

Contact-UAN: Unsupervised Actuator Networks as a Residual World Model for Humanoid Control

Nolan Fey^{1,*}, Khai Nguyen^{*}, Yash Prabhu¹, Gabriel B. Margolis¹, Pulkit Agrawal¹

MIT, ¹Improbable AI Lab, ^{*}Co-first authors.

{nolanfey,khain,yprabhu,gmargo,pulkitag}@mit.edu

<http://contact-uan.csail.mit.edu/>

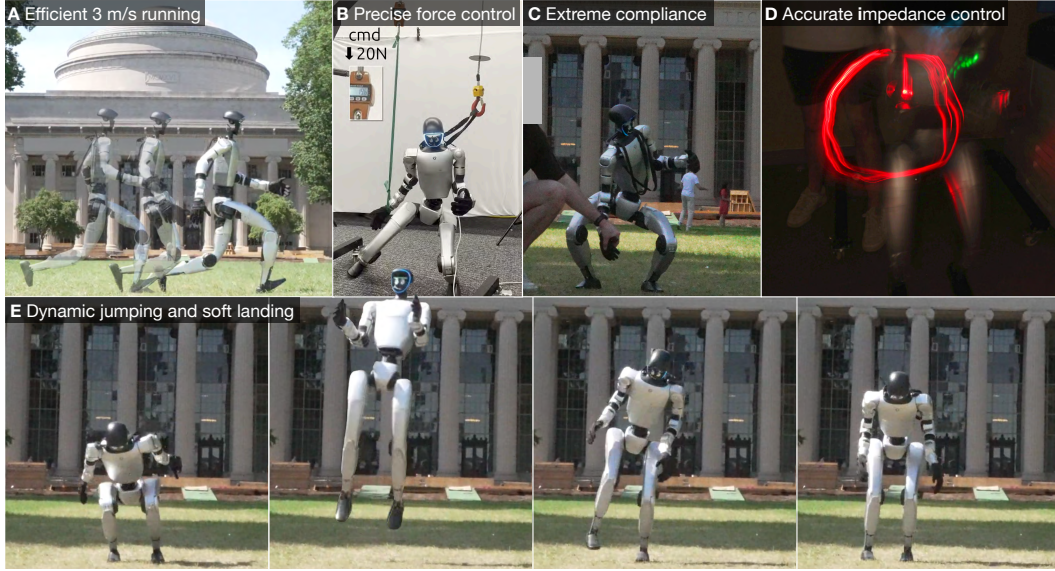


Figure 1: **Sim-to-real transfer of efficient, precise, and dynamic humanoid behaviors.** *Contact-UAN* enables residual world model learning from hardware trajectories with contact by inferring contact-consistent simulator reset states from onboard proprioception. The resulting residual world model constrains policy learning to actuator-feasible behaviors.

Abstract: Humanoid policies trained in simulation can transfer qualitatively to hardware while failing to transfer the force interactions learned during training, compromising torque efficiency, load regulation, and robustness to impacts. This failure can arise from actuator sim-to-real gaps because forces depend on actuator torques. We introduce *Contact-UAN*, a residual world model that preserves a simulator’s physics priors while correcting the actuator command channel to align simulated rollouts with recorded hardware rollouts. For humanoid motion with contact, this requires contact-consistent replay: if hardware playback in simulation begins with misaligned contact wrenches, the residual model learns to correct reset artifacts rather than actuator discrepancies. Using only onboard proprioception, *Contact-UAN* infers the *simulator-side stance-foot pose* needed to initialize replay with consistent contact forces. We show that a UAN trained on 6.5 minutes of real-world walking data generalizes to tasks that benefit from accurate force and impulse regulation and produces policies that transfer seamlessly to the real world, yielding 20% lower running torque, 43% lower force-tracking error, extreme compliance, accurate impedance control, and dynamic, robust jumps.

Keywords: Sim-to-Real Transfer, Humanoids, Residual World Models

1 Introduction

Humanoid robots can perform diverse tasks by virtue of their general-purpose morphology, and recent advances in learning-based control within physics simulators have enabled increasingly complex behaviors [1–4]. For sim-to-real humanoid control, qualitative motion transfer is only a coarse measure of success; the robot must also generate force profiles with the magnitude and timing needed to move efficiently, regulate loads, and absorb impacts. Policy optimization can exploit imperfect actuator dynamics, producing policies that may transfer qualitative motions but not force interactions, leading to excessive torques, poor commanded-force tracking, or failures during high-impulse phases (Section 4.2). Because these forces are mediated by joint torques, we study how improving the actuator model used during policy training reduces force-level sim-to-real gaps.

The actuator model fidelity required for sim-to-real transfer is task-dependent: simple models may suffice for quasi-static behaviors, while behaviors such as efficient running, force tracking, jumping, and landing depend on accurate actuator response, force generation, and impact absorption. Analytic simulators [5, 6] provide useful physical structure, yet they often approximate actuator effects such as torque saturation and load-dependent nonlinear friction, precisely the effects that shape ground-force generation, external-force regulation, and impact absorption [7, 8]. Domain randomization improves robustness by training over parameter ranges, but it can produce conservative policies that under-utilize hardware [9–12]. Online adaptation and SysID can recover performance by estimating dynamics parameters, but require choosing which simulator parameters to randomize and identify [1, 2, 13–17]. Fully learned world models (WMs) take another approach: fit dynamics directly from data and can, in principle, capture effects omitted by analytical simulators [18–21], but must model the full robot-environment dynamics needed for policy optimization. We therefore follow prior work in residual modeling and actuator learning [7, 22–27]: retain the physical priors of an existing simulator and learn only the sim-to-real gap in the actuator command channel.

Learning such an actuator-channel residual to improve downstream transfer requires a training signal. Supervised actuator networks use measured motor torques as this signal [7], but reliable output torque sensing is unavailable on many humanoid platforms. Without such labels, residual methods can use trajectory replay-based training instead: reset the simulator to a recorded hardware state, apply the recorded commands, and train residual commands so the simulated rollout matches the recorded hardware trajectory [8, 28]. Critically, this objective isolates modeling error only if replay begins from a hardware-equivalent simulator state. Otherwise, the residual model may learn to correct simulation reset artifacts rather than underlying actuator discrepancies.

Matching a given hardware state in simulation is a kinematics problem for suspended, free-space motions [8, 15]. However, such motion lacks the contact information needed to capture load-dependent actuator behavior during stance, force generation, impact, and landing. He et al. [28] collect data with contacts and use a motion capture system to provide ground-truth body poses. However, consider a humanoid robot standing still: on hardware, ground-reaction forces balance its weight, but initializing a *soft-contact* simulator with the feet exactly at the ground surface may not produce equivalent forces. Instead, matching the hardware force interaction requires introducing slight ground penetration via adjustments to the simulated foot height and orientation, not a precise height measurement. This is necessary because millimeter-scale foot-height and degree-scale orientation errors can substantially change simulated contact wrenches (Fig. 3). These observations point to a critical need to address contact-consistent state initialization for replay-based residual learning.

We introduce *Contact-UAN*, a residual world-modeling approach to training humanoid policies for force- and impulse-sensitive behaviors. The core contribution is contact-consistent state initialization from proprioception, which enables replay-based residual learning from hardware trajectories with contacts. From proprioception alone, *Contact-UAN* infers the simulator-side stance-foot pose, including the small ground penetration and tilt needed to initialize replay with contact wrenches consistent with the hardware trajectory, together with the base velocity needed to complete the reset state. With replay initialized in this contact-consistent state, the UAN residual can learn to correct true actuator modeling errors in simulation.

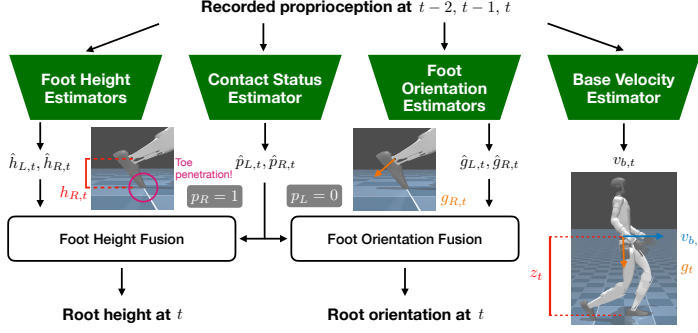


Figure 2: **Our proprioceptive state estimation framework.** Simulation supervision maps load-dependent proprioceptive cues to the simulator-side stance-foot pose consistent with them under the soft-contact model. Equations (4)–(6) fuse confident contacts into the root pose used for hardware replay in simulation.

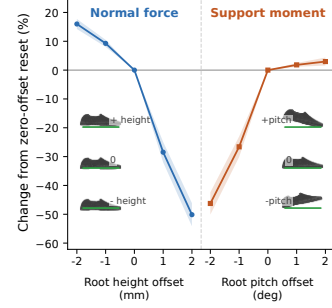


Figure 3: **Contact-wrench sensitivity.** A mm-scale height error substantially changes force, and a degree-scale pitch error substantially changes moment.

We evaluate Contact-UAN on a Unitree G1 using the state-of-the-art BeyondMimic sim-to-real recipe as the nominal simulator [2, 29], and compare against the same pipeline without UAN (Section 4). We choose downstream tasks that expose distinct actuator-sensitive force and impulse regimes: running requires torque-efficient ground-force generation, force control and impedance require accurate external-force regulation, and jumping requires takeoff impulse and landing absorption. A single UAN trained from 6.5 minutes of real-world walking data improves transfer across all of these behaviors. On held-out running replay, the UAN-augmented simulator reduces the aggregate torque-usage gap to hardware by 69.2%, showing that it gives policy learning a more accurate torque signal. Training in this corrected simulator yields a running policy that preserves velocity tracking while reducing measured hardware torque usage by 20% relative to the no-UAN baseline. The same residual reduces force-tracking error by 43% and enables robust jumping policies that fail without UAN (Section 4.2). Ablation experiments further show that Contact-UAN can close larger actuator gaps and is more data-efficient than fully learned WMs (Section 4.3).

