Bridging the Sim-to-Real Gap for Athletic Loco-Manipulation

Nolan Fey, Gabriel B. Margolis, Martin Peticco, and Pulkit Agrawal
Improbable AI Lab
Massachusetts Institute of Technology, Cambridge, MA 02139

***Abstract*—Achieving athletic loco-manipulation on robots requires moving beyond traditional tracking rewards—which simply guide the robot along a reference trajectory—to task rewards that drive truly dynamic, goal-oriented behaviors. Commands such as "throw the ball as far as you can" or "lift the weight as quickly as possible" compel the robot to exhibit the agility and power inherent in athletic performance. However, training solely with task rewards introduces two major challenges: these rewards are prone to exploitation (reward hacking), and the exploration process can lack sufficient direction. To address these issues, we propose a two-stage training pipeline. First, we introduce the Unsupervised Actuator Net (UAN), which leverages real-world data to bridge the sim-to-real gap for complex actuation mechanisms without requiring access to torque sensing. UAN mitigates reward hacking by ensuring that the learned behaviors remain robust and transferable. Second, we use a pre-training and fine-tuning strategy that leverages reference trajectories as initial hints to guide exploration. With these innovations, our robot athlete learns to lift, throw, and drag with remarkable fidelity from simulation to reality.**

I. Introduction

General whole-body control comes naturally to animals after years of evolution, yet it remains a long-standing challenge in robotics. Fluid whole-body motion requires balancing multiple competing tasks and constraints that depend on both the robot's morphology and its environment [40]. Recent work [6, 32] demonstrates that sim-to-real reinforcement learning (RL), using methods such as Proximal Policy Optimization (PPO) [37], is a promising paradigm for learning these behaviors by leveraging parallel simulations [36].

For dynamic, goal-oriented loco-manipulation, it is natural to train robots with task rewards—commands like "throw the ball as far as possible" or "lift the weight as quickly as possible" that drive athletic behaviors. However, these task rewards pose two major challenges: (i) they are prone to reward hacking, where the policy exploits imperfections in the simulation, and (ii) the exploration process can lack sufficient guidance. To circumvent these issues, many works on sim-to-real transfer instead train whole-body controllers (WBCs) to track dense reference motions [4, 6, 12, 21, 32]. Dense tracking objectives provide strong regularization by constraining the policy to adhere to a reference trajectory—thereby reducing reward hacking—and they offer a structured path for exploration. However, this strategy relies on defining high-quality reference commands a priori, which in turn demands access to high-quality reference data. For robots with non-human morphologies like legged manipulators, obtaining such data is particularly challenging, and the resulting reference commands may not capture the optimal, athletic strategies that a policy might otherwise discover.

To fully harness the benefits of task rewards, it is crucial to ensure that the simulation faithfully replicates real-world

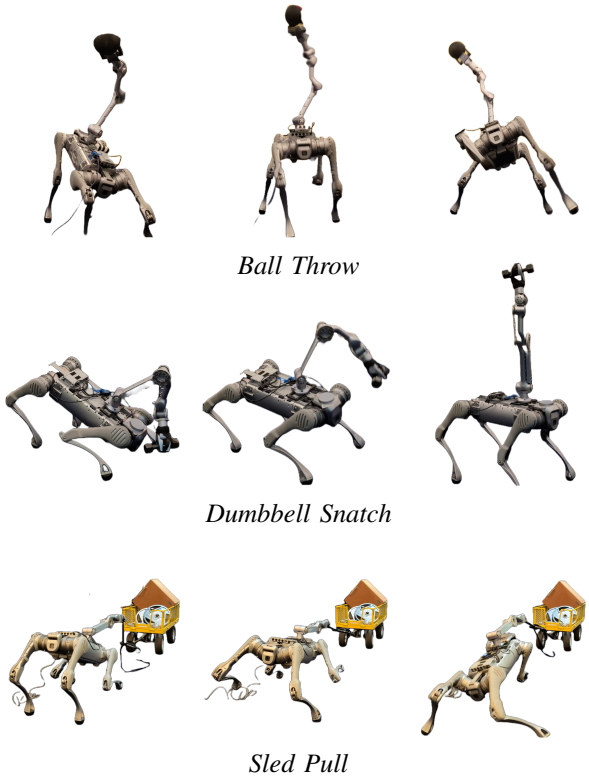


Fig. 1: **Sim-to-real transfer of athletic loco-manipulation.** We reduce the sim-to-real gap for a quadruped manipulator by **learning a corrective model for the simulated actuator dynamics based on real-world data**, formulated as an *unsupervised actuator net (UAN)*. Policies trained with the corrected simulator exhibit improved sim-to-real transfer and push the limits of the robot's physical capabilities in athletic tasks involving whole-body coordination. Videos of the robot's behaviors are available at https://uan.csail.mit.edu/.

[1]Authors are also affiliated with Computer Science and Artificial Laboratory (CSAIL), the Laboratory for Information and Decision Systems (LIDS), and the MIT-IBM Watson AI Lab at MIT. Correspondence to nolanfey@mit.edu

dynamics. Inaccurate simulation models allow policies to exploit imperfections, leading to reward hacking, particularly so when the reward is underspecified. Although techniques like domain randomization [46, 47, 49] and online system identification [14, 17, 22, 27, 29, 33] address this by sampling over parameter distributions, they rely on a priori assumptions that may not fully capture the complex dynamics of real hardware. For instance, harmonic drive actuators exhibit non-linear friction, hysteresis, and lag—behaviors that render traditional proxies like motor current unreliable for torque estimation.

A promising alternative is to enhance the simulation's physics model directly with real-world data, focusing on accurately modeling the actuator dynamics. With this motivation, we introduce the *Unsupervised Actuator Net* (UAN), a framework for learning corrective actuator models without the need for torque sensors. UAN is trained using reinforcement learning to predict corrective torques, $\delta\boldsymbol{\tau} = \pi_{\mathrm{UAN}}(\mathbf{e})$, by minimizing discrepancies between simulated and real-world joint encoder measurements. In doing so, UAN effectively bridges the sim-to-real gap even for robots with complex transmission mechanisms and noisy or unavailable torque measurements.

Building on this enhanced simulation environment, we address the challenge of guided exploration for athletic behaviors. Rather than enforcing strict adherence to a reference trajectory, we propose treating it as a *hint* to guide exploration. In our approach, a WBC is first pre-trained on random base velocities and end-effector pose commands to establish a strong motion prior. Then, to learn a new athletic behavior, we initialize the controller with a reference trajectory and fine-tune it using a task-specific reward—allowing the policy to depart from the reference when beneficial.

In summary, our paper presents an *easy-to-use* training pipeline for whole-body athletic behaviors that reliably transfer to reality. First, we employ the Unsupervised Actuator Net (UAN) to calibrate actuator dynamics and mitigate reward hacking, ensuring our simulator accurately reflects real-world physics. With this improved simulation environment, we then pre-train a whole-body controller (WBC) to establish fundamental motion skills and fine-tune it with task-specific rewards—using a reference trajectory merely as a hint to guide exploration. This integrated approach enables our robot to perform dynamic tasks such as throwing, lifting, and dragging with remarkable fidelity.

II. Method

Our training pipeline (see Figure 2) is separated into two phases: 1) real-to-sim calibration (Section II-A) and 2) WBC training (Sections II-B and II-C). The real-to-sim calibration phase involves collecting data on the real robot and training a UAN to close the sim-to-real gap for non-ideal actuation mechanisms. Similar to past work [7, 21, 31, 42], our WBC training is split into two distinct sub-phases: pre-training (Section II-B) and fine-tuning (Section II-C). After pre-training, the policy can track reference trajectories if provided as a sequence of base velocity and end effector pose commands. During the fine-tuning phase, the policy observes a reference task trajectory. This helps warm start exploration when learning a new task because the policy can simply track these commands to achieve reasonable task performance. Through training with the task reward itself rather than a tracking reward, the policy learns how to depart from the reference trajectory to achieve higher task performance. Our simulation environments for the pre-training and fine-tuning phases rely on the same strategies for sim-to-real transfer, including domain randomization (Section II-B) and a UAN (Section II-A).

Our experiments consider a Unitree B2 quadruped with a modified Unitree Z1 Pro arm mounted on its back. The quadruped is $65\,\mathrm{cm}$ tall when standing and weighs $60\,\mathrm{kg}$, while the arm is $74\,\mathrm{cm}$ fully extended and weighs $6.8\,\mathrm{kg}$. The system has 19 actuated joints: 3 for each leg, 6 for the arm, and 1 for the gripper.

A. Unsupervised Actuator Net

Some actuators are challenging to model in simulation, especially when they have complex transmission mechanisms. In such cases, standard domain randomization and online system identification techniques may be insufficient, and instead, it is preferable to learn to model the actuator directly from hardware data. Previous approaches rely on output torque sensing [13], which is still uncommon in consumer hardware, to learn how to predict the motor's torque. Alternatively, we propose a method for matching the transition dynamics of the actuator such that

$$\min_{f_{\mathrm{sim}}} ||f_{\mathrm{real}}(\mathbf{s}, \boldsymbol{\tau}) - f_{\mathrm{sim}}(\mathbf{s}, \boldsymbol{\tau})||. \tag{1}$$

To influence the simulator dynamics, f_{sim}, we learn a residual model, $\pi_{\mathrm{UAN}}(\mathbf{e}$, that observes a history of position and velocity errors, $\mathbf{e}$, and outputs a corrective torque, $\delta\boldsymbol{\tau}$, for the simulator such that

$$\min_{\pi_{\mathrm{UAN}}} ||f_{\mathrm{real}}\big(\mathbf{s}, \boldsymbol{\tau}\big) - f_{\mathrm{sim}}\left(\mathbf{s}, \boldsymbol{\tau} + \pi_{\mathrm{UAN}}\left(\mathbf{e}\right)\right)||. \tag{2}$$

The corrective torques needed to minimize the transition error are unlabeled, so we parametrize π_{UAN} as a neural network and train it with RL.

