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

Nolan Fey[1,*], Khai Nguyen[*], Yash Prabhu[1], Gabriel B. Margolis[1], Pulkit Agrawal[1]
MIT, [1]Improbable AI Lab, [*]Co-first authors.
{nolanfey,khain,yprabhu,gmargo,pulkitag}@mit.edu

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

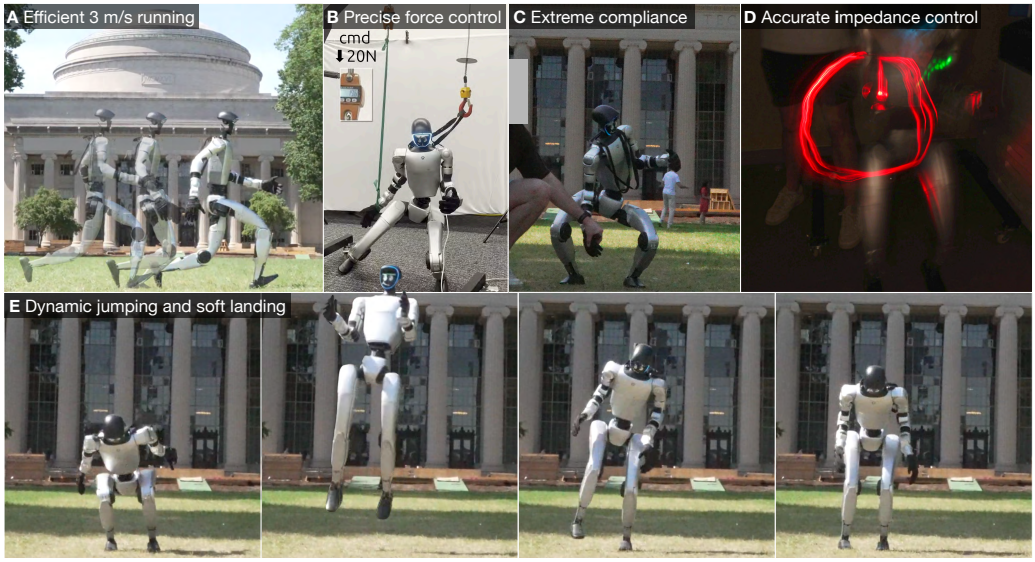

Fig. 1: **Sim-to-real transfer of efficient, precise, and dynamic humanoid behaviors.** *Contact-UAN* enables residual world model learning from hardware trajectories with contact by inferring contact-consistent simulator reset states from onboard proprioception. The resulting residual world model constrains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

***Abstract*—Humanoid policies trained in simulation can transfer qualitatively to hardware while failing to transfer force interactions, compromising torque efficiency, load regulation, and robustness to impacts. This failure can arise from actuator sim-to-real gaps because forces are downstream of 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.**

***Index Terms*—Sim-to-Real Transfer, Humanoids, Residual World Models**

I. Introduction

For sim-to-real humanoid control, qualitative motion transfer is only a coarse measure of success: the robot must also generate forces with the magnitude and timing needed to transfer efficiency, load regulation, and impact absorption [1–4]. Policy optimization can exploit imperfect actuator dynamics, producing motions that transfer while the underlying forces do not, leading to excessive torques, poor commanded-force tracking, or failures during high-impulse phases. We therefore study how improving the actuator model used during policy training reduces this force-level sim-to-real discrepancy.

Analytic simulators [5, 6] approximate the actuator effects, such as torque saturation and load-dependent nonlinear friction, on which efficient running, force tracking, and jumping depend [7, 8]. Domain randomization can yield conservative policies that under-utilize hardware [9–12]; adaptation and SysID require choosing which parameters to randomize and identify [2, 13–16]; fully learned world models (WMs) capture omitted effects but must model the full robot-environment dynamics [17–20]. We therefore follow residual modeling and actuator learning [7, 21–26]: retain the simulator's physics priors and learn only the sim-to-real gap in the actuator command channel.

Learning such a residual requires a training signal. Supervised actuator networks use measured motor torques [7],

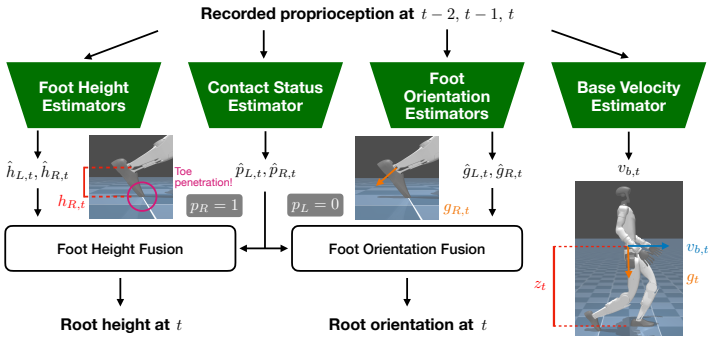

Fig. 2: **Our proprioceptive state estimation framework.** Simulation supervision maps load-dependent proprioceptive cues to the simulator-side stance-foot pose consistent with them under the soft-contact model. Equations (4)–(6) fuse confident contacts into the root pose used for hardware replay in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