2 Method

Contact-UAN has three stages (Fig. 4): relabel hardware data for contact-consistent replay initialization (Section 2.1); train the UAN residual (Section 2.2); and train downstream policies in the augmented simulator (Section 2.3). Only the task policy runs on hardware.

2.1 Contact-Consistent Replay Initialization

Accurate replay requires each window to begin with simulated contacts that reproduce hardware contact wrenches. Onboard sensing provides neither the contact wrench nor absolute base height, and reconstructing global pose is inadequate because soft-contact wrenches depend on simulator-specific, millimeter-scale foot penetration and tilt (Fig. 3). We therefore infer the *simulator-side* contact-consistent reset state from proprioception.

Contact affects joint motion and IMU response, making contact wrenches inferable from a short proprioceptive history [4, 30]. The state estimators assume flat terrain, enabling use of a root- xy and -yaw invariant frame. They predict contact probabilities, stance-foot pose, and base linear velocity from a K -step history of IMU measurements, joint states, and previous joint targets.

Although proprioception could be used to estimate contact force directly, a force estimate alone does not specify the simulator state used at reset. We therefore regress simulator-side stance-foot pose, learning in one step the map from load-dependent proprioception to the penetrating height and orientation that reproduce the contact wrenches. Initialization from this pose avoids separately inverting the simulator’s contact model. It is critical that absolute base height is omitted from the

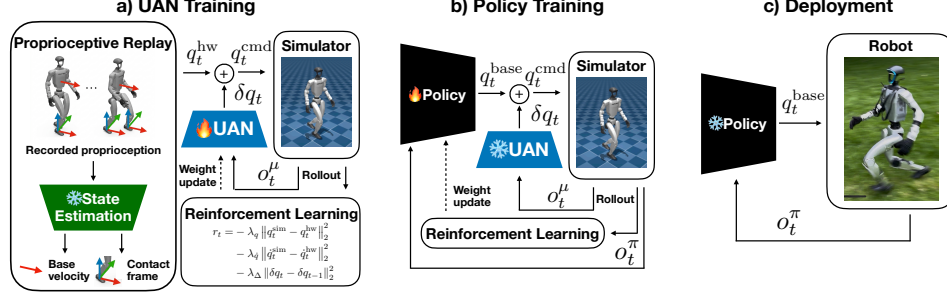


Figure 4: **Contact-UAN.** a) Train a UAN to add joint-position target corrections to reduce simulation transition error relative to recorded hardware data, with base velocities and contact poses labeled via proprioceptive state estimators. b) Train a task policy in the UAN-augmented simulator with no robot-specific domain randomization. c) Deploy only the learned policy on hardware.

observations: otherwise, the estimator can minimize its simulation loss by copying height rather than using load-dependent cues. This shortcut need not transfer because hardware height may differ from the simulator height that reproduces the same contact force.

We first estimate contact status and train the pose regressors only on stance samples. Swing poses largely map to zero contact wrench, so their variation dominates the objective without informing the contact model. In stance, however, small pose differences strongly affect the wrench: in our MuJoCo model (Table 4) [6], a +2 mm root-height offset reduces normal force by roughly 50%, while a -2° pitch offset changes the support moment by roughly 45% (Fig. 3). Stance conditioning therefore focuses the regression on the contact-sensitive variation needed at reset (Figs. 5 and 6).

Each confident stance-foot estimate yields a candidate root height and orientation through forward kinematics. We fuse candidates using contact-probability weights and discard windows with no confident contact; full equations are in Appendix A.4. Because the estimators are trained in simulation, their key transfer assumption is that load-dependent proprioceptive cues remain informative across domains. Figure 5A probes this assumption in controlled sim-to-sim: the stance estimator, trained in the base simulator, achieves low contact-height error in the hard target simulator despite actuator perturbations. Figure 5B–C provide complementary end-to-end evidence through improved hardware replay. The estimators are used only for offline relabeling and are absent from downstream policy training and deployment. Together with estimated base velocity, the fused root pose initializes contact-consistent UAN replay, enabling the UAN to learn to correct actuator discrepancies rather than simulator reset artifacts.

2.2 UAN as a Residual World Model

Let s_t denote the robot state and let a_t denote the action. The real robot transition is $f(s_t, a_t)$, while the baseline simulator transition is $f_{\text{sim}}(s_t, a_t)$. We learn a residual model μ that corrects the simulator input rather than the simulator state:

$$\min_{\mu} \mathbb{E}_{(s_t, a_t)} \left[\|f(s_t, a_t) - f_{\text{sim}}(s_t, a_t + \mu(o_t^\mu))\|_2^2 \right]. \quad (1)$$

In implementation, μ outputs a residual correction δq_t that is added to a base target q_t^{base} :

$$q_t^{\text{cmd}} = q_t^{\text{base}} + \delta q_t, \quad \delta q_t = \mu(o_t^\mu). \quad (2)$$

During UAN training, $q_t^{\text{base}} \equiv q_t^{\text{hw}}$ is the recorded replay target, and during downstream policy training, q_t^{base} is the policy output. The UAN therefore preserves the simulator’s rigid-body dynamics, actuator implementation, contact solver, and numerical integration; it only modulates the command that those existing simulation components integrate forward.

UAN observations and architecture. The UAN observes joint-local tracking signals over a short history. For each joint, the observation history contains the target-position error and joint velocity,

$$o_t^\mu = [(q^{\text{base}} - q)_{t-H_\mu+1:t}, \dot{q}_{t-H_\mu+1:t}]. \quad (3)$$

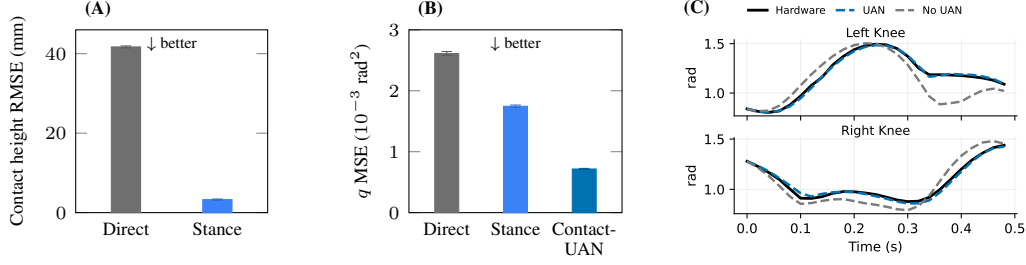


Figure 5: **State estimation ablation and UAN fit.** (A) compares contact-height estimates in simulation of an estimator trained on *both stance and swing* data (direct) versus *only stance* data (stance, ours). Both estimators were trained on base simulation data and evaluated in the hard target environment (Sec. 3). Panel (B) reports 24-step hardware replay joint-position MSE on WTW training data, and (C) shows hardware replay knee trajectories on running validation data.

This compact observation limits the residual model to local actuator-tracking information to avoid overfitting [7, 8]. In our experiments we set $H_\mu = 3$ (60 ms).

To improve data efficiency, we share network parameters across joints with identical actuator hardware [7, 8]. Each actuator type has its own two-layer MLP with hidden dimensions (128, 128) and ELU activations. This enforces a hardware-structured residual model while having enough capacity to model actuator-specific nonlinearities.

Data collection. We collect data by executing a Walk-These-Ways-style policy [31] on the robot (details in Appendices), enabling changes in gait frequency and swing height to produce trajectories with diverse stance durations, swing speeds, impacts, and load-dependent tracking errors. We record onboard proprioception: measured base angular velocity, projected gravity, joint positions and velocities, and commanded joint targets.

UAN task design. The UAN is trained in a replay environment built from labeled hardware windows via the PPO algorithm [32]. At reset, each simulated robot is initialized from a randomly sampled valid replay window. We use a plane terrain, 4096 parallel environments, a 24-step (480 ms) replay horizon, and PPO rollouts of the same length. The reward penalizes joint-position MSE, joint-velocity MSE, and residual-rate magnitude. This replay task encourages the UAN to learn corrections that make simulated joint transitions match hardware while discouraging unnecessarily high-frequency residual commands to aid with exploration.

2.3 Policy Training and Deployment

After UAN training, we freeze its weights and insert it into downstream task training as a deterministic residual correction. We remove all robot-specific DR, but still randomize the terrain mesh, ground friction, and external pushes to encourage robust, deployable behaviors. The task policy outputs joint-position targets, but the simulator receives the UAN-corrected target from Eq. (2). This exposes the downstream policy to actuation behavior closer to the hardware, so high-reward behaviors must remain feasible under the corrected actuator response. At deployment, only the learned task policy is run on the robot.

3 Experimental Setup

We evaluate Contact-UAN on a 29-DoF Unitree G1 humanoid, which has a small sim-to-real gap relative to larger commercially available humanoids such as the Unitree H1/H1-2, due to its quasi-direct-drive actuators and low-inertia limbs. The nominal simulator follows the BeyondMimic/mjlab G1 sim-to-real recipe [2, 29], including its PD gains and identified armature values, and serves as our strong baseline when trained without the UAN.