1) Architecture and observation space: The network is designed as a 2-layer MLP with layer sizes [128, 128] and ELU activations. It is executed at every simulation time step (5 ms). Assuming each arm joint is identical, a single UAN is shared across all of the arm's actuators, with each actuator being processed independently by the shared network [13]. We constrain the observation space to include a history of the past 20 (equivalent to 100 ms) position and velocity errors for each relevant actuator. These design choices help prevent overfitting to other aspects of the training data, such as inertial coupling. Also, sharing the data across actuators improves data efficiency. For example, the actuator net is trained on data with various loads, as actuators closer to the robot's base generally experience more load than those near the gripper.

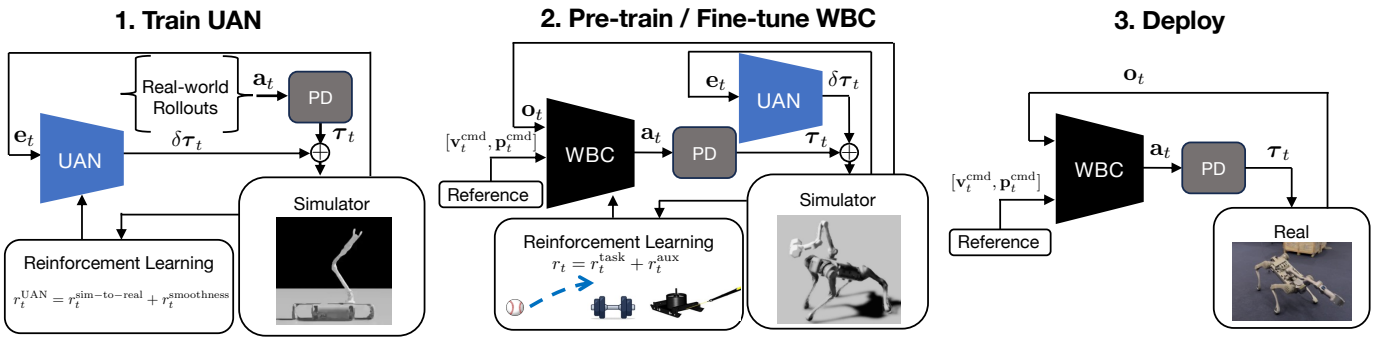


Fig. 2: **Unsupervised Actuator Network (UAN) approach for real-to-sim-to-real.** Our training pipeline involves three steps: 1) Train a UAN to close the sim-to-real gap for actuators with complex transmission mechanisms by mapping a history of joint position and velocity errors, $\mathbf{e}_t$, to corrective torques, $\delta\boldsymbol{\tau}_t$, 2) Pre-train a WBC using random motion references (base velocity and EE pose), then and fine-tune it on an athletic task reward with the UAN in loop, and 3) Deploy. During the fine-tuning phase, the WBC initially tracks the task-specific reference, and then gradually learns to depart from the reference to maximize task performance.

2) Data collection: We collect data on the real hardware to construct a dataset of transitions $\{(\mathbf{s}_t, \boldsymbol{\tau}_t, \mathbf{s}_{t+1})_i\}_{i=0}^N$ from each actuator. Our intention during data collection was to sufficiently cover the state space to avoid overfitting. Thus, we opted not to use policy data, and instead, collected data with three types of action sequences: 1) square waves, 2) sine waves, and 3) gaussian noise. For the square and sine wave data, we passed torque commands to one actuator at a time, while keeping the rest of the actuators at a fixed position target. We swept 12 different combinations of amplitude and frequency for each wave, resulting in about 50 seconds of data for each actuator. For the gaussian noise data, we passed torque commands to all the robot's joints simultaneously. We sampled a new action from a gaussian distribution every 5 to 400 ms for about 5 minutes.

3) Training Environment: We designed the training environment in Isaac Sim [30] with 4096 parallel environments. We train policies with the RSL-RL implementation [36] of PPO [37] with default hyperparameters, minus a few modifications (see Appendix A for the full list of learning algorithm hyperparameters). Following Radosavovic et al. [34], we apply a separate, fixed learning rate to the critic while using an adaptive learning rate for the actor. Additionally, we divide the data of each epoch into four mini-batches for the actor while using the entire batch for the critic, as we found that larger batch sizes produce more stable gradients and result in lower value function loss.

4) Task Design: For each environment at each timestep, we uniformly sample a real-world transition, $(\mathbf{s}_t, \boldsymbol{\tau}_t, \mathbf{s}_{t+1})_k$, and set the state of the simulator to match $\mathbf{s}_t$ and the initial torque to $\boldsymbol{\tau}_t$. After policy inference, we modify the torque by adding the correction, $\delta\boldsymbol{\tau}_t$, and then step the simulator. We then compute the reward as

$$r_t^{\text{UAN}} = r_t^{\text{sim-to-real}} + r_t^{\text{smoothness}} \tag{3}$$

where $r_t^{\text{sim-to-real}}$ aims to minimize the difference between the real joint position and the simulated joint position, and $r_t^{\text{smoothness}}$ biases exploration to gradual deviations. For a complete list of reward terms, please refer to Appendix A.

Each training episode consists of a 20 s rollout executing the torque sequence from the hardware data from $\delta\boldsymbol{\tau}_t$ to $\delta\boldsymbol{\tau}_{t+20\text{s}}$. Through training on rollouts, the actuator net learns to remain stable across many simulation time steps.

B. Whole-body Controller Pre-training

Before training on task-specific behaviors, we pre-train the WBC to learn foundational trajectory-tracking skills. Our training scheme builds upon the method proposed in [6] by incorporating a strategy for learning to track an EE orientation command. As in Section II-A3, we designed the training environment in Isaac Sim with 4096 parallel environments and trained the policies with PPO [36, 37] (using separate learning rates and batch sizes for the actor and critic).

1) Policy Architecture: The WBC is a control policy, $\mathbf{a}_t = \pi_\theta(\mathbf{o}_{t-H:t})$, where the action at time t, $\mathbf{a}_t$, is a vector of position targets for each of the robot's joints and $\mathbf{o}_{t-H:t}$ is an observation history of length $H = 10$ timesteps (200 ms). We parameterize π_θ as a 3-layer multi-layer perceptron (MLP) with layer sizes $[512, 512, 512]$ and ELU activations. The value function approximator network has the same architecture but does not share weights with the policy.

2) Observation Space: The policy's observation space consists of proprioceptive readings from the robot's onboard sensors, including the gravity vector projected in the robot's body frame $\mathbf{g}$, a base velocity command $\mathbf{v}_t^{\text{cmd}}$, an end effector pose command $\mathbf{p}_t^{\text{cmd}}$, the joint positions $\mathbf{q}$, the joint velocities $\dot{\mathbf{q}}$, the previous actions $\mathbf{a}_{t-1}$, and a timing variable $\omega_t = \sin(2\pi f t)$ with $f = 2.2$ Hz corresponding to the gait cycle frequency. Additionally, the observation includes a d-dimensional task embedding vector $\mathbf{z}_t$ (set to zero during pre-training).

3) Sim-to-Real Considerations: Our approach for bridging the sim-to-real gap uses a combination of domain randomization (DR) and real-to-sim calibration. To learn locomotion behaviors robust to terrain variations, we randomize terrain

roughness, friction, and restitution. To account for inaccuracies in the robot's URDF, we randomize the mass and center of mass position of each of the robot's links. We also randomize the PD gains and stall torques for each actuator in the robot's legs, and the policy lag length to learn robustness to latencies observed on hardware. To encourage learning recovery behaviors, we randomize the initial joint and body states of the robot and periodically perturb it with external forces and torques at the base, hips, feet, and end-effector, following the approach proposed in [8]. The DR ranges used for both pre-training and fine-tuning are provided in Appendix A.

Inspired by [43], we clip the commanded motor torques $\boldsymbol{\tau}$ such that

$$\boldsymbol{\tau} \geq -\boldsymbol{\tau}_{\max}\left(1+\max\left(\min\left(\frac{\dot{\mathbf{q}}}{\dot{\mathbf{q}}_{\max}},0\right),-1\right)\right), \quad (4)$$

$$\boldsymbol{\tau} \leq \boldsymbol{\tau}_{\max}\left(1-\max\left(\min\left(\frac{\dot{\mathbf{q}}}{\dot{\mathbf{q}}_{\max}},1\right),0\right)\right). \quad (5)$$

where $\boldsymbol{\tau}_{\max}$ and $\dot{\mathbf{q}}_{\max}$ are the maximum torques and velocities of the actuators, respectively. This clipping strategy enforces a physical motor constraint by ensuring that torque commands do not demand power beyond the motor's maximum output capacity. Furthermore, we clip the arm torques a second time to satisfy the constraint

$$|\boldsymbol{\tau}|^{\top}|\dot{\mathbf{q}}| \leq P_{\max}, \quad (6)$$

where $P_{\max}$ is the maximum total power of the arm joints, because we found experimentally this helps prevent the arm from entering a power protect state enforced by the robot's manufacturer.

Since typical DR strategies were insufficient for athletic behaviors in the arm (which uses harmonic drives), we incorporate the UAN (Section II-A) for the arm actuators. Therefore, no DR is applied to arm joint properties.