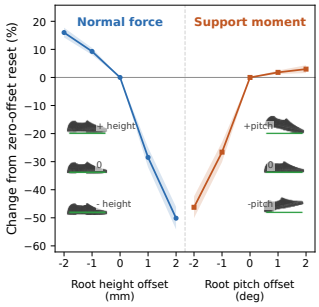

Fig. 3: **Contact-wrench sensitivity.** mm-scale foot height errors substantially change force, and degree-scale pitch errors substantially change moment.

which many humanoid platforms lack. Without labels, residual methods can *replay* recorded trajectories: reset the simulator to a recorded hardware state, apply the recorded commands, and train residual commands so the simulated rollout matches the recorded one [8, 27]. This objective isolates modeling error only if replay begins from a hardware-equivalent simulator state. Suspended, free-space data makes this a kinematics problem [8, 15], but lacks the contact information that shapes load-dependent actuator behavior. ASAP [27] collects contact data with motion capture, yet even ground-truth poses do not suffice: reproducing hardware ground-reaction forces in a *soft-contact* simulator requires slight ground penetration and tilt, as millimeter- and degree-scale pose errors substantially change simulated wrenches (Fig. 3). Replay from wrong foot poses thus teaches the residual to correct reset artifacts instead of actuator discrepancies.

We introduce *Contact-UAN*, a residual world-modeling approach for force- and impulse-sensitive humanoid behaviors that extends Unsupervised Actuator Networks (UAN) [8] to floating-base motion with contacts. The core contribution is contact-consistent state initialization from proprioception alone (Fig. 2): Contact-UAN infers the simulator-side stance-foot pose, including the small penetration and tilt that reproduce the hardware contact wrenches, together with the base velocity completing the reset state, so the UAN residual learns true actuator modeling errors. We evaluate on a Unitree G1 with the state-of-the-art BeyondMimic recipe as the nominal simulator [2, 28], on tasks spanning distinct actuator-sensitive regimes: torque-efficient running, precise force control and impedance, and impulsive jumping and landing. A single UAN trained from **6.5 minutes** of real-world walking data improves transfer across all of them, delivering 20% lower running torque at preserved velocity tracking, 43% lower force-tracking error, extreme compliance and impedance control, and robust jumps that fail without the UAN (Fig. 1). In sim-to-sim, Contact-UAN closes larger controlled actuator gaps and is 400× more data-efficient than fully learned WMs.

II. Method

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

A. 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, so we 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, 29]. 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 (Fig. 2).

Rather than estimating contact force, which alone does not specify the reset state, we 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. Absolute base height is omitted from the observations: otherwise the estimator can copy height instead of using load-dependent cues, a shortcut that need not transfer since hardware height may differ from the simulator height reproducing the same contact force.

We first estimate contact status and train the pose regressors only on stance samples: swing poses largely map to zero wrench, so their variation dominates the objective without informing the contact model, whereas in stance small pose differences strongly affect the wrench. In our MuJoCo model (Table IV) [6], a $+2$ mm root-height offset reduces normal force by roughly 50% and a $-2°$ pitch offset changes the support moment by roughly 45% (Fig. 3). Stance conditioning thus focuses the regression on the contact-sensitive variation needed at reset (Figs. 10 and 11).

Each confident stance-foot estimate yields a candidate root height and orientation through forward kinematics; candidates are fused with contact-probability weights, and windows with no confident contact are discarded (equations in Appendix A-C). The estimators are trained in simulation, so their

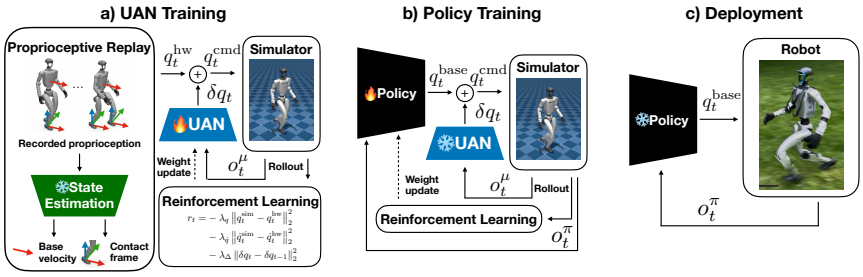

Fig. 4: **Contact-UAN.** a) Train a UAN to add joint-position target corrections to reduce simulation transition error relative to recorded hardware data, with base velocities and contact poses labeled via proprioceptive state estimators. b) Train a task policy in the UAN-augmented simulator with no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

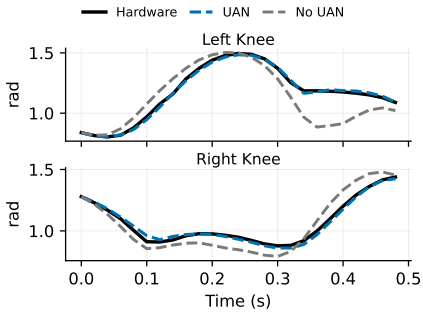

Fig. 5: **Hardware replay knee trajectories** on held-out running data: our estimation and the UAN closely match hardware, while ablations deviate.

key transfer assumption is that load-dependent proprioceptive cues remain informative across domains: Fig. 10A probes this in controlled sim-to-sim, and Fig. 10B–C provide end-to-end evidence through improved hardware replay. The estimators are used only for offline relabeling. Together with estimated base velocity, the fused root pose initializes contact-consistent replay, so the UAN learns actuator discrepancies rather than simulator reset artifacts.