To train the UAN, we deploy a Walk-These-Ways-style policy [31] on hardware for 6.5 minutes while varying commanded speed, gait frequency, and swing height. We record onboard propriocep-

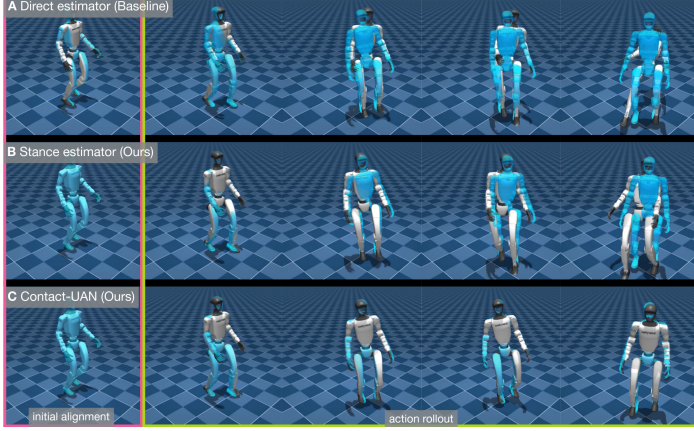


Figure 6: **Open-loop rollouts.** We set the initial simulator state to the estimated state and rollout recorded hardware actions. Our estimated simulator reset state is in blue. (A) uses a foot height estimator trained on *both stance and swing* data; (B) uses our estimator trained on *only stance* data to align initial contacts; (C) uses our estimator and the UAN to close the actuation gap.

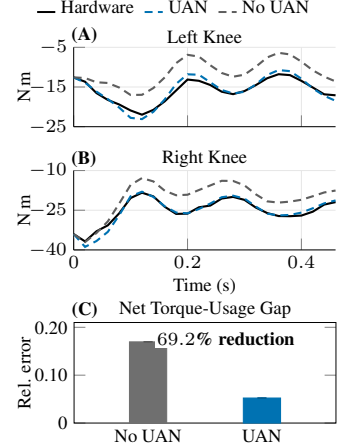


Figure 7: **Torque replay gap.** Contact-UAN better matches held-out left (A) and right (B) knee torques and reduces aggregate all-joint net torque-usage mismatch by 69.2% (C).

tion: base angular velocity, projected gravity, joint positions and velocities, and commanded joint targets; then relabel each hardware window with base velocity and pose following Section 2.1.

For downstream evaluation, we train running, force-control, and jumping policies in the frozen UAN-augmented simulator with no robot-specific DR, and compare against the same tasks trained in the nominal simulator. These tasks are chosen because their performance benefits from accurate actuation while the robot exchanges forces with the environment: running benefits from efficient ground-force generation, force control and impedance from precise external force regulation, and jumping from impulse generation and impact absorption. Running follows the mjlab G1 velocity-tracking task [29]; force control adapts legged force-control objectives [30]; and jumping follows a BeyondMimic-style motion-tracking setup [2], followed by fine-tuning for higher jumps and softer landings. Details for all tasks are in the Appendix.

For controlled sim-to-sim evaluation, we construct easy, medium, and hard target simulators by scaling deterministic actuator perturbations to damping, friction loss, action delay, encoder bias, and armature; exact values are in Appendix E. We also compare against RWM-U-style fully learned offline WMs [33], trained from scratch or fine-tuned from the base policy using 33 hours of target transitions, $400\times$ more than the UAN training data; details are in Appendix F.

4 Results

4.1 Residual World Model Accuracy

Does our estimation pipeline accurately recover foot penetration? We evaluate the contact-pose estimator in the hard sim-to-sim setting, where ground-truth contact pose is available. Fig. 5A shows that directly regressing foot height over all gait phases yields large errors, because swing-foot variation dominates the supervised objective. In contrast, our contact-conditioned estimator, which first identifies contact status and then estimates pose only for confident contacts (at stance), reduces contact-height RMSE by an order of magnitude. This shows that the proposed estimation pipeline provides sufficiently accurate contact initialization for replay-based residual learning.

Does Contact-UAN reduce the sim-to-real gap? Relabeling alone improves over direct initialization, but the remaining error reflects actuator mismatch between the nominal simulator and the robot. Adding the UAN trained from WTW data further reduces 24-step joint-position MSE (Fig. 5B) and produces knee trajectories that closely match hardware on held-out running data (Fig. 5C). We visu-

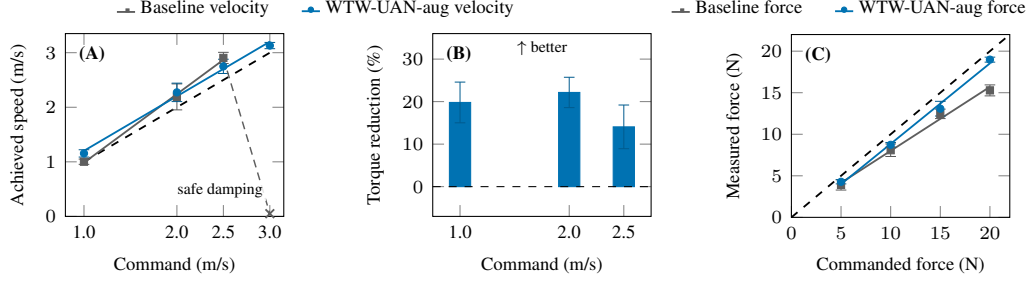


Figure 8: **Improved running and force control sim-to-real transfer.** UAN running policy retains tracking ability (A) while improving efficiency (B). The baseline marker at 3.0 m s^{-1} denotes protection-triggered safe damping rather. UAN force policy reduced average force tracking error by $42.8 \pm 12.1\%$ (C). Both baselines were trained with BeyondMimic-style SysId [2].

alize the effects of ablating the UAN and our estimation approach when doing open-loop playback of hardware data in Fig. 6. To directly measure the actuator sim-to-real gap, we replay the same held-out running hardware commands in the nominal simulator and the UAN-augmented simulator, using the same relabeled initial states. We define aggregate torque usage as $U(\tau) = \sum_{t,j} |\tau_{t,j}|$ over the replay horizon, and report the relative error $|U(\tau^{sim}) - U(\tau^{hw})| / U(\tau^{hw})$. The UAN-augmented simulator better matches hardware left- and right-knee torques and reduces aggregate all-joint net torque-usage mismatch by 69.2% relative to the nominal simulator (Fig. 7). Because both simulators use identical commands and initial states, the reduced torque mismatch reflects improved actuator modeling on motions with contacts.

4.2 Sim-to-Real Task Transfer

Does Contact-UAN improve torque efficiency? We compare the baseline and UAN policy at commanded forward speeds of 1.0, 2.0, and 2.5 m s^{-1} . For each policy and command, we run five hardware trials over a fixed 30 ft course. We compute achieved speed from the traversal time and log joint torques over the same interval to measure torque usage. The UAN policy preserves velocity tracking while reducing torque usage relative to the baseline policy (Fig. 8A,B). Since both policies achieve similar speeds, the torque reduction reflects improved efficiency rather than reduced task execution. The UAN policy also transfers to a 3.0 m s^{-1} command, achieving $3.13 \pm 0.06 \text{ m s}^{-1}$ on hardware (Fig. 1A). In contrast, the baseline policy triggers motor-overheating protection and enters safe-damping mode after a few seconds of tracking this command.

Does Contact-UAN improve force control precision? We evaluate end-effector force tracking using a dynamometer. For each policy and commanded force, we perform four hardware trials, increasing the commanded downward force in increments of 5 N. The robot is connected to the dynamometer through an exercise band, requiring whole-body coordination rather than solely arm motion to realize the commanded force (Fig. 1B). The UAN force policy reduces overall force-tracking error by $42.8 \pm 12.1\%$ relative to the baseline (Fig. 8C). The improvement is largest at higher commanded forces, where the robot must bend its legs and coordinate lower-body posture to generate the desired force, increasing sensitivity to actuator-modeling errors. At zero commanded force, the UAN policy is qualitatively *compliant*: the end-effector can be pulled in different directions while the body flexibly yields rather than resisting (Fig. 1C). We also compose the force-control policy with a proprioceptive end-effector estimator and PD outer loop to perform *impedance position control*. The resulting controller tracks a circular trajectory using onboard sensing alone (Fig. 1D and Fig. 12), suggesting that Contact-UAN’s force-control gains transfer to higher-level compliant interaction behaviors; details are in Appendix D.2.

Does Contact-UAN enable robust, dynamic motions? We further evaluate whether the same residual world model supports highly dynamic motions with flight and impact. We train a jumping policy in two stages: first tracking a reference jumping motion [2], then fine-tuning to increase jump height while reducing landing impact. A baseline policy trained without the UAN residual

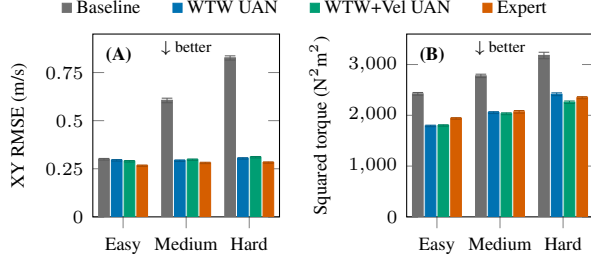


Figure 9: **Sim-to-sim velocity policy transfer.** Policies trained with UAN reach expert-level performance. Adding task-specific data makes a negligible difference, showing WTW data is sufficient.

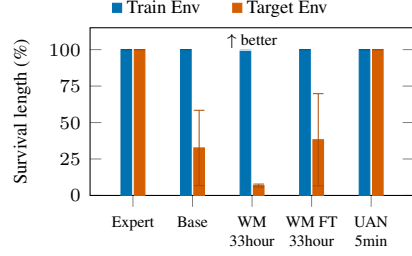


Figure 10: **Data efficiency of UAN.** UAN avoids falling with only 5 minutes of target data, while fully-learned WMs struggle even with $400\times$ more data.

correction fails to land safely on hardware, indicating that jumping is sensitive to actuator-modeling errors during takeoff and touchdown. In contrast, the UAN-trained policy transfers successfully to the real robot (Fig. 1E and Fig. 11): after fine-tuning, it jumps higher by squatting deeper and generating larger ground-reaction forces, while landing softly and returning to a stable stance. The policy can also execute consecutive jumps and recover from external disturbances; additional video-frame sequences and implementation details are provided in Appendix D.3.