4) Task Specification: The pre-training task for the WBC is to track a desired base velocity and EE pose. The velocity command, $\mathbf{v}_t^{\text{cmd}} = \left[v_{x,t}^{\text{cmd}}, v_{y,t}^{\text{cmd}}, \omega_{z,t}^{\text{cmd}}\right]$, consists of a desired forward velocity $v_{x,t}^{\text{cmd}}$, a desired lateral velocity $v_{y,t}^{\text{cmd}}$, and a desired yaw-rate $\omega_{z,t}^{\text{cmd}}$. We command the EE pose in a yaw-rotated frame aligned with the robot's center of mass at a fixed height above the terrain. The choice of frame encourages the robot to coordinate with its legs to expand its workspace. The EE command $\mathbf{p}_t^{\text{cmd}} = \left[p_{\text{EE},t}^{\text{cmd}}, o_{\text{EE},t}^{\text{cmd}}\right]$ comprises a cartesian position p_t^{EE} and orientation o_t^{EE} (provided as the first two columns of a rotation matrix).

5) Reward Function: The reward function is split as $r_t = r_t^{\text{track}} + r_t^{\text{aux}}$, where r_t^{track} include tracking terms (EE pose, base velocity) and gait terms, while r_t^{aux} includes regularization and smoothing terms. The EE tracking term rewards minimizing the distance between four key points, where one key point is positioned at the frame's origin, and the others are positioned along each axis of the frame. Full details are provided in Appendix A.

6) Command Sampling Scheme: We adopt the approach first proposed in [6] to sample commands during training. We sample a new base velocity command and a new goal end effector pose every 7 seconds of simulation time. Upon sampling, the command is linearly interpolated (over 2 to 5 seconds) from the previous command. While this sampling scheme suffices for foundational loco-manipulation skills, it may be too smooth for highly agile motions – this motivates our task-specific fine-tuning (Section II-C).

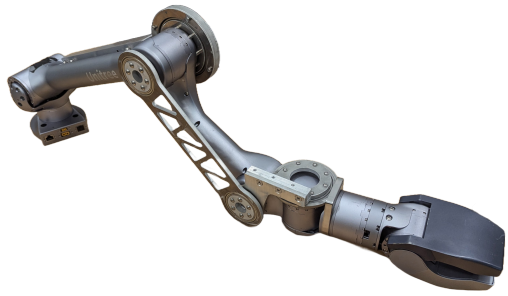


Fig. 3: **Unitree Z1 Pro arm.** This arm's harmonic actuators behave substantially differently from the quasi-direct-drive motors common in small legged robots. This image also shows the reinforcements we designed to ensure that the limit on athleticism comes from actuation rather than the linkage structural integrity.

C. Task-Specific Finetuning

After pre-training, the policy can track reference trajectories, but struggles on high-acceleration tasks. To address this, we fine-tune the policy directly with task rewards. The same WBC base policy weights can be reused for multiple task policies, thus avoiding repeated pre-training.

1) Initialization: The policy weights are initialized to those learned during pre-training. To avoid policy collapse, we set a low initial learning rate (1×10^{-5}) for the actor and retain the standard deviation from pre-training. Additionally, we set the entropy coefficient in PPO to zero during fine-tuning to improve training stability.

2) Reference trajectory and task embedding: During fine-tuning, the policy receives a task-specific reference trajectory and a one-hot task embedding to inform which phase of the task (e.g., set-up, execute, settle) is active. We hand-designed the reference trajectories through joint interpolation and forward kinematics, but they could also come from an expert policy or human demonstration.

3) Fine-tuning with task reward: The environment for fine-tuning phase extends that of pre-training (same DR ranges, external pushes, etc.). The reward becomes $r_t^i + r_t^{\text{aux}}$, where r_t^i is task-specific. Initially, the policy tracks the reference, aiding exploration; later, it learns to deviate to maximize task performance.

III. Experimental Setup

We chose the Unitree B2 with Unitree Z1 Pro arm as our hardware platform, and we consider three athletic tasks: throwing, weight lifting, and sled pulling (see Section IV-C).

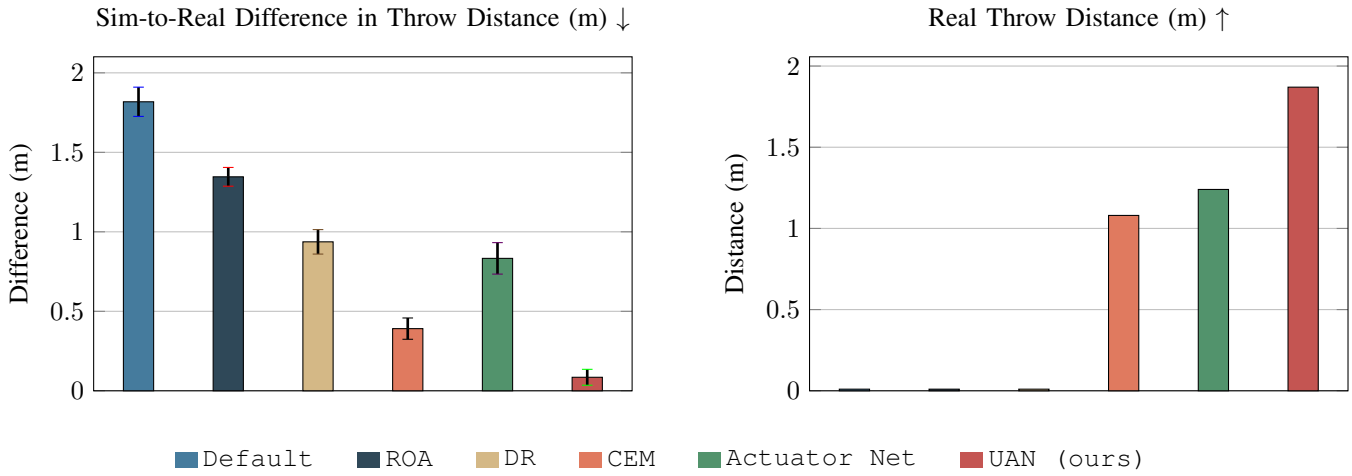


Fig. 4: **UAN improves simulator accuracy and real throwing performance.** `UAN (Ours)` achieves lower sim-to-real difference in throw distance as compared to standard baselines, resulting in a better real throw distance. For this comparison, we train and test policies with a fixed-base arm, to avoid the risk of the legged base falling during performance-critical ablations.

Structural upgrades to the arm were custom designed and fabricated to withstand the high loads during athletic behaviors (see Section III-A).

Following our UAN training (Section II-A), we pre-trained a WBC (Section II-B) and then fine-tuned policies for each task (Section II-C). Ablations comparing our method with alternatives are described in Section IV-A and Section IV-B.

A. Arm Modifications

During development, the Unitree Z1 Pro arm experienced structural failures at links 2 and 4, with minor deformations at link 5. The damage resulted from the highly dynamic movements in the athletic experiments, which applied loads to the links that exerted excessive stress and strain on the links exceeding the material's yield strength. Modifications were made to reinforce links 2, 3, 4, and 5 by adding supports at the joints. This prevents loads from being transferred solely through the motors which are cantilevered. A mass-efficient aluminum square tube was used for link 2, which experiences the highest stress of all the links. Idler bearings are used to apply support at the motor outputs without restricting their movement. In the URDF, link masses, centers of mass, and inertias were updated based on CAD calculations and the parallel axis theorem. Figure 3 shows the reinforced arm.

IV. Experimental Results

In this section, we report ablations that identify the contribution of key system components and present results for the athletic tasks. Supplemental videos are provided on the project website: https://uan.csail.mit.edu/.

Our experiments address the following questions:

1) Does our **unsupervised actuator net** reduce the sim-to-real gap and improve transfer?
2) What are the benefits of our two-stage **pre-training and fine-tuning** pipeline relative to alternatives?
3) Does our approach enable sim-to-real transfer of **athletic whole-body control tasks**?

A. Comparing System Identification Approaches

We compare several methods for modeling the actuator dynamics of the Unitree Z1 Pro arm in Isaac Sim. In particular, we consider:

1) **`Default`**: The baseline simulator with no additional modifications.
2) **`DR`**: The simulator augmented with domain randomization (randomizing PD gains, friction, and armature parameters).
3) **`ROA`**: A domain randomization baseline enhanced with an online system identification module via Regularized Online Adaptation [6].
4) **`Actuator Net`**: A supervised actuator network following Hwangbo et al. [13] where torque labels are estimated from the motor current. (Note that these labels do not capture the nonlinear effects introduced by the harmonic reducers.)
5) **`CEM`**: A method in which friction, frictional damping, and armature parameters are optimized using the cross-entropy method to minimize the mean-square joint position error between simulation and hardware.
6) **`UAN`**: Our proposed unsupervised actuator network that learns corrective torques without requiring torque supervision, thereby capturing both lag and nonlinearities from harmonic reduction.

We first evaluate the modeling accuracy of these approaches by reporting the mean-square joint position error on both the training data and on an unseen test trajectory (see Figure 7). Our results show that the `UAN` method achieves the best fit, suggesting excellent generalization. For example, detailed windows of simulator rollouts for a single arm joint are provided in Figures 7b (training data) and 7c (test data);

Fig. 5: **End-to-end fine-tuning from a pre-trained WBC leads to the best task performance.** Throwing evaluation metrics across 100 simulated throws for four policies: Our fine-tuned WBC (`Ours`) achieves the longest throw distance with lower peak leg power as compared to a throwing policy trained from scratch (`No-Pre-Training`) or a high-level policy for a frozen WBC (`No-E2E`). The WBC before finetuning (`No-Fine-Tuning`) has the lowest peak leg power but throws the ball a much shorter distance.

additional results for the other arm joints are included in the appendix (Figures 8 and 9). In our observations, the `CEM` method helps prevent overshoot (by effectively slowing the arm to match the lower joint velocities seen on hardware). `Actuator Net` can improve over the baseline by capturing lag effects, but it diverged on the 5 min rollouts on the training data. However, only `UAN` achieves a tight fit to the training data, thanks to its capacity to model the nonlinear effects introduced by the harmonic reducers. A shown by Figure 7, `UAN` can even accurately capture the arm's response to Gaussian noise control input, which is commonly used for exploration in reinforcement learning but represents a challenging regime for accurate simulation where the baseline methods degrade substantially.