B. UAN as a Residual World Model

Let s_t denote the robot state and a_t the action. The real transition is $f(s_t, a_t)$ and 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 its 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 δq_t added to a base target q_t^{base}:

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

During UAN training, $q_t^{\text{base}} \equiv q_t^{\text{hw}}$ is the recorded replay target; during downstream training, q_t^{base} is the policy output. The UAN thus preserves the simulator's rigid-body dynamics, actuator implementation, contact solver, and integration, modulating only the command they integrate forward. To avoid overfitting, the observation gives the UAN only local actuator-tracking information [7, 8]: for each joint it stacks the target-position error and joint velocity over a short history,

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

with $H_\mu = 3$ (60 ms). Parameters are shared across joints with identical actuators: each of the 5 actuator types has a two-layer $(128, 128)$ ELU MLP. The UAN is trained by PPO [30] on replayed hardware windows (4096 envs, 24-step / 480 ms horizon) with a reward penalizing joint-position and joint-velocity MSE and residual rate. Data comes from a Walk-These-Ways-style policy [31] eliciting diverse stance durations, swing speeds, and impacts (Section C).

C. Policy Training and Deployment

After UAN training, we freeze its weights and insert it into downstream simulation as a deterministic residual correction, removing all robot-specific DR while still randomizing terrain mesh, ground friction, and external pushes. The task policy outputs joint-position targets, but the simulator receives the UAN-corrected target of Eq. (2), so high-reward behaviors must remain feasible under the corrected actuator response. Only the task policy runs on the robot.

III. Experiments and Results

We evaluate on a 29-DoF Unitree G1 (5 actuator types), whose quasi-direct-drive actuators give a small sim-to-real gap: the nominal BeyondMimic simulator [2, 28] is already strong, testing whether a residual WM still helps. We deploy a WTW-style policy for 6.5 minutes, relabel each window (Sec. II-A), and train the UAN. We then freeze it and train running, force-control, and jumping policies with the UAN in the loop and *no* robot-specific DR; running and force-control use no human motion references, with strong baselines trained by the same pipeline without the UAN. As a sim-to-real proxy, we also build `easy`/`medium`/`hard` targets by scaling a deterministic actuator perturbation, and compare against fully learned WM baselines inspired by RWM-U [32] trained on $400\times$ more data. Details are in Sections D to F.

Residual WM accuracy. In the `hard` sim-to-sim setting (where ground-truth contact pose is available), directly regressing foot height over all gait phases yields large errors, while our contact-conditioned estimator reduces contact-height RMSE by an order of magnitude (Fig. 10A). Relabeling alone improves over direct initialization; adding the UAN further reduces 24-step joint-position MSE (Fig. 10B) and produces knee trajectories that closely match hardware on held-out running data (Fig. 5, Fig. 11), indicating it closes the actuator component of the gap rather than compensating for reset error. On the same replay, Contact-UAN reduces aggregate all-joint torque-usage mismatch to hardware by 69.2% (Fig. 9), giving downstream policy learning a more accurate torque signal.

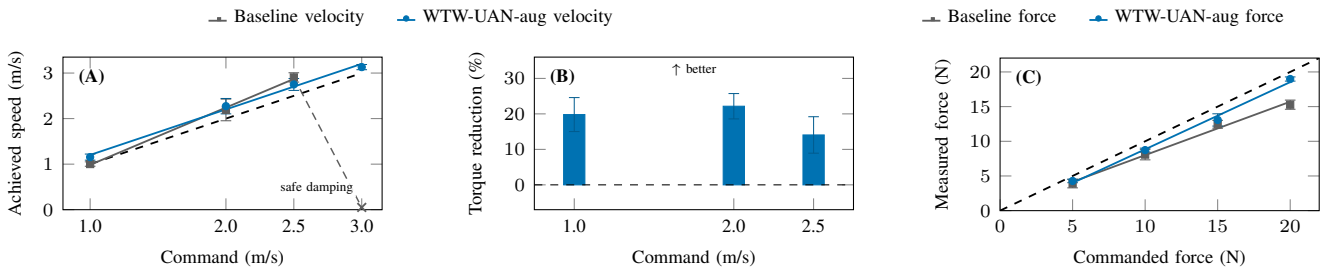

Fig. 6: **Improved running and force control sim-to-real transfer.** UAN running policy retains tracking ability (A) while improving efficiency (B). The baseline marker at $3.0\ \mathrm{m\,s^{-1}}$ denotes protection-triggered safe damping mode. UAN force policy reduced average force tracking error by $42.8 \pm 12.1\%$ (C). Both baselines were trained with BeyondMimic-style SysId [2].