4.3 Sim-to-Sim Ablation

Can Contact-UAN overcome larger sim-to-real gaps? To test whether the UAN is effective when the nominal simulator is substantially inaccurate, we evaluate transfer in sim-to-sim experiments with progressively larger deterministic actuator perturbations. The WTW UANs were trained on 5 minutes of WTW data, while the WTW+Vel UAN was trained on 5 minutes of WTW data and 5 minutes of running data. Across the easy, medium, and hard target simulators, the baseline policy degrades sharply as the gap increases, while policies trained with the UAN achieve expert-level tracking error and torque usage (Fig. 9). Adding task-specific running data to the UAN training set provides little additional benefit over the WTW data alone, suggesting that this walking dataset is sufficient to learn a residual model that generalizes to running. These results indicate that the UAN can compensate for larger actuator-modeling errors than those observed on hardware.

Is Contact-UAN more data-efficient than a fully learned world model? To evaluate data efficiency, we study sim-to-sim transfer of locomotion policies trained with different WMs. A policy trained in the base physics simulator (analytic WM) collects data in the hard target environment for training each WM. As shown in Fig. 10, just 5 minutes of the hard target transitions suffices for the UAN to bridge the dynamics gap and match the expert policy’s survival length in the target environment. In contrast, fully learned offline WMs, from scratch (WM 33hour) or fine-tuned from the base policy (WM FT 33hour), yield poor performance even with $400\times$ more data. This data-efficiency gap highlights the advantage of the UAN’s physics priors: by correcting only the residual actuation gap rather than learning dynamics from scratch, it generalizes reliably, whereas fully learned WMs struggle even though all policies achieve high performance in their own training environments.

5 Limitations

Our current Contact-UAN does not capture coupled actuator effects or motor-temperature-dependent effects [34]. Both are small in practice and often neglected. Coupled actuator effects could be introduced by using a single model that observes the whole body or through cross-attention. Temperature effects are harder to identify from short datasets because motor temperature evolves slowly relative to the control loop. Capturing them may also require heat modeling or additional sensors.

6 Related Work

System Identification and Domain Randomization. System identification estimates inertial and dynamic parameters from robot data, traditionally via least-squares regression on rigid-body dynamics [35, 36], with recent geometric, convex, differentiable, and exploration-based variants improving robustness and excitation [37–41]. Other approaches identify simulators as black boxes using trajectory-matching [15] or distribution-matching objectives [42]. These methods remain limited by the structure of the chosen parametric model. Domain randomization instead trains over parameter distributions [9, 10, 43–46], but wide randomization can yield conservative or inefficient policies [11, 12]. Online adaptation mitigates this by inferring latent dynamics at deployment [13, 14, 47–49]. Recent humanoid systems have therefore moved toward less randomization and more accurate models [1, 2, 15]. Our work follows this direction by replacing actuator randomization with a residual actuator correction learned from proprioceptive hardware data.

Robotic World Models. Fully learned world models replace hand-designed simulators with dynamics models trained from robot data. Dyna-style approaches alternate data collection, model learning, and policy optimization through imagined rollouts [20, 21, 50], while RWM and RWM-U extend this paradigm to legged robots in online [19] and fully offline settings [33]. Such models reduce the structural bias of parametric simulators but require large datasets and mechanisms to prevent exploitation outside the data distribution [18]. We instead learn a residual world model: the simulator retains rigid-body dynamics, contact, and integration, while learning is restricted to the actuator channel. This physics prior reduces required model capacity and mitigates reward hacking.

Residual Models and Actuator Networks. Residual learning combines analytical physics with learned corrections across object, system, simulator, and calibration levels [22–27, 51]. At the actuator level, supervised actuator networks predict output torques from tracking histories and have enabled effective sim-to-real transfer [7], but require torque sensing; current-based alternatives reduce sensing requirements but cannot capture all gearbox effects [39]. UAN removes torque supervision by learning residual actions that match real-to-sim joint transitions [8], but was previously demonstrated only on a quadruped’s arm. Closely related humanoid methods such as ASAP and MOSAIC also learn residual corrections, but rely on motion capture or target interface-specific gaps rather than reusable actuator dynamics [28, 52]. We extend UAN to a floating-base 29-DoF humanoid by using proprioceptive state estimators for base velocity and contact pose, enabling contact-aligned residual model learning without external sensing.

7 Conclusion

We present *Contact-UAN*, a residual world-modeling approach for humanoid control that preserves the structure of an analytical simulator while learning an actuator-level correction from offline proprioceptive data with state estimation. To make replay-based residual learning possible for humanoid motion with contacts with minimal sensing, we estimate base velocity, contact status, and contact pose from onboard observations, allowing hardware trajectories to be initialized in simulation with consistent contact forces. On a humanoid platform, our UAN-augmented simulator enables sim-to-real transfer of whole-body control policies. Using only 6.5 minutes of real-world walking data for training, the UAN generalizes across diverse tasks to deliver 20% more torque-efficient running, 40% more precise force control, extreme compliance, accurate impedance control, and dynamic, robust jumps. Sim-to-sim experiments further show that the same residual world model remains effective under larger controlled actuator gaps. These results suggest that residual world models provide a data-efficient alternative to fully learned dynamics models, retaining the generalization benefits of physics simulation while improving fidelity where the sim-to-real gap is most consequential.

Acknowledgments

We thank the members of the Improbable AI lab, especially Venkateshkrishna Venganallore Par-suram, Michelle Wang, and Ryan Bahlous-Boldi for supporting hardware experiments. We also thank Patrick Wensing for helpful discussions. The research was partly supported by the Ministry of Trade, Industry, and Energy (MOTIE), Korea, under the "Global Industrial Technology Cooperation Center program" supervised by the Korea Institute for Advancement of Technology (KIAT) under Grant No. P0028435 and the National Science Foundation Graduate Research Fellowship under Grant No. 2141064. This work was also partly supported by the Sony Research Award.

References

- [1] J. P. Sleiman, H. Li, A. Adu-Bredu, R. Deits, A. Kumar, K. Bergamin, M. Bhardwaj, S. Biddlestone, N. Burger, M. A. Estrada, et al. Zest: Zero-shot embodied skill transfer for athletic robot control. *arXiv preprint arXiv:2602.00401*, 2026.
- [2] Q. Liao, T. E. Truong, X. Huang, Y. Gao, G. Tevet, K. Sreenath, and C. K. Liu. Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion. *arXiv preprint arXiv:2508.08241*, 2025.
- [3] Z. Wu, X. Huang, L. Yang, Y. Zhang, X. Chen, P. Abbeel, R. Duan, A. Kanazawa, C. Sferrazza, G. Shi, and C. K. Liu. Perceptive humanoid parkour: Chaining dynamic human skills via motion matching, 2026. URL <https://arxiv.org/abs/2602.15827>.
- [4] G. B. Margolis, M. Wang, N. Fey, and P. Agrawal. Softmimic: Learning compliant whole-body control from examples. *arXiv preprint arXiv:2510.17792*, 2025.
- [5] NVIDIA. Isaac Sim. URL <https://github.com/isaac-sim/IsaacSim>.
- [6] E. Todorov, T. Erez, and Y. 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.
- [7] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), Jan. 2019. ISSN 2470-9476. doi:10.1126/scirobotics.aau5872. URL <http://dx.doi.org/10.1126/scirobotics.aau5872>.
- [8] N. Fey, G. B. Margolis, M. Peticco, and P. Agrawal. Bridging the sim-to-real gap for athletic loco-manipulation. In *Robotics: Science and Systems (RSS)*, 2025. URL <https://openreview.net/forum?id=0HwPNCqQft>.
- [9] 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, 2017.
- [10] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters. Robot learning from randomized simulations: A review. *Frontiers in Robotics and AI*, 9:799893, 2022.
- [11] Z. Xie, X. Da, M. Van de Panne, B. Babich, and A. Garg. Dynamics randomization revisited: A case study for quadrupedal locomotion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4955–4961. IEEE, 2021.
- [12] L. Campanaro, S. Gangapurwala, W. Merkt, and I. Havoutis. Learning and deploying robust locomotion policies with minimal dynamics randomization. In *6th Annual Learning for Dynamics & Control Conference*, pages 578–590. PMLR, 2024.
- [13] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems (RSS)*, 2021.