To further assess these system identification methods in a task context, we trained arm-only throwing policies in simulation augmented with each approach and deployed them on hardware. The average throwing performance in simulation and reality is presented in Figure 4. In simulation, although the `Actuator Net` and `CEM` produced a promising throw, its behavior did not transfer as well to hardware. In contrast, the `UAN` policy achieved the farthest throws on hardware with the smallest sim-to-real gap. Meanwhile, the `Default`, `DR`, and `ROA` policies produced unstable behaviors—the `Default` policy, for instance, strayed excessively and failed to throw the ball at all.

B. Finetuning Foundational WBC

We compare four throwing policies to assess the impact of our pre-training and fine-tuning approaches:

1) **`No-Fine-Tuning`**: a pre-trained WBC that tracks a throwing reference trajectory.
2) **`No-Pre-Training`**: a throwing policy trained from scratch.
3) **`No-E2E`**: a high-level policy that outputs commands for a frozen pre-trained WBC.
4) **`Ours`**: our method that initializes with the pre-trained WBC and fine-tunes with RL.

All methods observe a hand-designed throwing reference trajectory.

Figure 1 presents the performance of each throwing policy across 100 simulated throws. `No-Fine-Tuning` successfully throws the ball by tracking the reference, but its performance is sub-optimal. However, the strong performance of `No-E2E` shows that the WBC's performance can be improved by providing a better reference trajectory. Still, the `No-E2E` policy does not perform to the maximum capability of the hardware. Through RL finetuning with the task reward, `Ours` can learn to throw farther while using a reduced peak power output in its leg motors. While `No-Pre-Training` could theoretically push the capabilities of the hardware, in practice, it struggles to do so due to exploration challenges. We found that `No-Pre-Training` achieved similar throwing performance to `No-E2E`, despite hitting a larger peak power output in its legs.

C. Hardware Results

1) Ball throwing: The task objective is to throw a 100 g ball as far as possible. Because grasping and releasing a ball directly is challenging for our gripper, a small bucket is attached to the robot's EE. The policy leans back prior to throwing, then pushes with its hind legs while swinging its arm forward to launch the ball. Figure 6 shows side-by-side snapshots from simulation and hardware. On hardware, the ball was thrown approximately 20 m, with the real robot

throwing slightly further than in simulation – possibly due to inaccuracies in the ball-bucket contact modeling.

2) Dumbbell snatch: The goal is to lift a dumbbell with the EE and hold it stably. The dumbbell is simulated by modifying the gripper's mass. The robot first lowers its EE to the ground, at which point the mass is added to its gripper. Then, its commanded to lift the weight in the air. When lifting, the robot is rewarded for maximizing the z position of its EE.

When training the lifting policy, we randomized the mass of the robot's EE from 0 to 10 kg. At convergence, the policy could consistently lift weights up to 8kg, but struggled to stabilize heavier weights above its body. Since the robot's arm is much weaker than the legs, the policy learns to pitch its base backwards to swing the weight upwards into the air. Figure 6 includes snapshots of the learned lifting behavior in simulation and reality. During hardware experiments, we secured the dumbbells inside the robot's gripper with a belt to prevent it from slipping out of the robot's grasp, We found the Z1 arm could not lift even a 5 lb. dumbbell to an upright position through simple joint interpolation. We first verified the whole-body policy could lift a 5 lb. dumbbell and then progressed to a 10 lb. dumbbell. In both experiments, the robot lifted the weight above its base and maintained it there stably for over $5\,\mathrm{s}$.

3) Sled pull: In this task, the robot pulls a heavy sled attached by a rope to its EE. The sled is modeled as avirtual, 3-dimensional mass-spring-damper system. The robot is rewarded for tracking a backward base velocity while minimizing lateral drift. The policy learns to adopt a low stance to maintain balance and extend its arm to avoid applying unnecessary torques to the arm's actuators. In simulation, policies successfully pulled weights up to $150\,\mathrm{kg}$. On hardware, the robot pulled a cart resisting a friction force of $113\,\mathrm{N}$ over 10 meters; a heavier cart (requiring $230\,\mathrm{N}$) was only pulled about 0.5 meters.

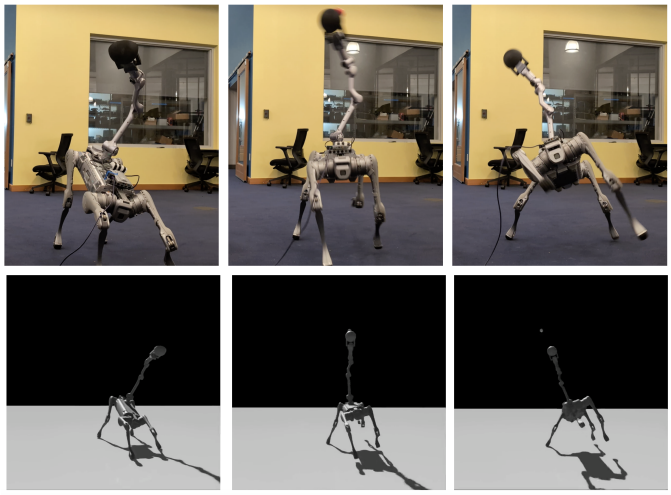
V. Limitations

Our fine-tuning approach requires a task reference trajectory, which may not be available for all robot morphologies or tasks. It also necessitates per-task engineering of the training environment (reward functions, object simulation, etc.). Future work might employ generative models to automatically synthesize task references. Additionally, our unsupervised actuator net focuses on the arm actuators. Extending real-to-sim calibration to other robot subsystems and modeling structural integrity are promising future directions.

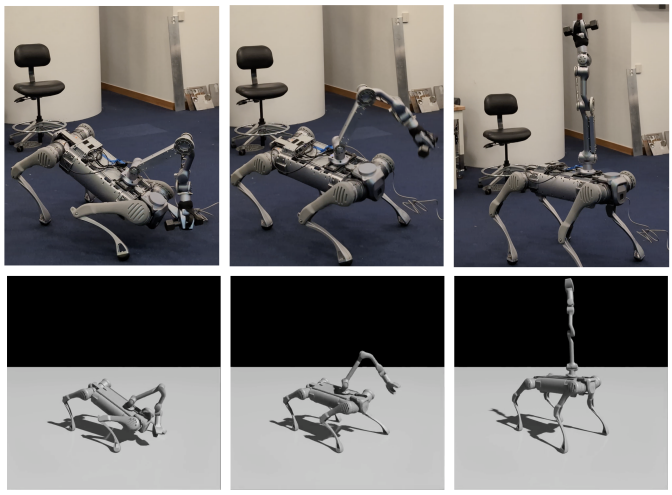
VI. Related Work

A. Whole-Body Control

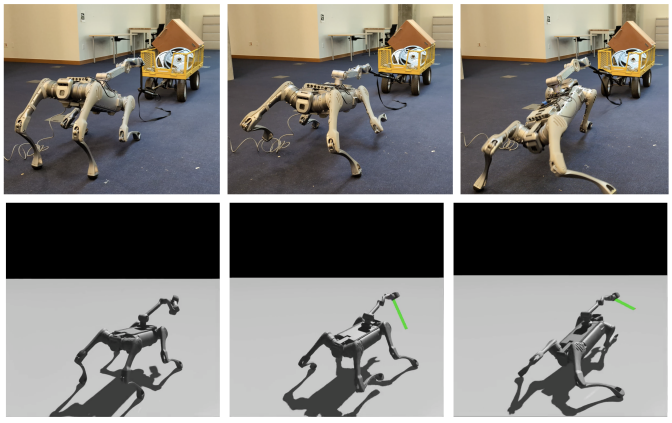
Walking robots with arms present a formidable challenge for control due to their many degrees of freedom and complex dynamics. A typical paradigm is to implement a *WBC* that optimizes actuation to achieve control objectives considering a model of the robot's kinematics and dynamics [41]. WBC approaches based on offline trajectory optimization or online



(a) Ball Throw



(b) Dumbbell Snatch



(c) Sled Pull

Fig. 6: **Real and simulated snapshots of athletic tasks.** Visualizing simulated and real rollouts of whole-body behaviors.