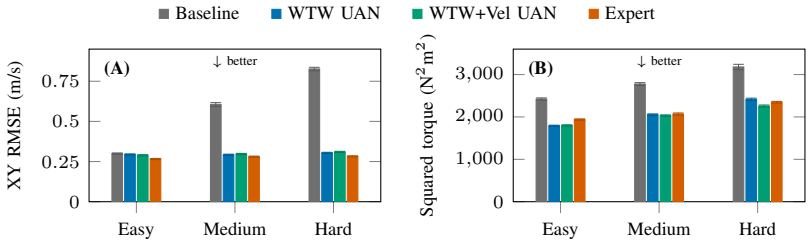

Fig. 7: **Sim-to-sim velocity policy transfer.** Policies trained with UAN reach expert-level performance across different gap settings. Adding task-specific data makes a negligible difference, showing WTW data is sufficient.

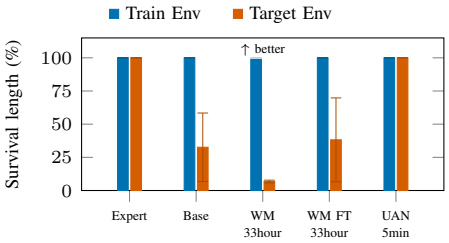

Fig. 8: **Data efficiency of UAN.** UAN avoids falling with only 5 minutes of target data, while fully-learned WMs fail with 400× more data.

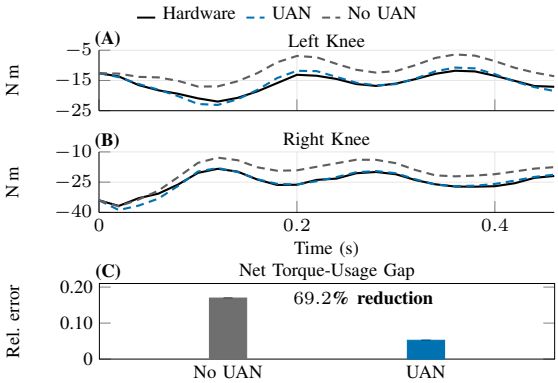

Fig. 9: **Torque replay gap.** Contact-UAN better matches held-out left (A) and right (B) knee torques and reduces aggregate all-joint net torque-usage mismatch by 69.2% (C).

Torque-efficient running. At commanded speeds of 1.0, 2.0, and 2.5 $\mathrm{m\,s^{-1}}$, the UAN policy preserves velocity tracking while reducing torque usage relative to the baseline (Fig. 6A,B). The UAN policy also transfers to a 3.0 $\mathrm{m\,s^{-1}}$ command, achieving $3.13 \pm 0.06\ \mathrm{m\,s^{-1}}$ (Fig. 1A), whereas the baseline triggers motor-overheating protection within seconds.

Whole-body force control, compliance, and impedance. Tracking end-effector force against a dynamometer through an exercise band (requiring whole-body coordination), the UAN force policy reduces force-tracking error by $42.8 \pm 12.1\%$ relative to the baseline (Fig. 6C). 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). Composing it with a proprioceptive end-effector estimator and PD outer loop yields *impedance position control* that tracks a circular trajectory using onboard sensing alone (Fig. 1D, Section D-B).

Dynamic, robust jumping and soft landing. Our jumping policy first tracks a reference jump [2], then is fine-tuned to increase height and reduce landing impact. A baseline trained without the UAN fails to land on hardware, while the UAN policy transfers successfully (Fig. 1E): it jumps higher by squatting deeper and generating larger ground-reaction forces, lands softly, executes consecutive jumps, and recovers from disturbances (Section D-C).

Sim-to-sim ablation and data efficiency. Across `easy`/`medium`/`hard` targets, the baseline policy degrades sharply as the gap grows while UAN policies achieve expert-level tracking error and torque usage (Fig. 7); adding task-specific running data gives little benefit, indicating WTW data suffices for a residual that generalizes to running. Just 5 minutes of `hard`-target transitions lets the UAN match the expert's survival length, whereas fully learned offline WMs perform poorly even with 400× more data (Fig. 8): physics priors give generalization, while a small learned residual supplies the missing actuator fidelity.

IV. Conclusion

We present *Contact-UAN*, a residual world model for humanoid control that learns an actuator-level correction from offline proprioceptive data while preserving an analytical simulator, using onboard estimation to initialize hardware replay with consistent contact forces. From 6.5 minutes of walking data it generalizes to efficient running, precise force control, compliance, impedance control, and robust jumps, a data-efficient alternative to fully learned dynamics models.

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.

[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."

[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*, pp. 5026–5033, 2012.

[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*, vol. 4, Jan. 2019.

[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.

[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*, vol. 9, p. 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)*, pp. 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*, pp. 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*, pp. 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, X. Fu, Y. Ji, and P. Agrawal, "Learning to see physical properties with active sensing motor policies," 2023.

[17] 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.

[18] 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.

[19] 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*, vol. 2025, pp. 62175–62195, 2025.

[20] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, "Daydreamer: World models for physical robot learning," in *Conference on robot learning*, pp. 2226–2240, PMLR, 2023.

[21] 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)*, pp. 7219–7226, IEEE, 2021.

[22] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, 2020.

[23] 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*, pp. 817–828, PMLR, 2018.

[24] 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*, vol. 9, no. 10, pp. 8523–8530, 2024.

[25] 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)*, pp. 2884–2890, IEEE, 2021.

[26] Y. Yu and L. Liu, "Neural fidelity calibration for informative sim-to-real adaptation," *arXiv preprint arXiv:2504.08604*, 2025.

[27] 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.