- [14] M. Liu, D. Pathak, and A. Agarwal. Locoformer: Generalist locomotion via long-context adaptation. In *Conference on Robot Learning*, pages 532–546. PMLR, 2025.
- [15] F. Bjelonic, F. Tischhauser, and M. Hutter. Towards bridging the gap: Systematic sim-to-real transfer for diverse legged robots. *arXiv preprint arXiv:2509.06342*, 2025.
- [16] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning, 2022. URL <https://arxiv.org/abs/2205.02824>.
- [17] G. B. Margolis, X. Fu, Y. Ji, and P. Agrawal. Learning to see physical properties with active sensing motor policies, 2023. URL <https://arxiv.org/abs/2311.01405>.
- [18] B. Hou, G. Li, J. Jia, T. An, X. Guo, S. Leng, H. Geng, Y. Ze, T. Harada, P. Torr, et al. World model for robot learning: A comprehensive survey. *arXiv preprint arXiv:2605.00080*, 2026.
- [19] C. Li, A. Krause, and M. Hutter. Robotic world model: A neural network simulator for robust policy optimization in robotics. *arXiv preprint arXiv:2501.10100*, 2025.
- [20] N. Hansen, J. SV, V. Sobal, Y. LeCun, X. Wang, and H. Su. Hierarchical world models as visual whole-body humanoid controllers. In *International Conference on Learning Representations*, volume 2025, pages 62175–62195, 2025.
- [21] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR, 2023.
- [22] J. Gruenstein, T. Chen, N. Doshi, and P. Agrawal. Residual model learning for microrobot control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7219–7226. IEEE, 2021.
- [23] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020.
- [24] F. Golemo, A. A. Taiga, A. Courville, and P.-Y. Oudeyer. Sim-to-real transfer with neural-augmented robot simulation. In *Conference on Robot Learning*, pages 817–828. PMLR, 2018.
- [25] J. Gao, M. Y. Michelis, A. Spielberg, and R. K. Katzschmann. Sim-to-real of soft robots with learned residual physics. *IEEE Robotics and Automation Letters*, 9(10):8523–8530, 2024.
- [26] Y. Jiang, T. Zhang, D. Ho, Y. Bai, C. K. Liu, S. Levine, and J. Tan. Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2884–2890. IEEE, 2021.
- [27] Y. Yu and L. Liu. Neural fidelity calibration for informative sim-to-real adaptation. *arXiv preprint arXiv:2504.08604*, 2025.
- [28] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan, et al. Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills. In *Robotics: Science and Systems (RSS)*, 2025.
- [29] K. Zakka, Q. Liao, B. Yi, L. L. Lay, K. Sreenath, and P. Abbeel. mjlal: A lightweight framework for gpu-accelerated robot learning. *arXiv preprint arXiv:2601.22074*, 2026.
- [30] T. Portela, G. B. Margolis, Y. Ji, and P. Agrawal. Learning force control for legged manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15366–15372. IEEE, 2024.
- [31] G. B. Margolis and P. Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *Conference on Robot Learning*, pages 22–31. PMLR, 2023.

- [32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.
- [33] C. Li, A. Krause, and M. Hutter. Uncertainty-aware robotic world model makes offline model-based reinforcement learning work on real robots. *arXiv preprint arXiv:2504.16680*, 2025.
- [34] D. Müller, E. Knoop, D. Mylonopoulos, A. Serifi, M. A. Hopkins, R. Grandia, and M. Bächer. Olaf: Bringing an animated character to life in the physical world. *IEEE Robotics and Automation Letters*, 2026.
- [35] P. K. Khosla and T. Kanade. Parameter identification of robot dynamics. In *1985 24th IEEE conference on decision and control*, pages 1754–1760. IEEE, 1985.
- [36] C. G. Atkeson, C. H. An, and J. 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>.
- [37] T. Lee, P. M. Wensing, and F. 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.
- [38] P. M. Wensing, G. Niemeyer, and J.-J. E. Slotine. A geometric characterization of observability in inertial parameter identification. *The International Journal of Robotics Research*, 43(14):2274–2302, 2024. doi:10.1177/02783649241258215. URL <https://doi.org/10.1177/02783649241258215>.
- [39] L. Schwendeman, A. SaLoutos, E. Stanger-Jones, and S. 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] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier. Differentiable simulation for physical system identification. *IEEE Robotics and Automation Letters*, 6(2):3413–3420, 2021.
- [41] N. Sobanbabu, G. He, T. He, Y. Yang, and G. Shi. Sampling-based system identification with active exploration for legged sim2real learning. In *9th Annual Conference on Robot Learning*, 2025.
- [42] A. Miller, F. Yu, M. Brauckmann, and F. Farshidian. High-performance reinforcement learning on spot: Optimizing simulation parameters with distributional measures, 2025. URL <https://arxiv.org/abs/2504.17857>.
- [43] 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 (RSS)*, 2018.
- [44] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47), Oct. 2020. ISSN 2470-9476. doi:10.1126/scirobotics.abc5986. URL <http://dx.doi.org/10.1126/scirobotics.abc5986>.
- [45] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2021.
- [46] J. R. Cha, W. Cha, J. Shin, D. Kim, and J. Park. Sim-to-real of humanoid locomotion policies via joint torque space perturbation injection. *arXiv preprint arXiv:2504.06585*, 2025.

- [47] G. Ji, J. Mun, H. Kim, and J. 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.
- [48] I. M. A. Nahrendra, B. Yu, and H. 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.
- [49] J. Long, Z. Wang, Q. Li, J. Gao, L. Cao, and J. Pang. Hybrid internal model: Learning agile legged locomotion with simulated robot response, 2024. URL <https://arxiv.org/abs/2312.11460>.
- [50] R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.
- [51] N. Sontakke, H. Chae, S. Lee, T. Huang, D. W. Hong, and S. 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.
- [52] Z. Sun, B.-S. Huang, Y. Peng, X. Li, J. Ma, Y. Sun, Z. Li, H. Jiang, B. Gao, Z. Bing, et al. Mosaic: Bridging the sim-to-real gap in generalist humanoid motion tracking and teleoperation with rapid residual adaptation. *arXiv preprint arXiv:2602.08594*, 2026.
- [53] C. Schwarke, M. Mittal, N. Rudin, D. Hoeller, and M. Hutter. Rsl-rl: A learning library for robotics research. *arXiv preprint arXiv:2509.10771*, 2025.
- [54] Y.-H. Shin, T.-G. Song, G. Ji, and H.-W. Park. Actuator-constrained reinforcement learning for high-speed quadrupedal locomotion, 2023.
- [55] Carnegie Mellon University. CMU graphics lab motion capture database. <http://mocap.cs.cmu.edu>.

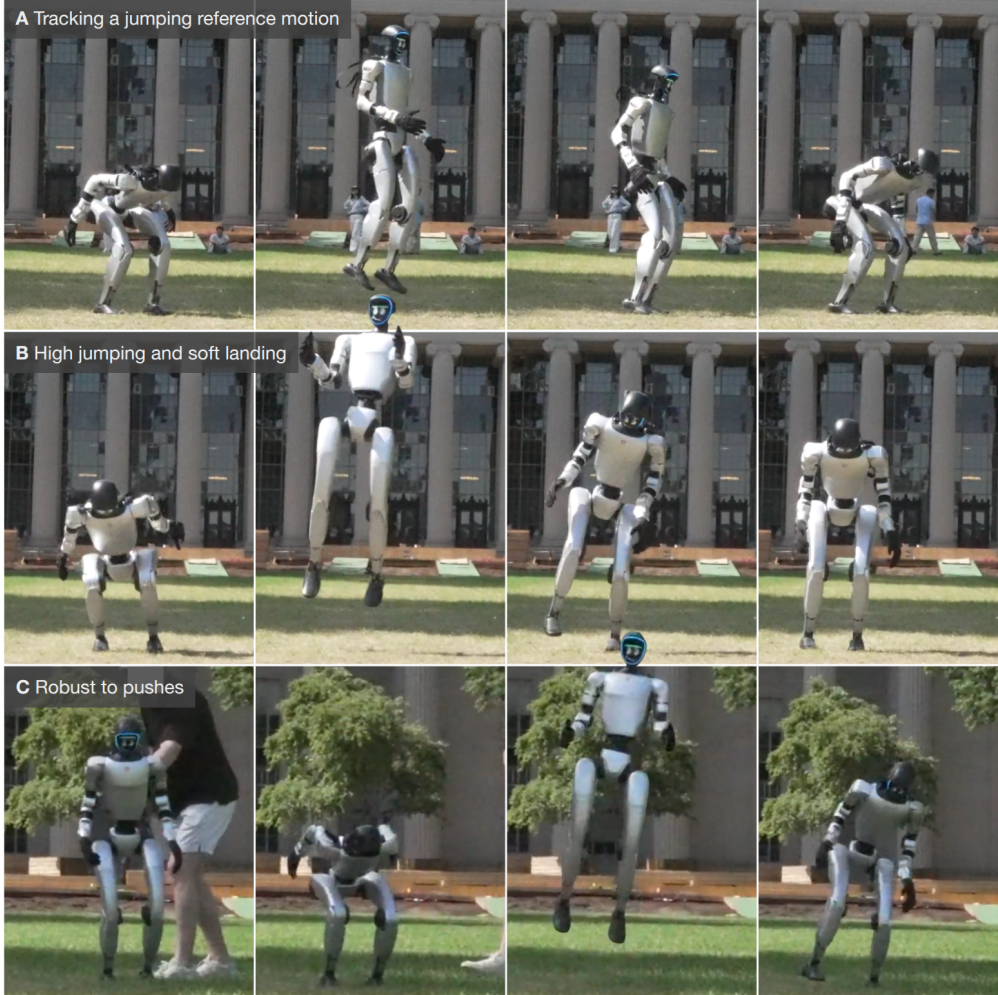


Figure 11: **Dynamic and robust jumping policies.** With the sim-to-real gap bridged by UAN, we demonstrate more challenging behaviors. The policy is first trained to track a jumping reference motion (A), then fine-tuned to maximize jump height while reducing landing impact (B). It also remains robust to external disturbances (C).

A State Estimation

A.1 Data Collection

We collect data with the WTW-style data collection policy in its training environment (Appendix C), but on flat terrain with contact model parameters identical to those used in UAN training (Table 4), for 20 simulation seconds with 4096 environments. The dataset is split 90%/10% into training and validation samples by holding out a random 10% of the environments.

A.2 Architectures

Table 1 summarizes the estimator architectures and shared optimization settings. Foot-height and foot-projected-gravity estimators are trained separately for left-stance and right-stance samples.



Figure 12: **Impedance position control.** Combining a precise force control policy with a proprioceptive state estimator, we develop an impedance controller for closed-loop end-effector position tracking. The controller tracks a circular trajectory reasonably well without any external sensing.

Table 1: State-estimator architectures. All estimators use ELU activations, Adam, learning rate 3×10^{-4} , batch size 4096, history length 3, and 200 epochs.