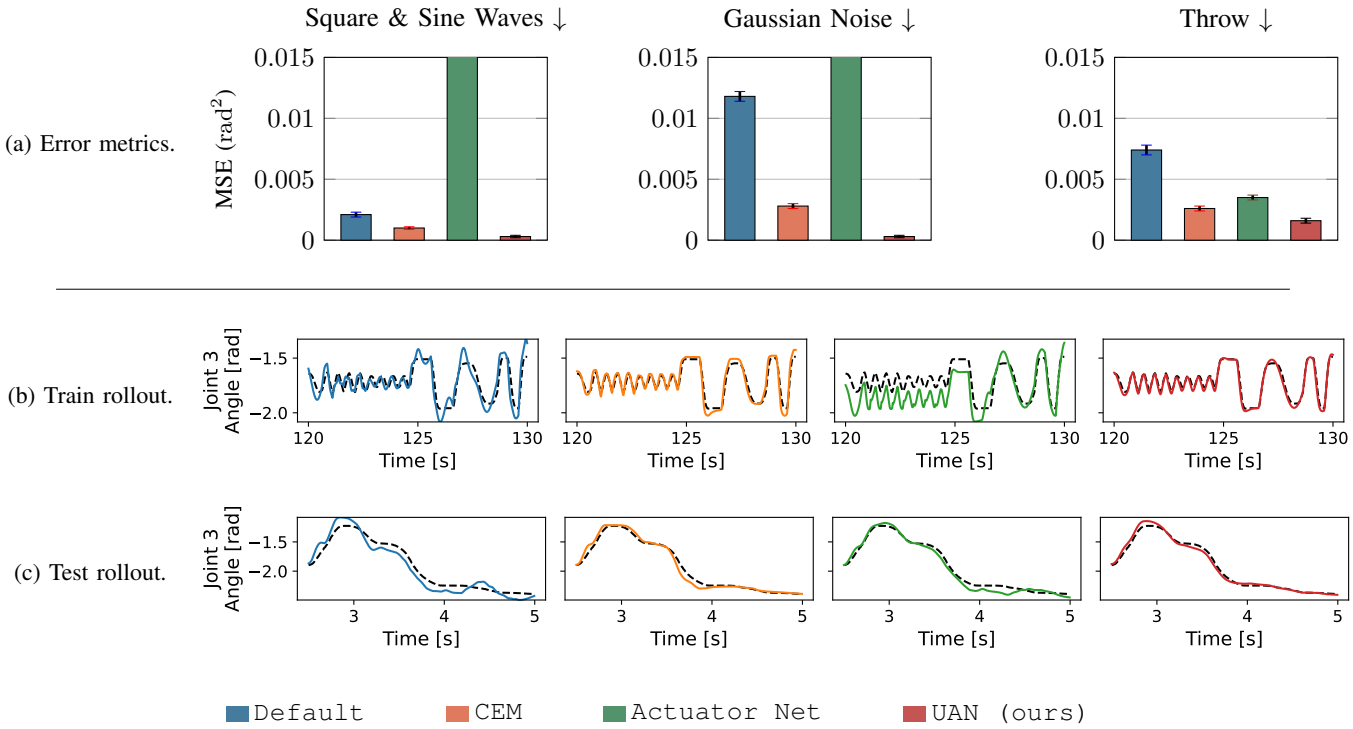


Fig. 7: **UAN achieves the tightest real-to-sim fit to the training data, as well as a throw trajectory unseen during training.** We rolled out three real-world joint trajectories: square & sine waves at each joint, Gaussian noise across all the joints, and a throw. Square waves, sine waves, and gaussian noise were seen during training, while the throw was not. We found that `Actuator-Net` error remains bounded on the 5 s throw trajectory but diverges when rolling out the 5 min training trajectories, while the `UAN` learned to remain accurate across long rollouts through RL training.

optimization with reduced-order models have achieved considerable success in dynamic walking and manipulation [1, 3, 28, 44]. Recently, reinforcement learning in simulation has enabled whole-body control that can naturally handle model uncertainty, e.g. uncertain terrain and robot properties [6]. In the case of reinforcement learning-based whole-body control, the controller is a neural network that is commanded with an input reference position [4, 6], force [32], or whole-body pose [5, 12, 23, 24] and outputs joint-space actions.

It is common to teleoperate legged-armed robots by parsing a reference trajectory from a human's movements in real-time and tracking it with a WBC; such an approach can accomplish expressive [4], forceful [32], or dexterous [7] tasks. One may also train a high-level policy to select reference trajectories or a latent representation autonomously in place of the teleoperator, using either learning from demonstration [7, 10] or reinforcement learning [21, 24]. However, some tasks may not be achievable by any choice of reference trajectory if they require a motion outside the training distribution of the WBC. It is challenging to formulate a generic pre-training scheme for whole-body control that anticipates all kinds of tasks one might want to perform for humanoids, motion capture datasets can provide diverse feasible reference commands [23], but for quadruped manipulators, pre-training commonly defaults to tracking procedurally generated smooth trajectories within the workspace [6].

To avoid the reliance on high-quality pre-training, another possibility is to discard the explicit notion of reference trajectories altogether and directly train end-to-end policies for specific tasks such as fall recovery [25], door opening [38], or soccer [11, 15, 16]. This enables the policy to learn highly dynamic motions to optimize the task reward, but, in practice, these motions can be hard to find due to fundamental exploration challenges in RL. We address this challenge by initializing the policy with pre-trained WBC weights and a reference trajectory.

B. Overcoming the sim-to-real gap

Prior work proposed simulated athletic tasks as a benchmark for learned whole-body control [42, 24], though they left sim-to-real transfer as future work. In contrast, other studies have demonstrated sim-to-real transfer of athletic tasks on small robots with transparent actuators [11, 15, 16]. Achieving sim-to-real transfer for athletic behaviors on large robots with non-ideal actuators is especially challenging because even minor modeling discrepancies can lead to reward hacking. To address this, we introduce UAN, which leverages real-world data to bridge the sim-to-real gap.

DR is a common strategy to mitigate discrepancies between simulation and reality [17, 18]. In the field of dynamic legged

robots, common parameters to randomize include the proportional and derivative gains of each joint, the stall torques, the link masses and inertias, and terrain properties [18, 51]. Excessive DR can reduce peak performance if the policy cannot identify key parameters of the environment necessary to optimize its reward function. To overcome this challenge, previous work employed teacher-student frameworks, where a student policy learns to imitate an expert policy that has access to privileged observations related to its environment [6, 17, 18]. Alternatively, the policy may learn online system identification directly from an observation history. Some policy architectures (i.e., CNNs [20] and transformers [33]) have been shown to achieve in-context adaptation without relying on a teacher-student distillation.

Accurate system identification can reduce reliance on DR by mitigating the sim-to-real gap directly. Methods for identifying inertial properties typically rely on least-squares estimation [2], including a notable approach that leverages insights about the geometric structure of the robot's dynamics to provide robustness against local optima [19]. This method was applied to identify the inertial parameters of the MIT Humanoid [39]. In our work, we rely on the inertial properties provided in the manufacturer's URDF file.

Actuator modeling methods traditionally rely on parameterized physics models to capture effects such as static friction, dynamic friction, and reflected inertia [8], the last of which can be set through the "armature" setting in physics simulations such as Isaac Sim [26] and MuJoCo [48]. This approach can be insufficient for actuators with complex transmission mechanisms. To address this, Hwangbo et al. [13] proposed learning an actuator net, which is a neural network trained to predict an actuator's output torque from a history of position and velocity errors. The actuator net was added to the simulator during policy training to reduce the sim-to-real gap in ANYmal's series elastic actuators. Their approach, however, relies on torque sensing, which is uncommon in consumer robotic hardware. Schwendeman et al. [39] avoided reliance on an output torque sensor when training an actuator net by measuring the torques from current. However, this is only accurate in low-reduction and low-torque-density actuators which are efficiently backdriveable and have minimal reflected inertia. In contrast, our approach, UAN, employs an actuator net without relying on torque data. Instead, we train the network to predict corrective torques for the simulator that minimize the discrepancy between the simulated and real-world transition dynamics.

When ground-truth labels are unavailable (i.e., the robot's actuators lack torque sensing), they can be discovered through interaction to better match the real-world dynamics. For example, Zeng et al. [50] learned a residual model to better predict the ballistic motions of objects, enabling a manipulator to accurately throw them. Similarly, Gruenstein et al. [9] proposed learning residual actions for a simplified dynamics model for a legged microbot so that it transits to the same future states as a more complex dynamics model. In another study, Sontakke et al. [45] proposed learning a corrective external force policy to improve simulation accuracy for a buoyancy assisted legged robot. Mentee Robotics has publicly stated that they applied RL to train a delta action model using real-world data to overcome the sim-to-real gap on their humanoid, but the technical details of their approach remain unpublished [35]. While we also apply RL to correct our simulation model, we specifically target the sim-to-real gap for the robot's actuators with harmonic drives, which are notably hard to model. This focus leverages the parts of the simulator that are more accurate (i.e., rigid body mechanics) to reduce overfitting and also avoids reliance on a motion capture system.

VII. Conclusion

Legged manipulators promise enhanced strength and a larger workspace by coordinating arms and legs. We proposed a training pipeline that first pre-trains a whole-body controller and then fine-tunes it using task rewards, while simultaneously reducing the sim-to-real gap via our UAN. Our experimental results on ball throwing, dumbbell lifting, and sled pulling demonstrate the viability of this approach. Future work may extend the sim-to-real calibration to additional subsystems and incorporate structural integrity constraints directly in the training. Future work may extend the real-to-sim calibration to additional subsystems and incorporate structural integrity constraints directly in the training.

Acknowledgment

We thank the members of the Improbable AI lab—especially Sandor Felber, Chen Bo Calvin Zhang, Srinath Mahankali, and Zhang-Wei Hong—for helpful discussions and feedback. We acknowledge Unitree Robotics for technical support provided for their robots. We are grateful to MIT Supercloud and the Lincoln Laboratory Supercomputing Center for providing HPC resources. We also acknowledge the MIT CSAIL Living Lab project for providing robot hardware. This research was partly supported by Hyundai Motor Company, the MIT-IBM Watson AI Lab, and the National Science Foundation under Cooperative Agreement PHY-2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, http://iaifi.org/) and the National Science Foundation Graduate Research Fellowship under Grant No. 2141064. This research was also sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-21-1-0328. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

Author Contributions

- **Nolan Fey** contributed to ideation, implementation of the entire system, experimental evaluation, and writing.
- **Gabriel B. Margolis** contributed to ideation, implementation of some parts of the system, and writing.
- **Martin Peticco** contributed to hardware modifications and writing.
- **Pulkit Agrawal** advised the project and contributed to its development, experimental design, and writing.