[28] K. Zakka, Q. Liao, B. Yi, L. L. Lay, K. Sreenath, and P. Abbeel, "mjlab: A lightweight framework for gpu-accelerated robot learning," *arXiv preprint arXiv:2601.22074*, 2026.

[29] T. Portela, G. B. Margolis, Y. Ji, and P. Agrawal, "Learn-

ing force control for legged manipulation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15366–15372, IEEE, 2024.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[31] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Conference on Robot Learning*, pp. 22–31, PMLR, 2023.

[32] 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.

[33] 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.

[34] Y.-H. Shin, T.-G. Song, G. Ji, and H.-W. Park, "Actuator-constrained reinforcement learning for high-speed quadrupedal locomotion," 2023.

[35] Carnegie Mellon University, "CMU graphics lab motion capture database." http://mocap.cs.cmu.edu.

[36] 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.

[37] P. K. Khosla and T. Kanade, "Parameter identification of robot dynamics," in *1985 24th IEEE conference on decision and control*, pp. 1754–1760, IEEE, 1985.

[38] 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*, vol. 5, no. 3, pp. 101–119, 1986.

[39] T. Lee, P. M. Wensing, and F. C. Park, "Geometric robot dynamic identification: A convex programming approach," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 348–365, 2020.

[40] 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*, vol. 43, no. 14, pp. 2274–2302, 2024.

[41] 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)*, pp. 1–8, 2023.

[42] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier, "Differentiable simulation for physical system identification," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3413–3420, 2021.

[43] 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.

[44] A. Miller, F. Yu, M. Brauckmann, and F. Farshidian, "High-performance reinforcement learning on spot: Optimizing simulation parameters with distributional measures," 2025.

[45] 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.

[46] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, Oct. 2020.

[47] 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.

[48] 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.

[49] 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*, vol. 7, no. 2, pp. 4630–4637, 2022.

[50] 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)*, pp. 5078–5084, IEEE, 2023.

[51] 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.

[52] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *ACM Sigart Bulletin*, vol. 2, no. 4, pp. 160–163, 1991.

[53] 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)*, pp. 392–399, IEEE, 2023.

[54] 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.

Appendix A
State Estimation

A. 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 IV), 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.

B. Architectures

Table I 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.

C. 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

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

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 $h^{\text{fk}}_{i,t}(z=0)$ be the foot height from forward kinematics when the root height is set to zero. The root height implied by foot i is

$$\hat{z}^{(i)}_t = \hat{h}_{i,t} - h^{\text{fk}}_{i,t}(z=0), \qquad \hat{z}_t = \sum\nolimits_i \alpha_{i,t}\,\hat{z}^{(i)}_t. \tag{5}$$

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

$$\begin{aligned} \hat{g}^{(i)}_t &= \text{normalize}\Big(R_{R\leftarrow F_i}(q_t)\,\hat{g}^{F_i}_{i,t}\Big), \\ \hat{g}_t &= \text{normalize}\big(\sum\nolimits_i \alpha_{i,t}\,\hat{g}^{(i)}_t\big). \end{aligned} \tag{6}$$

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 (Fig. 2).

D. Observation Noise

The simulated observation-noise model used for estimator-data collection is listed in Table II. 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 IX).

Appendix B
UAN Training Details

Table III lists the five G1 actuator groups over which the UAN shares its per-group actor MLPs (Sec. II-B). Table IV lists relevant contact model parameters used in our mjlab simulation [28].

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 VI; PPO settings for the UAN, the data-collection policy, and all downstream task policies are consolidated in Table V. We use the RSL-RL library [33].

Appendix 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. II-B). 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 V.

Appendix 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

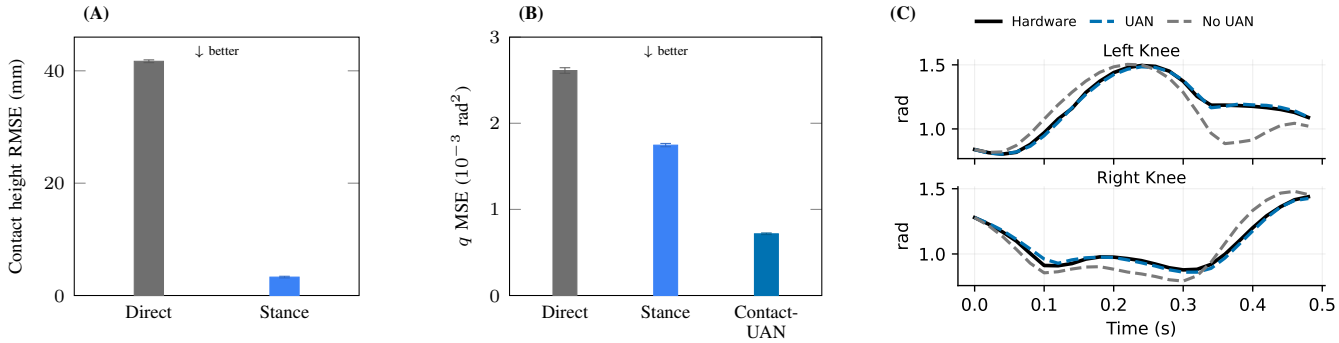

Fig. 10: **State estimation ablation and UAN fit.** (A) compares contact-height estimates in simulation of an estimator trained on *both stance and swing* data (direct) versus *only stance* data (stance, ours). Both estimators were trained on base simulation data and evaluated in the `hard` target environment (Appendix E). 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.