Setting	Base velocity	Contact status	Foot height	Foot proj. gravity
Output dimension	3	2	1 per foot	3 per foot
Hidden layers	(256, 256, 256)	(128, 128)	(256, 256, 256)	(256, 256, 256)
Loss	MSE	BCE-logits	MSE	MSE

A.3 Observation Noise

The simulated observation-noise model used for estimator-data collection is listed in Table 2. The same noise model is used when training the data-collection policy and the downstream policies except the jumping task, which uses a smaller joint-velocity noise (Table 9).

Table 2: Observation noise used when collecting simulated data (uniform additive).

Observation	Noise range
Base angular velocity	± 0.2 rad/s
Projected gravity	± 0.05
Joint position	± 0.01 rad
Joint velocity	± 1.5 rad/s
Previous target	none

A.4 Contact-Weighted Root-Pose Fusion

We initialize replay from a root pose whose simulated contacts reproduce the support wrench implied by the hardware trajectory. Since contact predictions are uncertain on hardware, we fuse pose estimates from all confident contacts rather than selecting a single stance foot.

Let $\hat{p}_{i,t} \in [0, 1]$ be the estimated contact probability for foot $i \in \{L, R\}$, and let p_{th} be the contact-confidence threshold. We define contact weights

$$w_{i,t} = \hat{p}_{i,t} \mathbf{1}[\hat{p}_{i,t} \geq p_{\text{th}}], \quad \alpha_{i,t} = \frac{w_{i,t}}{\sum_j w_{j,t} + \varepsilon}. \quad (4)$$

States with no confident contact are excluded from the UAN initial-state buffer, since the stance-foot pose estimates cannot define a contact-consistent root pose.

Each confident contact provides an independent estimate of root height. Let $\hat{h}_{i,t}$ be the estimated height of foot i relative to the terrain, and let $h_{i,t}^{\text{fk}}(z=0)$ be the foot height from forward kinematics when the root height is set to zero. The root height implied by foot i is

$$\hat{z}_t^{(i)} = \hat{h}_{i,t} - h_{i,t}^{\text{fk}}(z=0), \quad \hat{z}_t = \sum_i \alpha_{i,t} \hat{z}_t^{(i)}. \quad (5)$$

We recover root orientation using the same contact-weighted fusion. The orientation estimator predicts projected gravity in each stance-foot frame, $\hat{g}_{i,t}^{F_i}$. Forward kinematics maps each prediction to a root-frame candidate,

$$\hat{g}_t^{(i)} = \text{normalize}\left(R_{R \leftarrow F_i}(q_t) \hat{g}_{i,t}^{F_i}\right), \quad \hat{g}_t = \text{normalize}\left(\sum_i \alpha_{i,t} \hat{g}_t^{(i)}\right). \quad (6)$$

On flat terrain, this projected-gravity estimate determines the roll and pitch used for replay initialization in the yaw-invariant local frame. Together with the fused root height $\hat{z}_t$ and the separately estimated base velocity, the resulting relabeled hardware state initializes replay with contact-consistent support forces.

B UAN Training Details

Table 3 lists the five G1 actuator groups over which the UAN shares its per-group actor MLPs (Sec. 2.2). Table 4 lists relevant contact model parameters used in our mjlal simulation [29].

Table 3: Unitree G1 actuator groups used by the shared UAN architecture.

Actuator group	Joints
7520_14	Hip pitch, hip yaw, waist yaw
7520_22	Hip roll, knee
5020_2x	Waist roll, waist pitch, ankle pitch, ankle roll
5020	Shoulder pitch, shoulder roll, shoulder yaw, elbow, wrist roll
4010	Wrist pitch, wrist yaw

Table 4: MuJoCo contact parameters used in the contact-pose sensitivity diagnostic.

Element	condim	friction	solref	solimp
Foot collision geoms	3	[0.6, 0.005, 0.0001]	[0.02, 1]	[0.9, 0.95, 0.001, 0.5, 2]
Terrain/ground geom	3	[1, 0.005, 0.0001]	[0.02, 1]	[0.9, 0.95, 0.001, 0.5, 2]

Each replay episode corresponds to one labeled hardware window: 3 recorded steps prime the UAN observation history, after which the simulator rolls the window’s 24 recorded joint-position targets forward with the UAN residual applied. Windows are sampled only where the relabeled initial state has at least one confident foot contact (contact probability ≥ 0.8), and the residual correction is clipped to ± 1.0 rad. The UAN replay reward terms are listed in Table 6; PPO settings for the UAN, the data-collection policy, and all downstream task policies are consolidated in Table 5. We use the RSL-RL library [53].

C Data Collection Policy Training Details

The data-collection policy is a Walk-These-Ways-style velocity-tracking policy with behavior-shaping commands [31]. In addition to the base velocity command ($v_x^{\text{cmd}} \in (-1.0, 1.0)$ m/s, $\omega_z^{\text{cmd}} \in (-1.0, 1.0)$ rad/s, no lateral command), the policy receives a gait-frequency command in (1.0, 2.5) Hz and a swing-height command in (0.05, 0.1) m; all commands are resampled every 5–10 s. Varying these commands is what produces the diverse stance durations, swing speeds, and impacts of the UAN training data (Sec. 2.2). In addition to standard velocity-tracking and regularization terms, training adds gait-shaping penalties: a gait-phase contact schedule (−0.5), a swing-height tracking term (−20.0), and a Raibert-heuristic foot-placement term (−5.0, stance width 0.237 m). The policy uses a 3-step observation history and is trained on procedurally generated bumpy terrain; PPO settings are in Table 5.

Table 5: PPO hyperparameters for the UAN, the WTW data-collection policy, and the downstream task policies. Bold rows highlight per-task differences; all other optimization settings are shared.

Parameter	UAN	WTW	Running	Force-control	Jumping
<i>Optimization</i>					
Parallel environments	4096	4096	4096	4096	4096
Steps per environment	24	24	24	24	24
Learning epochs	5	5	5	5	5
Mini-batches	4	4	4	4	4
Learning rate		10^{-3} , adaptive (target KL 0.01) for all			
Max gradient norm	1.0	1.0	1.0	1.0	1.0
<i>PPO objective</i>					
Clip parameter	0.2	0.2	0.2	0.2	0.2
Entropy coefficient	0	0.002	0.01	0.002	0.005
Value-loss coefficient	1.0	1.0	1.0	1.0	1.0
Use clipped value loss	yes	yes	yes	yes	yes
Discount γ	0.95	0.99	0.99	0.99	0.99
GAE λ	0.95	0.95	0.95	0.95	0.95
<i>Networks (ELU activations)</i>					
Actor MLP	per-group (128, 128)	(512, 512, 256, 128)	(512, 256, 128)	(512, 512, 256, 128)	(512, 256, 128)
Critic MLP	(512, 512, 512, 512)	(512, 512, 512, 512)	(512, 256, 128)	(512, 512, 512, 512)	(512, 256, 128)
Initial action std.	0.3	1.0	1.0	1.0	1.0

Table 6: UAN replay reward weights.

Term	Math	Weight
joint_position_mse	$\frac{1}{n_q} \sum_j (q_{t,j}^{\text{sim}} - q_{t,j}^{\text{hw}})^2$	-1000
joint_velocity_mse	$\frac{1}{n_q} \sum_j (\dot{q}_{t,j}^{\text{sim}} - \dot{q}_{t,j}^{\text{hw}})^2$	-0.1
residual_rate_l2	$\frac{1}{n_q} \sum_j (\delta q_{t,j} - \delta q_{t-1,j})^2$	-0.01

D Downstream Policy Training Details

Common setup. All downstream tasks share the same simulation and training setup. Simulation runs at 200 Hz (MuJoCo, $\Delta t = 5$ ms) with a control decimation of 4, so policies act at 50 Hz; episodes last 20 s. Actions are joint-position offsets around the default standing configuration, applied to per-joint PD controllers. During training, the frozen UAN deterministically adds its residual correction to the commanded target before the simulator applies its actuator and contact models; at deployment, only the downstream task policy is exported. The observation-noise model is identical to that used during state-estimation data collection (Table 2), except that the jumping task uses a smaller joint-velocity noise of ± 0.5 rad/s (Table 9).

All tasks randomize foot-ground friction (uniform in $[0.3, 1.2]$, shared across all foot geoms) and push the robot by perturbing its root velocity at random intervals of 1–3 s, with task-specific push ranges. Policies trained *without* the UAN additionally randomize per-joint encoder bias (uniform in ± 0.015 rad, applied to joint position observations) and the torso center of mass (additive offset, uniform in ± 2.5 cm in x and y , ± 3 cm in z); in the UAN-augmented simulator all such robot-specific randomization is removed (Sec. 2.3). PPO hyperparameters for all tasks are listed in Table 5.

The running and jumping policies are additionally trained with a torque-speed actuator envelope [54]. For joint j , let $\bar{\tau}_j$ be the simulated stall torque (the torque available at zero speed) and $\bar{\dot{q}}_j$ the simulated no-load speed. The achievable PD torque is bounded symmetrically by

$$|\tau_j| \leq \bar{\tau}_j \left[1 - \left| \frac{\dot{q}_j}{\bar{\dot{q}}_j} \right| \right]_+, \quad (7)$$

where $[x]_+ = \max(x, 0)$. Rather than clipping a computed torque, each control step the commanded joint-position target is clipped to the range whose PD torque satisfies this bound, matching the clipping applied at deployment.

Table 7: Running policy reward weights.