References

[1] Yeuhi Abe, Benjamin Stephens, Michael P Murphy, and Alfred A Rizzi. Dynamic whole-body robotic manipulation. In *Unmanned Systems Technology XV*, volume 8741, pages 280–290. SPIE, 2013.

[2] Christopher G. Atkeson, Chae H. An, and John M. Hollerbach. Estimation of inertial parameters of manipulator loads and links. *The International Journal of Robotics Research*, 5(3):101–119, 1986. doi: 10.1177/027836498600500306. URL https://doi.org/10.1177/027836498600500306.

[3] C Dario Bellicoso, Koen Krämer, Markus Stäuble, Dhionis Sako, Fabian Jenelten, Marko Bjelonic, and Marco Hutter. Alma-articulated locomotion and manipulation for a torque-controllable robot. In *2019 International conference on robotics and automation (ICRA)*, pages 8477–8483. IEEE, 2019.

[4] Xuxin Cheng, Yandong Ji, Junming Chen, Ruihan Yang, Ge Yang, and Xiaolong Wang. Expressive whole-body control for humanoid robots. *arXiv preprint arXiv:2402.16796*, 2024.

[5] Pranay Dugar, Aayam Shrestha, Fangzhou Yu, Bart van Marum, and Alan Fern. Learning multi-modal whole-body control for real-world humanoid robots. *arXiv preprint arXiv:2408.07295*, 2024.

[6] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: Learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, pages 138–149. PMLR, 2023.

[7] Zipeng Fu, Qingqing Zhao, Qi Wu, Gordon Wetzstein, and Chelsea Finn. Humanplus: Humanoid shadowing and imitation from humans, 2024. URL https://arxiv.org/abs/2406.10454.

[8] Ruben Grandia, Espen Knoop, Michael Hopkins, Georg Wiedebach, Jared Bishop, Steven Pickles, David Müller, and Moritz Bächer. Design and control of a bipedal robotic character. 07 2024. doi: 10.15607/RSS.2024.XX.103.

[9] Joshua Gruenstein, Tao Chen, Neel Doshi, and Pulkit Agrawal. Residual model learning for microrobot control, 2021. URL https://arxiv.org/abs/2104.00631.

[10] Huy Ha, Yihuai Gao, Zipeng Fu, Jie Tan, and Shuran Song. Umi on legs: Making manipulation policies mobile with manipulation-centric whole-body controllers. *arXiv preprint arXiv:2407.10353*, 2024.

[11] Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H. Huang, Dhruva Tirumala, Jan Humplik, Markus Wulfmeier, Saran Tunyasuvunakool, Noah Y. Siegel, Roland Hafner, Michael Bloesch, Kristian Hartikainen, Arunkumar Byravan, Leonard Hasenclever, Yuval Tassa, Fereshteh Sadeghi, Nathan Batchelor, Federico Casarini, Stefano Saliceti, Charles Game, Neil Sreendra, Kushal Patel, Marlon Gwira, Andrea Huber, Nicole Hurley, Francesco Nori, Raia Hadsell, and Nicolas Heess. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89), April 2024. ISSN 2470-9476. doi: 10.1126/scirobotics.adi8022. URL http://dx.doi.org/10.1126/scirobotics.adi8022.

[12] Tairan He, Zhengyi Luo, Wenli Xiao, Chong Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Learning human-to-humanoid real-time whole-body teleoperation. *arXiv preprint arXiv:2403.04436*, 2024.

[13] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), January 2019. ISSN 2470-9476. doi: 10.1126/scirobotics.aau5872. URL http://dx.doi.org/10.1126/scirobotics.aau5872.

[14] Gwanghyeon Ji, Juhyeok Mun, Hyeongjun Kim, and Jemin Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, 7(2):4630–4637, 2022. doi: 10.1109/LRA.2022.3151396.

[15] Yandong Ji, Zhongyu Li, Yinan Sun, Xue Bin Peng, Sergey Levine, Glen Berseth, and Koushil Sreenath. Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1479–1486, 2022. doi: 10.1109/IROS47612.2022.9981984.

[16] Yandong Ji, Gabriel B. Margolis, and Pulkit Agrawal. Dribblebot: Dynamic legged manipulation in the wild. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5155–5162, 2023. doi: 10.1109/ICRA48891.2023.10160325.

[17] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots, 2021. URL https://arxiv.org/abs/2107.04034.

[18] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47), October 2020. ISSN 2470-9476. doi: 10.1126/scirobotics.abc5986. URL http://dx.doi.org/10.1126/scirobotics.abc5986.

[19] Taeyoon Lee, Patrick M. Wensing, and Frank C. Park. Geometric robot dynamic identification: A convex programming approach. *IEEE Transactions on Robotics*, 36(2):348–365, 2020. doi: 10.1109/TRO.2019.2926491.

[20] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control, 2024. URL https://arxiv.org/abs/2401.16889.

[21] Minghuan Liu, Zixuan Chen, Xuxin Cheng, Yandong Ji, Ruihan Yang, and Xiaolong Wang. Visual whole-body control for legged loco-manipulation. *arXiv preprint arXiv:2403.16967*, 2024.

[22] Junfeng Long, Zirui Wang, Quanyi Li, Jiawei Gao, Liu Cao, and Jiangmiao Pang. Hybrid internal model: Learning agile legged locomotion with simulated robot response, 2024. URL https://arxiv.org/abs/2312.11460.

[23] 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.

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

[25] Yuntao Ma, Farbod Farshidian, Takahiro Miki, Joonho Lee, and Marco Hutter. Combining learning-based locomotion policy with model-based manipulation for legged mobile manipulators. *IEEE Robotics and Automation Letters*, 7(2):2377–2384, 2022.

[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] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning, 2022. URL https://arxiv.org/abs/2205.02824.

[28] Michael P Murphy, Benjamin Stephens, Yeuhi Abe, and Al-

fred A Rizzi. High degree-of-freedom dynamic manipulation. In *Unmanned Systems Technology XIV*, volume 8387, pages 339–348. SPIE, 2012.

[29] I Made Aswin Nahrendra, Byeongho Yu, and Hyun Myung. Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5078–5084. IEEE, 2023.

[30] NVIDIA. Nvidia isaac-sim. *https://developer.nvidia.com/isaac-sim*, May 2022.

[31] Guoping Pan, Qingwei Ben, Zhecheng Yuan, Guangqi Jiang, Yandong Ji, Jiangmiao Pang, Houde Liu, and Huazhe Xu. Roboduet: A framework affording mobile-manipulation and cross-embodiment. *arXiv preprint arXiv:2403.17367*, 2024.

[32] Tifanny Portela, Gabriel B Margolis, Yandong Ji, and Pulkit Agrawal. Learning force control for legged manipulation. *arXiv preprint arXiv:2405.01402*, 2024.

[33] Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Real-world humanoid locomotion with reinforcement learning, 2023. URL https://arxiv.org/abs/2303.03381.

[34] Ilija Radosavovic, Sarthak Kamat, Trevor Darrell, and Jitendra Malik. Learning humanoid locomotion over challenging terrain, 2024. URL https://arxiv.org/abs/2410.03654.

[35] Mentee Robotics. Ai for humanoid robotics - a lecture by mentee robotics' ceo, prof. lior wolf, Aug 2024. URL http://www.youtube.com/watch?v=y1LG4YwUtoo&t=1391s.

[36] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning, 2022. URL https://arxiv.org/abs/2109.11978.

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[38] Clemens Schwarke, Victor Klemm, Matthijs Van der Boon, Marko Bjelonic, and Marco Hutter. Curiosity-driven learning of joint locomotion and manipulation tasks. In *Proceedings of The 7th Conference on Robot Learning*, volume 229, pages 2594–2610. PMLR, 2023.

[39] Laura Schwendeman, Andrew SaLoutos, Elijah Stanger-Jones, and Sangbae Kim. Improving domain transfer of robot dynamics models with geometric system identification and learned friction compensation. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, pages 1–8, 2023. doi: 10.1109/Humanoids57100.2023.10375233.

[40] Luis Sentis and Oussama Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 02(04):505–518, 2005. doi: 10.1142/S0219843605000594. URL https://doi.org/10.1142/S0219843605000594.

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

[42] Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation, 2024. URL https://arxiv.org/abs/2403.10506.

[43] Young-Ha Shin, Tae-Gyu Song, Gwanghyeon Ji, and Hae-Won Park. Actuator-constrained reinforcement learning for high-speed quadrupedal locomotion, 2023.

[44] 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.

[45] Nitish Sontakke, Hosik Chae, Sangjoon Lee, Tianle Huang, Dennis W Hong, and Sehoon Hal. Residual physics learning and system identification for sim-to-real transfer of policies on buoyancy assisted legged robots. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 392–399. IEEE, 2023.

[46] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots, 2018.

[47] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world, 2017.

[48] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

[49] Zhaoming Xie, Xingye Da, Michiel van de Panne, Buck Babich, and Animesh Garg. Dynamics randomization revisited:a case study for quadrupedal locomotion, 2021.

[50] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020.

[51] Chong Zhang, Wenli Xiao, Tairan He, and Guanya Shi. Wococo: Learning whole-body humanoid control with sequential contacts. *arXiv preprint arXiv:2406.06005*, 2024.

