with the observation before being sent to the policy. **(2)** An adaptation stage, which is to address the problem that the physics parameter of the real world is unknown. During this stage, an adaptation module is trained via supervised learning to reconstruct the latent z from the past observations and actions.

Compared to RMA, our approach bears differences in each stage: for the first stage, our method uses a FiLM layer to modulate the policy network, which is a more flexible and powerful way to utilize the latent information. In contrast, RMA simply concatenates the latent information with the observation. In the second stage, our method does not require the adaptation module to reconstruct the latent z, but instead, we optimize a universal latent z^* that is shared across all the environments. This could serve as a good initialization for further training of the optimal real-world latent z^*_{real}. On the other hand, RMA tries to reconstruct the latent z from the past observations and actions, which is prone to overfitting, especially when facing the largely out-of-distribution real-world dynamics.

In the following section, we set up simulation and real-world experiments to compare our method with RMA. We try to answer the following questions: **(1)** How does the FiLM layer modulation affect the performance compared to RMA's simple concatenation? **(2)** How good is the adaptation module compared to our approach to get a universal latent? For each experiment, we conduct ablation studies and keep all other parameters the same. We then evaluate the fully trained RMA model in the real world to prove that our method is indeed a better choice from each perspective for real-world adaptation.

D.1 RMA Implementation Details

We describe the details we use to implement the RMA algorithm on our humanoid platform. During phase one training, we randomize the simulation environment using the same parameter range as described in Appendix C.1, and the randomized parameters are then concatenated to form a physics-information vector. We drop the unchanged digits during the encoding process.

The RMA algorithm is originally designed for a quadruped robot that has relatively low DoFs. Compared with the original implementation, we use a similar process to train a stage one model that takes in the observation and the dynamics latent and outputs the action. During stage two, we make the following changes to adapt the algorithm to our hardware: (1) We decrease the window length of past observations and actions that used to predict the current dynamics latent from 50 to 15, as the observation of the humanoid platform has higher dimensions, and a too long horizon will cause difficulty in training. This adjustment also aligns with the stack frame length we use during training. (2) We change the 1D convolution layers used to process the past states and actions accordingly, since the context window size is reduced. The new convolution layers now have kernel sizes of (5, 3, 3) and stride steps of (1, 1, 1). Besides these changes, the RMA training reuses the existing real-world learning pipeline, while only optimizing the adaptation module in stage two instead of the dynamics latent as in the implementation of RTR.

D.2 FiLM Latent Modulation

We first compare the effect of FiLM layer modulation used by RTR with the concatenation approach used by the first stage in RMA. To this end, we run three seeds for the RMA to train a dynamics latent conditioned policy in 1024 parallel simulation environments. While the FiLM layer-based policy easily reaches an average evaluation return of higher than 300, the concatenated policy can only reach a return of just over 200. We suspect that this is due to the high dimensionality introduced by concatenation, which will hinder the policy performance. The co-existence of the observation and the dynamics latent in the MLP policy input is also likely to cause the network to ignore the dynamics. The detailed results can be found in the first column of Table 3, where the "Ground Truth" denotes that both methods have access to the dynamics latent directly predicted from the encoder.

D.3 Adaptation Module Performance

During the second stage of RMA, an adaptation module is trained to predict the dynamics latent from past observations. While in RTR, a universal latent is trained across randomized environments for initialization of a real-world learning stage. We want to investigate which method could lead to a better estimation of the real dynamics latent. In addition to the design choice of concatenation or FiLM layer modulation studied in the previous subsection, we conduct ablation studies to fully compare the performance of both methods. The results can be found in Table 3.

In the first row of Table 3, we use the stage one of RMA to train a dynamics conditioned policy via directly concatenating the dynamics latent to the observation, which will lead to inferior performance than the FiLM layers. We then use both universal latent optimization and the adaptation module to predict the dynamics latent. The result shows that both methods did pretty well to recover the latent, resulting in a nearly identical performance to the true latent used in phase one.

In the second row of Table 3, we add the dynamics conditioning to the policy via FiLM layers as in RTR. The performance of the conditioned policy is higher than that of the concatenated latent. We then compare the two methods' ability to estimate the dynamics latent for the FiLM layers. This

Table 3: We compare the performance for each stage of RTR (ours) and RMA [30] in this table

	Ground Truth	Adaptation Module (RMA)	Universal Latent (RTR)
Concatenation (RMA)	210.32 ± 20.75	205.74 ± 15.63	207.32 ± 5.89
FiLM layer (RTR)	316.10 ± 11.30	10.32 ± 2.01	**305.28 ± 5.47**

Table 4: We compare RTR and RMA [30] in this table, extending the results from Table 1.

Method	Torso Roll $\downarrow$	Torso Pitch $\downarrow$	EE Force X $\downarrow$	EE Force Y $\downarrow$	EE Force Z $\downarrow$
RMA	0.167 ± 0.010	0.212 ± 0.006	1.172 ± 0.007	0.799 ± 0.028	2.548 ± 0.216
RTR (ours)	**0.093 ± 0.020**	**0.053 ± 0.044**	**0.943 ± 0.202**	**0.754 ± 0.122**	**0.954 ± 0.445**

time, the RMA-style adaptation module failed to recover a feasible dynamics latent for the FiLM layers, causing the performance to stay at a low position, almost close to that of a random policy. While the prediction error of the adaptation module is decreasing, we suspect this is due to the FiLM layers being more sensitive to the small changes of the dynamics layer, and the adaptation module can't fully close this gap from the past observations and actions.

D.4 Real-World Performance

Finally, we run the RMA model in the real world and test its performance using the same metrics as the main paper. While our implementation of the RMA method successfully adapts to the real world to produce a walking behavior somewhat similar to the simulation, the walking gait of the RMA policy is not that stable, and the humanoid often leans forward and is frequently recovered by the RTR hardware. The metrics in Table 4 show that our method clearly outperforms RMA for real-world adaptation.

In summary, during the first stage of training, the approach of using the FiLM layer in RTR outperforms the concatenation method used by RMA. During the second stage, our approach of universal latent optimization could provide similar initial latent compared to RMA, and is more stable when combined with the FiLM layers. What's more, our approach of universal latent optimization leads to a third real-world tuning stage, and has the ability to further boost the real-world performance with the RTR hardware. The results in both simulation and real-world experiments lead to the conclusion that the dynamics latent optimization pipeline of RTR is better suited for sim-to-real adaptation of humanoids than the approach used by RMA.