Term	Math	Weight
<i>Task rewards</i>		
track_linear_velocity	$\exp(-(\ v_{xy}^{\text{cmd}} - v_{xy}^b\ ^2 + (v_z^b)^2)/0.25)$	2.0
track_angular_velocity	$\exp(-(\omega_z^{\text{cmd}} - \omega_z^b)^2/0.25)$	2.0
<i>Regularization</i>		
upright	$\exp(-\ g_{\text{torso},xy}^b\ ^2/0.2)$	1.0
pose	$\exp\left(-\frac{1}{n_q} \sum_j \frac{(q_j - q_j^0)^2}{\sigma_j (\ v^{\text{cmd}}\)^2}\right)$	1.0
<i>Penalties</i>		
dof_pos_limits	$\sum_j ([q_j^{\min} - q_j]_+ + [q_j - q_j^{\max}]_+)$	-10.0
foot_clearance	$I_{\text{cmd}} \sum_i h_i - 0.1 \ v_{i,xy}^w\ $	-2.0
self_collisions	$\sum_\ell \mathbf{1}[\max_k \ F_{k,\ell}^{\text{self}}\ > 10]$	-1.0
foot_swing_height	$I_{\text{cmd}} \sum_i I_i^{fc} \left(h_i^{\text{peak}}/0.1 - 1\right)^2$	-0.25
foot_slip	$I_{\text{cmd}} \sum_i I_i^c \ v_{i,xy}^w\ ^2$	-0.1
action_rate_l2	$\sum_j (a_{t,j} - a_{t-1,j})^2$	-0.1
angular_momentum	$\ h\ ^2$	-0.02
soft_landing	$I_{\text{cmd}} \sum_i I_i^{fc} \ F_i\ $	-10^{-5}

For the task-policy reward tables, I_{cmd} denotes the command-active mask, I_i^c denotes contact for foot i , I_i^{fc} denotes first contact, q^0 denotes the default pose, and $\sigma_j(\cdot)$ denotes the command-dependent posture tolerance.

D.1 Running Policy

To train a running policy, we adapt the default mujoco G1 velocity tracking task [29] with several modifications. We remove the base linear velocity observation to encourage the policy to rely on proprioception rather than privileged state, and extend the observation history to 5 steps. We add rough-terrain randomization and the torque-speed clipping of Eq. (7) to improve robustness. To improve motion quality, we lower the standing-command threshold for slow walking, add a low-height termination at 0.5 m to prevent kneeling local optima, and increase the penalty weight on joint position limits to -10. Table 7 gives the reward terms.

On the command side, we remove heading-tracking environments and instead dedicate 20% of environments to pure turning. Commands follow a curriculum over training iterations: $v_x^{\text{cmd}} \in (-1.0, 1.0)$ m/s and $\omega_z^{\text{cmd}} \in (-0.5, 0.5)$ rad/s from iteration 0, widening to $(-1.5, 2.0)$ m/s and $(-0.7, 0.7)$ rad/s at 5,000 iterations and to $(-2.0, 3.0)$ m/s at 10,000 iterations.

D.2 Force Control Policy

The commanded force is realized by a virtual force field: an impedance spring-damper attached to the right wrist and anchored at the end-effector position of the default standing pose. Every 10 s, a target force is sampled per axis from ± 30 N (zero force with probability 0.2) together with a band stiffness $k \in (25, 400)$ N/m and damping $0.1 k$; horizontal force commands are clipped so that the implied per-axis displacement $|F|/k$ stays within a 0.3 m reach. The virtual force is recomputed from the current end-effector position and velocity and applied at every physics substep (200 Hz), ramped up over 2–4 s, held for 2 s, and ramped back down, with 2 s of zero force at the start of each episode; the applied force is clamped to 50 N. The policy observes the commanded force and uses a 20-step observation history. The reward combines standing stabilization and force tracking (Table 8), inspired by [30].

Impedance Control. As an additional experiment, we use the force-control policy for compliant end-effector position control. We train a proprioceptive end-effector state estimator with the setup

Table 8: Force-control policy reward weights.

Term	Math	Weight
<i>Task rewards</i>		
track_force_ee	$\exp(-\ F^{\text{cmd}} - F^{\text{applied}}\ ^2/50)$	3.0
track_lin_vel_xy_exp	$\exp(-\ v^b\ ^2/0.25)$	2.0
track_ang_vel_z_exp	$\exp(-\ \omega^b\ ^2/0.5)$	2.0
<i>Regularization</i>		
upright	$\exp(-\ g_{\text{torso},xy}^b\ ^2/0.2)$	1.0
pose	$\exp\left(-\frac{1}{n_q} \sum_j \frac{(q_j - q_j^0)^2}{\sigma_j^2}\right)$	1.0
<i>Penalties</i>		
dof_pos_limits	$\sum_j ([q_j^{\min} - q_j]_+ + [q_j - q_j^{\max}]_+)$	-1.0
self_collisions	$\sum_\ell \mathbf{1}[\max_k \ F_{k,\ell}^{\text{self}}\ > 10]$	-1.0
action_rate_l2	$\sum_j (a_{t,j} - a_{t-1,j})^2$	-0.1
body_ang_vel	$\ \omega_{\text{torso},xy}^w\ ^2$	-0.05
angular_momentum	$\ h\ ^2$	-0.02

of Sec. ??: it maps a history of onboard observations (base angular velocity, projected gravity, joint positions and velocities, and previous joint targets) to the right end-effector position and velocity in the yaw-aligned base frame. At deployment, an outer impedance loop converts the end-effector position error into a force command, $F^{\text{cmd}} = K_p(\hat{\mathbf{p}} - \mathbf{p}^{\text{des}}) + K_d \hat{\mathbf{v}}$ with $K_p = 50 \text{ N/m}$ and $K_d = 0.2 K_p$, which is fed to the frozen force-control policy as its commanded force. The sign follows the training convention: the policy realizes a commanded band force by displacing the end-effector against it, so commanding a force pointing away from $\mathbf{p}^{\text{des}}$ drives the end-effector toward it. This realizes compliant end-effector position control (e.g., tracking circular trajectories of radius up to 0.3 m) without any arm-specific controller.

D.3 Jumping Policy

Overview. The jumping task requires the G1 to perform repeated standing vertical jumps: starting from rest, the robot must leave the ground with both feet, gain height, and land softly back into a stable stance. The task is built on a BeyondMimic-style motion tracking formulation [2]: a single retargeted reference clip provides the jumping “style,” while a training curriculum gradually hands the objective over from imitating the clip to explicitly maximizing flight height and minimizing landing impact. After the hand-off, the reference acts only as a low-weight style prior, and the final behavior (jump apex, landing strategy) is shaped by the task-specific reward terms rather than the clip. The reference motion is the jump from CMU MoCap clip 02_04 (“jump and balance”) [55], retargeted to the G1 and resampled at 50 Hz (200 frames, 4 s). The clip is padded with 1 s of standing at the beginning and end (300 frames, 6 s total), so the policy must initiate the jump from rest and hold a stable stance after landing.

Episode structure and reference-state initialization. At every reset, and whenever the reference clip reaches its final frame mid-episode, the robot is re-initialized to a reference frame of the clip: the start frame is sampled adaptively in proportion to an exponential moving average of per-frame failure rates, and the corresponding reference state is perturbed with uniform noise on the root pose ($\pm 5 \text{ cm}$ in xy , $\pm 1 \text{ cm}$ in z , $\pm 0.1 \text{ rad}$ roll/pitch, $\pm 0.2 \text{ rad}$ yaw), root velocity ($\pm 0.5 \text{ m/s}$ linear xy , $\pm 0.2 \text{ m/s}$ vertical, $\pm 0.52 \text{ rad/s}$ roll/pitch, $\pm 0.78 \text{ rad/s}$ yaw), and joint positions ($\pm 0.1 \text{ rad}$). Since the padded clip lasts 6 s, each 20 s episode contains roughly three full jump cycles.

Observations and actions. The actor receives only quantities available on hardware: no global position and no root linear velocity. Table 9 lists the observation terms; concatenated, they form a 157-dimensional policy input. The reference joint targets in the *command* term provide the policy with an implicit phase signal for the clip. The critic additionally observes privileged, noise-free

Table 9: Actor observation space for the jumping task. Noise is additive, sampled uniformly from $[-n, n]$ per component. The anchor body is the torso link.

Term	Dim.	Noise n
Reference joint positions and velocities (current frame)	58	–
Reference anchor orientation in base frame (6D rotation)	6	0.05
Projected gravity (IMU)	3	0.05
Base angular velocity (IMU)	3	0.2
Joint positions (relative to default)	29	0.01
Joint velocities	29	0.5
Previous action	29	–
Total	157	

state: the root linear velocity, the reference anchor position expressed in the base frame, and the positions and orientations of all 14 tracked bodies relative to the reference. Actions follow the common setup with a per-joint action scale of $0.25 \tau_j^{\max}/k_{p,j}$, where $\tau_j^{\max}$ and $k_{p,j}$ are the effort limit and proportional gain of joint j , and the torque–speed envelope of Eq. (7).

Rewards and curriculum. Table 10 lists all reward terms. Let $\hat{(\cdot)}$ denote reference quantities, $\mathbf{p}_a$ and $\mathbf{q}_a$ the position and orientation of the anchor body (torso), $\mathcal{B}$ the set of 14 tracked bodies, and $\mathbf{q}_1 \ominus \mathbf{q}_2$ the geodesic angle between two orientations. Before computing body position and orientation errors, the reference bodies are rigidly re-anchored to the robot’s current anchor xy -position and yaw heading, making these terms invariant to global horizontal drift. The flight indicator $\mathbf{1}_{\text{air}}$ is 1 only when *both* feet are out of contact, so the jump-height reward cannot be exploited by standing tall or lifting a single foot; z_{root} is the pelvis height with $z_0 = 0.79$ m the nominal standing height. The landing-impact penalty sums the contact-force magnitudes $\|\mathbf{f}_i\|$ of each foot i on the step it first regains contact, normalized by the robot’s weight $W = 327$ N. $\mathcal{S}$ denotes the 10 left/right symmetric joint pairs (legs, shoulders, elbows).