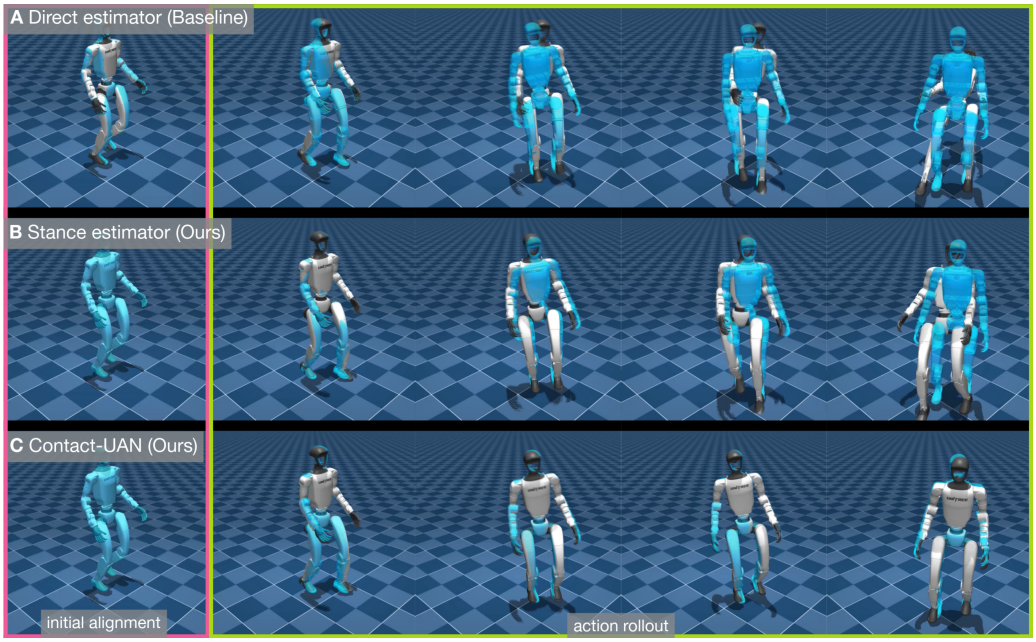

Fig. 11: **Open-loop rollouts.** We set the initial simulator state to the estimated state and rollout recorded hardware actions. Our estimated simulator reset state is in blue. (A) uses a foot height estimator trained on *both stance and swing* data; (B) uses our estimator trained on *only stance* data to align initial contacts; (C) uses our estimator and the UAN to close the actuation gap.

to that used during state-estimation data collection (Table II), except that the jumping task uses a smaller joint-velocity noise of ± 0.5 rad/s (Table IX).

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. II-C). PPO hyperparameters for all tasks are listed in Table V.

The running and jumping policies are additionally trained with a torque-speed actuator envelope [34]. 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

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

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.