APPENDIX A
TRAINING DETAILS

The PPO hyperparameters across all tasks are provided in Table I. The ranges for domain randomization are provided in Table II. Table III details the WBC reward components, while Table IV shows the reward function for UAN training.

APPENDIX B
TASK ENVIRONMENTS

The auxiliary rewards (and scales) for each task match those in Table III underneath the dashed line.

A. Ball Throwing

The ball throwing task has three separate task states: `throw-set-up`, `throw`, and `settle`. The policy knows which state it is in by observing its task embedding, which includes a one-hot vector. Each state has a separate task reward for the desired behavior, such that

$$r_t^{\text{ball}} = b_{\texttt{throw-set-up}} r_t^{\texttt{throw-set-up}} + b_{\texttt{throw}} r_t^{\texttt{throw}} + b_{\texttt{settle}} r_t^{\texttt{settle}}$$

where b_i is 1 when the environment is in state i and 0 otherwise. The task reward for each state is

$$\begin{aligned} r_t^{\texttt{throw-set-up}} &= 5\big(-(p_z - 0.6)^2 - g_x^2 - g_y^2 \\ &\quad + ||q_{i,t} - q_{i,t}^{\text{ref}}||_1\big), \\ r_t^{\texttt{throw}} &= 20 \max\left(v_{x,t}^{\text{ball}}, 0\right) \\ &\quad + 20 \max\left(v_{z,t}^{\text{ball}}, 0\right) - 0.5 \left|v_{y,t}^{\text{ball}}\right|^2, \\ r_t^{\texttt{settle}} &= 5\big(-(p_z - 0.6)^2 - g_x^2 - g_y^2\big). \end{aligned}$$

The `throw-set-up` reward encourages an upright posture and tracking a trajectory to bring back the arm for a throw, then the `throw` reward is to maximize the ball's forward and upward velocities while minimizing its lateral velocity. The `settle` reward encourages the robot to stand upright after throwing and avoid falling over.

The environment transitions from `throw-set-up` to `throw` after 2.5 s in simulation time, and it transitions from `throw` to `settle` at 3.5 s. We set the ball's state to an arbitrary position at the start of the episode, and then we place it in the robot's bucket after 1.5 s. If the robot drops the ball in the `throw-set-up`, the environment terminates.

With this set up, we found that the policy would sometimes learn to lean during the `throw-set-up` task and eventually fall over unless it transitioned to the `throw` state. Thus, we modified the environment so that the policy remains in the `throw` state with a probability of 0.3. Similarly, we added a probability that the robot enters the `settle` state on reset to help the policy learn to stay still after executing the throw.

We found that the policy may repeat the throwing motion multiple times during the `throw` state because it cannot sense whether it is holding the 100 g ball through proprioception. Thus, we include a timer in the task embedding that linearly increments from 0 to 1 over 1 second during only the `throw` state, else it is 0. The policy can key into this timer to infer whether it has already thrown the ball.

B. Dumbbell Lifting

We separate the dumbbell lifting task into two states: `snatch-set-up` and `snatch`. The task rewards for each state are

$$\begin{aligned} r_{\texttt{snatch-set-up}} &= 5 \sum_i^3 \frac{1}{3} \exp\left(-2\left|p_{i,t}^{\text{cmd}} - p_{i,t}\right|\right) \\ &\quad - 0.005||F_t^{\text{EE}} - F_{t-1}^{\text{EE}}||, \\ r_t^{\texttt{snatch}} &= 4p_{z,t}^{\text{EE}} - 0.005||F_t^{\text{EE}} - F_{t-1}^{\text{EE}}|| \\ &\quad - \sum_i^{\{4,5,6\}} |q_i^{\text{arm}}|^2 . \end{aligned}$$

The reward during `snatch-set-up` encourages tracking a reference that guides the EE near the ground. The force smoothness term penalizes the robot for suddenly slamming its EE into the ground. The `snatch` task reward is to maximize the end-effector height. To avoid damaging the robot's arm, we added a reward term penalizes deviations from the wrist's nominal, straightened position.

We used a similar strategoy to the throwing task to ensure stable transitions—the environment transitions from `snatch-set-up` to `snatch` after 2.5 s with a probability of 0.9. Without this addition, the robot may fall over during an extended `snatch-set-up` state.

C. Sled Pulling

We model the sled as a virtual mass-spring-damper. When an episode begins, the robot is at a distance l from the sled. The pulling force between the EE and the sled is defined as

$$F_t^{\text{pull}} = \max\left(1000\,|d - l|_2^2 - \left|\dot{d}\right|_2^2, 0\right),$$

A lateral force is applied on the sled based on the friction and mass:

$$F^{\text{sled}} = F_x^{\text{pull}} + \mu\ m_{\text{sled}}\ g,$$

The sled task also has two states, `sled-set-up` and `pull`, where the reward terms are

$$\begin{aligned} r_t^{\texttt{pull}} &= 5 \sum_i^3 \frac{1}{3} \exp\left(-2\left|p_{i,t}^{\text{cmd}} - p_{i,t}\right|\right), \\ r_t^{\texttt{pull}} &= -5\left|v_{x,t}^{\text{cmd}} - v_{x,t}\right|_2^2 - \delta y_t - \delta\theta_t^{\text{yaw}} - \sum_i^{\{4,5,6\}} q_i^2, \end{aligned}$$

where δy_t and $\delta\theta_t^{\text{yaw}}$ are deviations from the starting state.

The environment transitions from `sled-set-up` to `pull` after 2.5 s of simulation time with full probability. Every 7 seconds, forward base velocity commands are sampled uniformly from 0 to $-1\,\text{ms}^{-1}$ with a probability of 0.8. Otherwise, the policy is given a command of $0\,\text{ms}^{-1}$. We found that the sled policies tended to drift excessively in the base-y and -yaw directions since the policy does not observe the robot's base position and orientation. Thus, when the forward base velocity command is non-zero, we set the lateral and yaw velocity commands as

$$v_{y,t}^{\text{cmd}} = -0.5\delta y_t, \tag{7}$$

$$\omega_{z,t}^{\text{cmd}} = -0.5\delta\theta_t^{\text{yaw}}, \tag{8}$$

to help guide the policy to pull the sled straight. On hardware, we provide forward, lateral, and yaw velocity commands from a joystick.

APPENDIX C
UAN TRAIN & TEST FITS

Simulator rollouts of each system identification method on training data and test data are provided in Figures 8 and 9.

Hyperparameter	UAN Value	Pre-train Value	Fine-tune Value
Discount Factor	0.995	0.99	0.99
GAE Parameter	0.95	0.95	0.95
Entropy Coefficient	0.0	0.01	0.0
Actor Learning Rate	Adaptive	Adaptive	Adaptive
Critic Learning Rate	5.e-4	5.e-4	5.e-4
KL Threshold	0.01	0.01	0.01
Horizon	96	24	24
Number of Environments	4096	4096	4096
Actor Minibatch Size	98304	24576	24576
Critic Minibatch Size	393216	98304	98304
# of Mini Epochs	5	5	5
Optimizer	AdamW	AdamW	AdamW
Weight Decay	0.01	0.01	0.01