Reward weights follow a piecewise-constant curriculum: tracking weights step down and jump-objective weights step up at 8,000 and again at 12,000 iterations (0.79×10^9 and 1.18×10^9 environment steps). Before 8k iterations training is pure motion tracking; after the ramp, the jump and landing terms dominate while the tracking terms persist only as a style prior. The final policy is taken from a checkpoint after the jump behavior has converged.

Terminations. Episodes terminate on timeout (20 s) or when the robot deviates too far from the reference. The same curriculum that ramps the reward weights also loosens the termination thresholds (Table 11), since a policy optimizing jump height must be allowed to deviate substantially from the clip during flight. The anchor tilt term compares the z -component of the gravity direction expressed in the anchor frame between robot and reference, i.e. it triggers on excessive relative tilt.

Domain randomization. Domain randomization follows the common setup. Pushes perturb the root velocity uniformly in ± 0.3 m/s horizontally, ± 0.1 m/s vertically, and ± 0.3 rad/s angularly, and are gated to the padded standing frames of the reference clip: they train push recovery in stance without ever disturbing the robot mid-jump, which would make jump timing unlearnable.

E Sim-to-Sim Experiment Details

Target simulator construction. For sim-to-sim evaluation, we construct easy, medium, and hard target simulators by applying deterministic actuator perturbations to the nominal G1 model. The perturbations modify joint damping, frictionloss, action delay, encoder bias, and armature. The easy target is chosen to produce a measurable but recoverable tracking gap for the nominal baseline; medium and hard scale the same perturbation vector to test increasingly severe actuator mismatch. The values are listed in Table 12.

Table 10: Reward terms for the jumping task. The three weight columns give the piecewise-constant curriculum values applied from iteration 0, 8,000, and 12,000 onward, respectively. Terms with a single value are constant throughout training.

Term	Math	w_0	w_{8k}	w_{12k}
<i>Motion tracking (style prior after ramp)</i>				
motion_global_root_pos	$\exp(-\ \hat{\mathbf{p}}_a - \mathbf{p}_a\ ^2/0.3^2)$	0.5	0.25	0.1
motion_global_root_ori	$\exp(-(\hat{\mathbf{q}}_a \ominus \mathbf{q}_a)^2/0.4^2)$	0.5	0.25	0.1
motion_body_pos	$\exp(-\frac{1}{ B } \sum_b \ \hat{\mathbf{p}}_b - \mathbf{p}_b\ ^2/0.3^2)$	1.0	0.5	0.2
motion_body_ori	$\exp(-\frac{1}{ B } \sum_b (\hat{\mathbf{q}}_b \ominus \mathbf{q}_b)^2/0.4^2)$	1.0	0.5	0.2
motion_body_lin_vel	$\exp(-\frac{1}{ B } \sum_b \ \hat{\mathbf{v}}_b - \mathbf{v}_b\ ^2/1.0^2)$	1.0	0.5	0.1
motion_body_ang_vel	$\exp(-\frac{1}{ B } \sum_b \ \hat{\boldsymbol{\omega}}_b - \boldsymbol{\omega}_b\ ^2/\pi^2)$	1.0	0.5	0.1
<i>Jump objective (curriculum ramps up)</i>				
jump_height_root	$\mathbf{1}_{\text{air}} \cdot \max(0, z_{\text{root}} - z_0)$	0	3.0	6.0
landing_impact	$\frac{1}{W} \sum_{i \in \text{feet}} \ \mathbf{f}_i\ \cdot \mathbf{1}[\text{first contact}_i]$	0	-2.0	-4.0
<i>Regularization (constant)</i>				
action_rate_l2	$\ a_t - a_{t-1}\ ^2$		-0.1	
joint_limit	$\sum_j \max(0, q_j - q_j^{\text{soft}})$		-10	
self_collisions	# substeps with self-contact force > 10 N		-10	
joint_torques_normalized	$\sum_j (\tau_j / \tau_j^{\text{max}})^2$		-10^{-3}	
joint_symmetry	$\sum_{(l,r) \in \mathcal{S}} (q_l - q_r)^2$		-0.2	
waist_deviation	$q_{\text{waist-yaw}}^2 + q_{\text{waist-roll}}^2$		-0.5	

Table 11: Termination conditions and their curriculum schedule.

Condition	Trigger	θ_0	θ_{8k}	θ_{12k}
Anchor height	$ \hat{z}_a - z_a > \theta$	0.5 m	0.75 m	1.0 m
Anchor tilt	$ \hat{g}_z - g_z > \theta$	0.8	1.0	1.2
End-effector height	$\max_{b \in \text{feet, wrists}} \hat{z}_b - z_b > \theta$	0.25 m	0.5 m	1.0 m

Sim-to-sim policy training. For each target simulator, we train downstream velocity-tracking policies either in the nominal simulator or in the UAN-augmented simulator. The WTW UAN is trained from 5 minutes of WTW target data. The WTW+Vel UAN is trained from the same 5 minutes of WTW data plus 5 minutes of running data, allowing us to test whether task-specific running data improves over the walking-only residual model.

F World Model Details

Overview. Every method is evaluated on a single hard target environment: omnidirectional base velocity tracking on flat ground from [33].

Observation, action, reward. The policy receives the same 99-dimensional observation as the on-policy baseline: base linear and angular velocity read from the IMU site `imu.in.pelvis` (6), projected gravity (3), a 3-D commanded base velocity, joint position residuals $q - q_{\text{default}}$ (29), joint velocities (29), and the previous action (29).

Data collection. We collect 6 million transitions (around 33 hours of data) on the hard target environment using a baseline policy. For data collection we record the IMU sensor readings rather than the body-state `lin.vel.w/ang.vel.w` pair, so the world model is trained on exactly the inputs the policy sees at evaluation time.

Robotic world model. The world model is a GRU ensemble (5 members, hidden size 256, 2 stacked layers) with a 32-step history horizon, an 8-step forecast horizon, and a contact channel of dimension 30 (Table 13). At policy-training time, a per-step epistemic-uncertainty signal formed from

Table 12: **Sim-to-sim target perturbations.** Each target simulator applies deterministic actuator perturbations to the nominal G1 model. The medium and hard targets scale the same perturbation vector used for easy.

Perturbation	easy	medium	hard
Damping offset	+2.0	+4.0	+6.0
Frictionloss offset	+0.5	+1.0	+1.5
Action delay	1 step	2 steps	3 steps
Encoder bias	+0.003 rad	+0.006 rad	+0.009 rad
Armature multiplier	1.5×	2.0×	2.5×

ensemble disagreement on next-state predictions is subtracted from the reward with MOPO penalty $\lambda = 1.0$ [33]. Imagined rollouts terminate when the learned termination head exceeds 0.9; lower thresholds collapsed training because the policy learned to exploit short, high-reward fragments inside the model.

Policy training. Given a trained world model, we run PPO on imagined rollouts using the same `rs1_rl` algorithm and hyperparameters as the baseline (Table 14). The outer loop runs for 5,000 iterations, following [33]; training longer overfits the policy to the inaccurate WM and degrades target performance. UAN is trained using only 5 minutes of data.

Paradigms compared. The *offline* world model is trained once on the 6 M expert dataset. Variant WM combines it with a randomly initialized actor and is the purest from-scratch instantiation of the recipe. Variant WM FT reuses the same world model but starts the actor from a baseline checkpoint and adds a behavior anchor $\beta \text{KL}(\pi \parallel \pi_{\text{baseline}})$ with $\beta = 1.0$ to the PPO loss.

Evaluation protocol. Every policy is evaluated on the hard target with 32 parallel environments, 2,000 simulator steps (up to two 1,000-step episodes per environment), and a 20 s episode cap. We report mean and standard deviation of *survival length (%)*, defined as episode length divided by the 1,000-step cap, across all completed episodes. The evaluation environment and seed distribution are identical across Expert, Baseline, UAN, and all RWM-U variants.

Table 13: World-model architecture and training hyperparameters used for RWM-U on G1.

Parameter	Value
<i>Architecture</i>	
Backbone	GRU ensemble
Ensemble size	5
RNN hidden size	256
RNN layers	2
Head MLP (state / ext. / contact / term.)	[128]
<i>Dimensions</i>	
History horizon H	32
Forecast horizon F	8
Observation dim	99
Action dim	29
Contact channel dim	30
<i>Data</i>	
Dataset size	6M transitions $\approx$ 33 hours
File shard size	10,000 rows
<i>Optimization</i>	
Batch size	1024
Epochs	4
Optimizer	Adam
Learning rate	$1 \times 10^{-4} \rightarrow 1 \times 10^{-6}$ (cosine)
Weight decay	1×10^{-5}
Gradient steps (approx.)	1.8×10^4

Table 14: PPO hyperparameters used for MOPO-PPO policy training inside the RWM-U world model. The baseline PPO column is shown for reference; only the bold rows differ.

Parameter	Baseline PPO	MOPO-PPO (RWM-U)
<i>Rollout collection</i>		
Parallel envs	4096	4096
Steps per environment	24	24
Rollout source	simulator	learned WM
Outer iterations	20,000	5,000
<i>Optimization</i>		
Learning epochs	5	5
Mini-batches	4	4
Learning rate	10^{-3}	10^{-3}
Schedule	adaptive	adaptive
Max gradient norm	1.0	1.0
<i>PPO objective</i>		
Clip parameter	0.2	0.2
Entropy coefficient	0.01	0.008
Value-loss coefficient	1.0	1.0
Use clipped value loss	yes	yes
Discount γ	0.99	0.99
GAE λ	0.95	0.95
Target KL	0.01	0.01
Actor / critic MLP	[256, 128, 128]	[256, 128, 128]
<i>RWM-U specific</i>		
Uncertainty penalty λ	–	1.0
Termination threshold	–	0.9