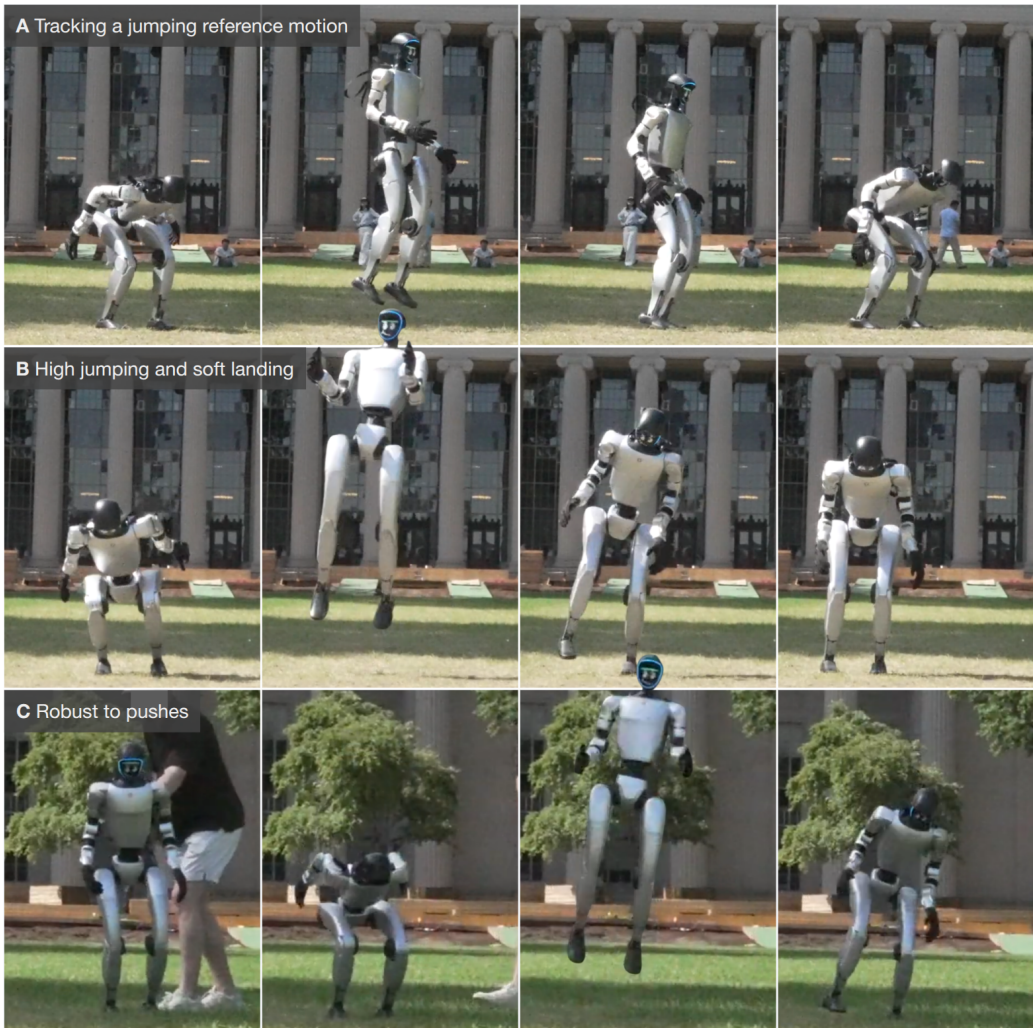

Fig. 12: **Dynamic and robust jumping policies.** With the sim-to-real gap bridged by UAN, we demonstrate more challenging behaviors. The policy is first trained to track a jumping reference motion (A), then fine-tuned to maximize jump height while reducing landing impact (B). It also remains robust to external disturbances (C).

TABLE I: 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

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.

A. Running Policy

To train a running policy, we adapt the default mjlab G1 velocity tracking task [28] 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 VII 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:

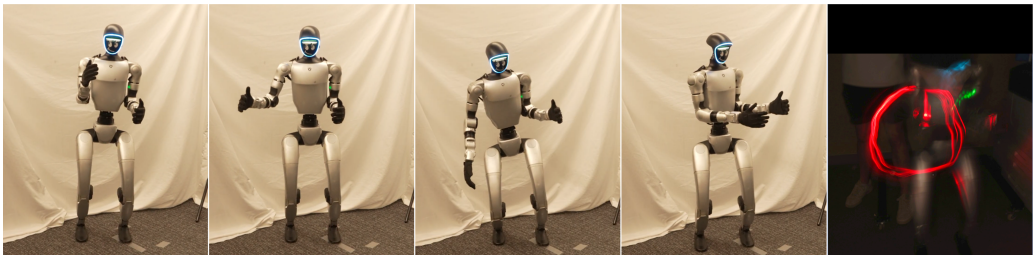

Fig. 13: **Impedance position control.** Combining a precise force control policy with a proprioceptive state estimator, we develop an impedance controller for closed-loop end-effector position tracking. The controller tracks a circular trajectory reasonably well without any external sensing.

TABLE II: 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

TABLE III: 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 IV: 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]

$v_x^{\mathrm{cmd}} \in (-1.0, 1.0)$ m/s and $\omega_z^{\mathrm{cmd}} \in (-0.5, 0.5)$ 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.

B. 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 VIII), inspired by [29].

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 of Sec. II-A: 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, $F^{\mathrm{cmd}} = K_p(\hat{\mathbf{p}} - \mathbf{p}^{\mathrm{des}}) + K_d\,\hat{\mathbf{v}}$ with $K_p = 50$ 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 $\mathbf{p}^{\mathrm{des}}$ 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-

TABLE V: 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
Optimization					
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
PPO objective					
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
Networks (ELU activations)					
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 VI: UAN replay reward weights.

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

specific controller.

C. 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") [35], 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 (± 5 cm in xy, ± 1 cm in z, ± 0.1 rad roll/pitch, ± 0.2 rad yaw), root velocity (± 0.5 m/s linear xy, ± 0.2 m/s vertical, ± 0.52 rad/s roll/pitch, ± 0.78 rad/s yaw), and joint positions (± 0.1 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 IX 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 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 X 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

TABLE VII: Running policy reward weights.

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

TABLE VIII: Force-control policy reward weights.

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

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 $z_0 = 0.79\,\text{m}$ the nominal standing height. The landing-impact penalty sums the contact-force magnitudes $\lVert \mathbf{f}_i\rVert$ of each foot i on the step it first regains contact, normalized by the robot's weight $W = 327\,\text{N}$. $\mathcal{S}$ 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 XI), 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