TABLE I: PPO Hyperparameters

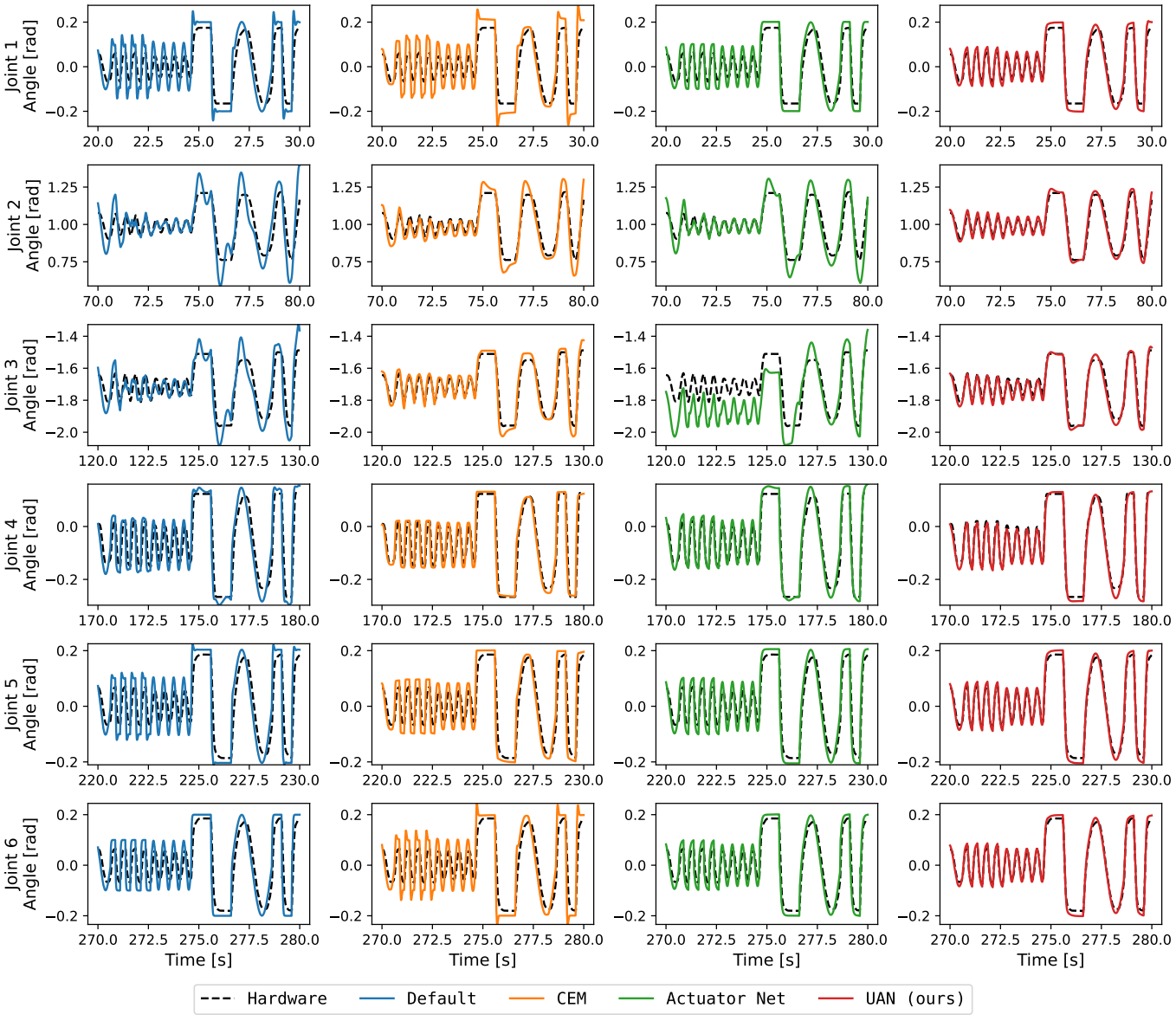


Fig. 8: Comparison of system identification methods on open-loop rollouts of the training data.

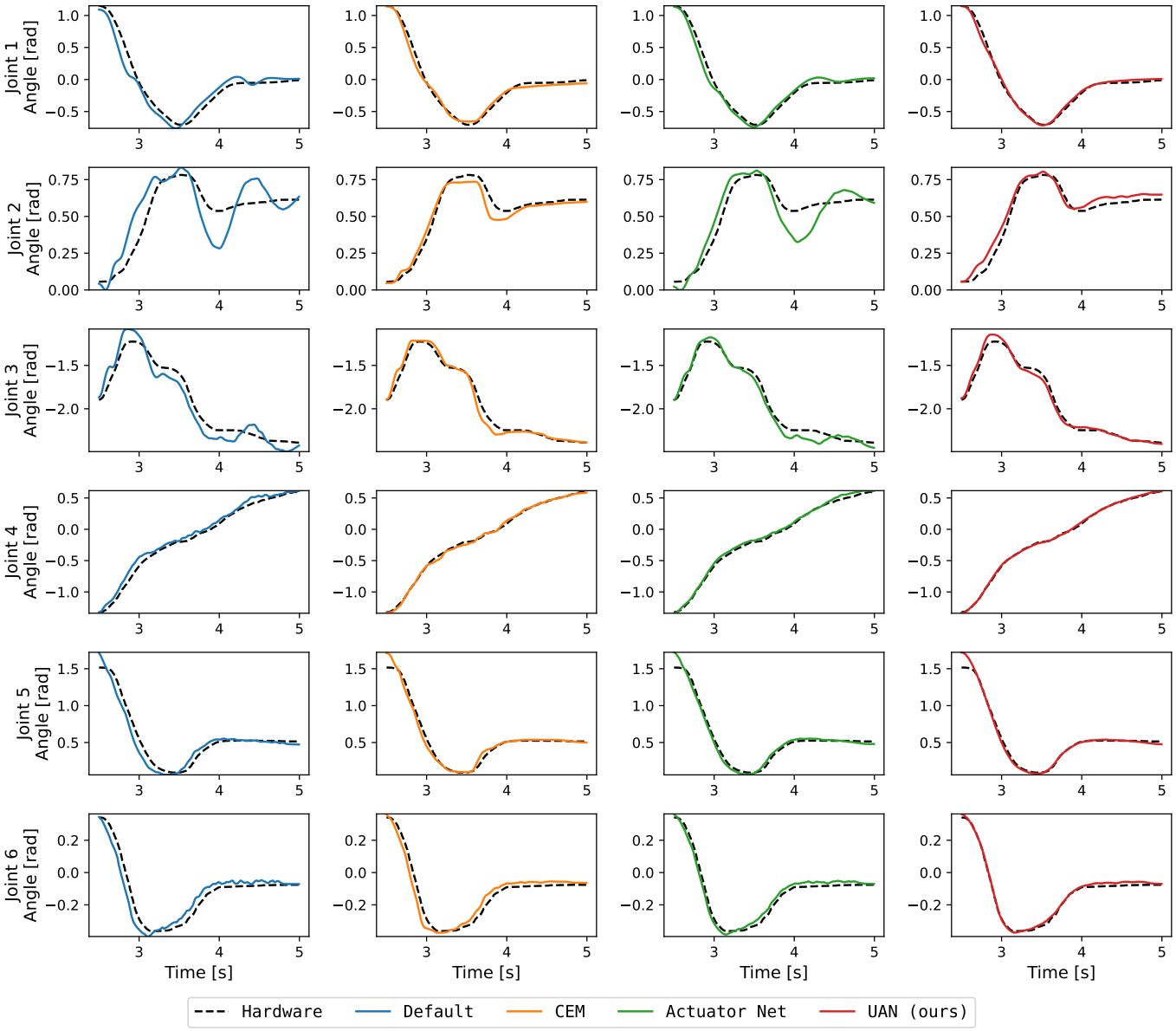


Fig. 9: Sim-to-real gap for a throwing trajectory unseen during training.

Parameter	Min	Max
Terrain Friction	0.5	4.0
Terrain Restitution	0.0	0.5
Terrain Roughness [cm]	0	2.5
Leg Joint Stiffness Scale	0.9	1.1
Leg Joint Damping Scale	0.5	1.5
Leg Stall Torque Scale	0.9	1.1
Link Mass Scale	0.5	1.5
Link Center of Mass Offsets [cm]	-2	2
Encoder Offset [rad]	-0.05	-0.05
Policy Lag Timesteps (5 ms)	0	6

TABLE II: WBC & Fine-tune Domain Randomization

Reward Component	Term	Scale
EE Pose Tracking	$\sum_i^3 \frac{1}{3}\exp\left(-2\left\|p_{i,t}^{\text{cmd}} - p_{i,t}\right\|\right)$	5.0
Linear Velocity Tracking	$\exp\left(-4\left\|\left[v_{x,t}^{\text{cmd}}, v_{y,t}^{\text{cmd}}\right] - \left[v_{x,t}, v_{y,t}\right]\right\|_2^2\right)$	2.0
Angular Velocity Tracking	$\exp\left(-4\left(\omega_{z,t}^{\text{cmd}} - \omega_{z,t}\right)^2\right)$	1.0
Gait	$\sum_{i\in\{\text{FR,FL,RR,RL}\}}\{\sim c_i\}\ \mathbb{1}\{p_{z,t}^i < 0.043\}$	-0.5
No Slip	$\sum_{i\in\{\text{FR,FL,RR,RL}\}}\{c_i\}\ \exp\left(-0.1\left\|\boldsymbol{v}_t^i\right\|_2^2\right)$	-0.5
Foot Clearance	$\sum_{i\in\{\text{FR,FL,RR,RL}\}}\{\sim c_i\}\ \left\|p_{z,t}^{\text{cmd},i} - p_{z,t}^i\right\|_2^2$	-40.0
Mechanical Power	$\left\|\boldsymbol{\tau}_t \cdot \dot{\boldsymbol{q}}_t\right\|$	-0.0001
Action Smoothness	$\left\|\mathbf{a}_t - \mathbf{a}_{t-1}\right\|_2^2 + \frac{1}{2}\left\|\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2}\right\|_2^2$	-0.05
Linear Velocity Z	$\left\|v_{z,t}\right\|^2$	-2.0
Angular Velocity XY	$\left\|\left[\omega_{x,t}, \omega_{y,t}\right]\right\|_2^2$	-0.05
Joint Positions	$\left\|\boldsymbol{q}_t^{\text{default}} - \boldsymbol{q}_t\right\|$	-0.25
Collision	$\{\left\|F_t^{\text{arm}}\right\|_2^2 > 0.1$ or $\left\|F_t^{\text{leg}}\right\|_2^2 > 0.1\}$	-5.0
Joint Position Limits	$\sum_i -\min(q_{i,t} - q_{i,t}^{\min}, 0) + \max(q_{i,t} - q_{i,t}^{\max}, 0)$	-10.0
Contact Force	$\sum_{i\in\{\text{FR,FL,RR,RL}\}}\left\|F_t^i\right\|_2^2$	-0.000004

TABLE III: WBC Rewards

Reward Component	Term	Scale
Joint Positions (L1)	$\left\|\boldsymbol{q}_t^{\text{real}} - \boldsymbol{q}_t^{\text{sim}}\right\|$	-1.5
Joint Positions (Relaxed)	$\exp\left(-100\left\|\boldsymbol{q}_t^{\text{real}} - \boldsymbol{q}_t^{\text{sim}}\right\|_2^2\right)$	4.0
Joint Positions (Moderate)	$\exp\left(-300\left\|\boldsymbol{q}_t^{\text{real}} - \boldsymbol{q}_t^{\text{sim}}\right\|_2^2\right)$	4.0
Joint Positions (Strict)	$\exp\left(-1000\left\|\boldsymbol{q}_t^{\text{real}} - \boldsymbol{q}_t^{\text{sim}}\right\|_2^2\right)$	5.0
Action Smoothness	$\exp\left(-0.5\left\|\mathbf{a}_t - \mathbf{a}_{t-1}\right\|\right)$	0.5

TABLE IV: Unsupervised Actuator Net Rewards