TABLE IX: 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	

TABLE X: 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}
Motion tracking (style prior after ramp)				
`motion_global_root_pos`	$\exp\big(-\lVert\hat{\mathbf{p}}_a - \mathbf{p}_a\rVert^2/0.3^2\big)$	0.5	0.25	0.1
`motion_global_root_ori`	$\exp\big(-(\hat{\mathbf{q}}_a \ominus \mathbf{q}_a)^2/0.4^2\big)$	0.5	0.25	0.1
`motion_body_pos`	$\exp\big(-\frac{1}{\lvert\mathcal{B}\rvert}\sum_b \lVert\hat{\mathbf{p}}_b - \mathbf{p}_b\rVert^2/0.3^2\big)$	1.0	0.5	0.2
`motion_body_ori`	$\exp\big(-\frac{1}{\lvert\mathcal{B}\rvert}\sum_b (\hat{\mathbf{q}}_b \ominus \mathbf{q}_b)^2/0.4^2\big)$	1.0	0.5	0.2
`motion_body_lin_vel`	$\exp\big(-\frac{1}{\lvert\mathcal{B}\rvert}\sum_b \lVert\hat{\mathbf{v}}_b - \mathbf{v}_b\rVert^2/1.0^2\big)$	1.0	0.5	0.1
`motion_body_ang_vel`	$\exp\big(-\frac{1}{\lvert\mathcal{B}\rvert}\sum_b \lVert\hat{\boldsymbol{\omega}}_b - \boldsymbol{\omega}_b\rVert^2/\pi^2\big)$	1.0	0.5	0.1
Jump objective (curriculum ramps up)				
`jump_height_root`	$\mathbf{1}_{\text{air}} \cdot \max(0,\, z_{\text{root}} - z_0)$	0	3.0	6.0
`landing_impact`	$\frac{1}{W}\sum_{i\in\text{feet}} \lVert\mathbf{f}_i\rVert \cdot \mathbf{1}[\text{first contact}_i]$	0	-2.0	-4.0
Regularization (constant)				
`action_rate_l2`	$\lVert a_t - a_{t-1}\rVert^2$		-0.1	
`joint_limit`	$\sum_j \max(0,\, \lvert q_j\rvert - q_j^{\text{soft}})$		-10	
`self_collisions`	# substeps with self-contact force $> 10\,\text{N}$		-10	
`joint_torques_normalized`	$\sum_j (\tau_j/\tau_j^{\max})^2$		-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 XI: Termination conditions and their curriculum schedule.

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

unlearnable.

APPENDIX 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 (Fig. 7A); `medium` and `hard` scale the same perturbation vector to test increasingly severe actuator mismatch. The values are listed in Table XII.

TABLE XII: **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	$1.5\times$	$2.0\times$	$2.5\times$

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.

Appendix F
World Model Details

For the data-efficiency comparison (Fig. 8), we include fully learned world model baselines inspired by RWM-U [32]; since a G1 implementation is not open-sourced, we set up an equivalent G1 velocity-tracking task in mjlab with the base and target environments above.

Overview. Every method is evaluated on a single `hard` target environment: omnidirectional base velocity tracking on flat ground from [32].

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 XIII). At policy-training time, a per-step epistemic-uncertainty signal formed from ensemble disagreement on next-state predictions is subtracted from the reward with MOPO penalty $\lambda = 1.0$ [32]. 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 `rsl_rl` algorithm and hyperparameters as the baseline (Table XIV). The outer loop runs for 5,000 iterations, following [32]; 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 $\beta\,\mathrm{KL}(\pi\|\pi_{\text{baseline}})$ 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 XIII: World-model architecture and training hyperparameters used for RWM-U on G1.

Parameter	Value
Architecture	
Backbone	GRU ensemble
Ensemble size	5
RNN hidden size	256
RNN layers	2
Head MLP (state / ext. / contact / term.)	[128]
Dimensions	
History horizon H	32
Forecast horizon F	8
Observation dim	99
Action dim	29
Contact channel dim	30
Data	
Dataset size	6M transitions $\approx$ 33 hours
File shard size	10,000 rows
Optimization	
Batch size	1024
Epochs	4
Optimizer	Adam
Learning rate	$1\times10^{-4} \rightarrow 1\times10^{-6}$ (cosine)
Weight decay	1×10^{-5}
Gradient steps (approx.)	1.8×10^{4}

Appendix G
Limitations

Our current Contact-UAN does not capture coupled actuator effects or motor-temperature-dependent effects [36]. 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.

Appendix H
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 [37, 38], with recent geometric, convex, differentiable, and exploration-based variants improving robustness and excitation [39–43]. Other approaches identify simulators as black boxes using trajectory-matching [15] or distribution-matching objectives [44]. These methods remain limited by the structure of the chosen parametric model. Domain randomization instead trains over parameter distributions [9, 10, 45–48], but wide randomization can yield conservative or inefficient policies [11, 12]. Online adaptation mitigates this by inferring latent dynamics at deployment [13, 14, 49–51]. Recent humanoid systems have therefore moved toward less randomization and more accurate models [1, 2, 15]. Our work follows this direction by replacing

TABLE XIV: 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)
Rollout collection		
Parallel envs	4096	4096
Steps per environment	24	24
Rollout source	**simulator**	**learned WM**
Outer iterations	**20,000**	**5,000**
Optimization		
Learning epochs	5	5
Mini-batches	4	4
Learning rate	10^{-3}	10^{-3}
Schedule	adaptive	adaptive
Max gradient norm	1.0	1.0
PPO objective		
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	$[256, 128, 128]$	$[256, 128, 128]$
RWM-U specific		
Uncertainty penalty λ	–	**1.0**
Termination threshold	–	**0.9**

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 [19, 20, 52], while RWM and RWM-U extend this paradigm to legged robots in online [18] and fully offline settings [32]. Such models reduce the structural bias of parametric simulators but require large datasets and mechanisms to prevent exploitation outside the data distribution [17]. 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 [21–26, 53]. 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 [41]. 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 [27, 54]. 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.