Visual Imitation Enables Contextual Humanoid Control

Arthur Allshire* Hongsuk Choi* Junyi Zhang* David McAllister* Anthony Zhang Chung Min Kim
Trevor Darrell Pieter Abbeel Jitendra Malik Angjoo Kanazawa

UC Berkeley

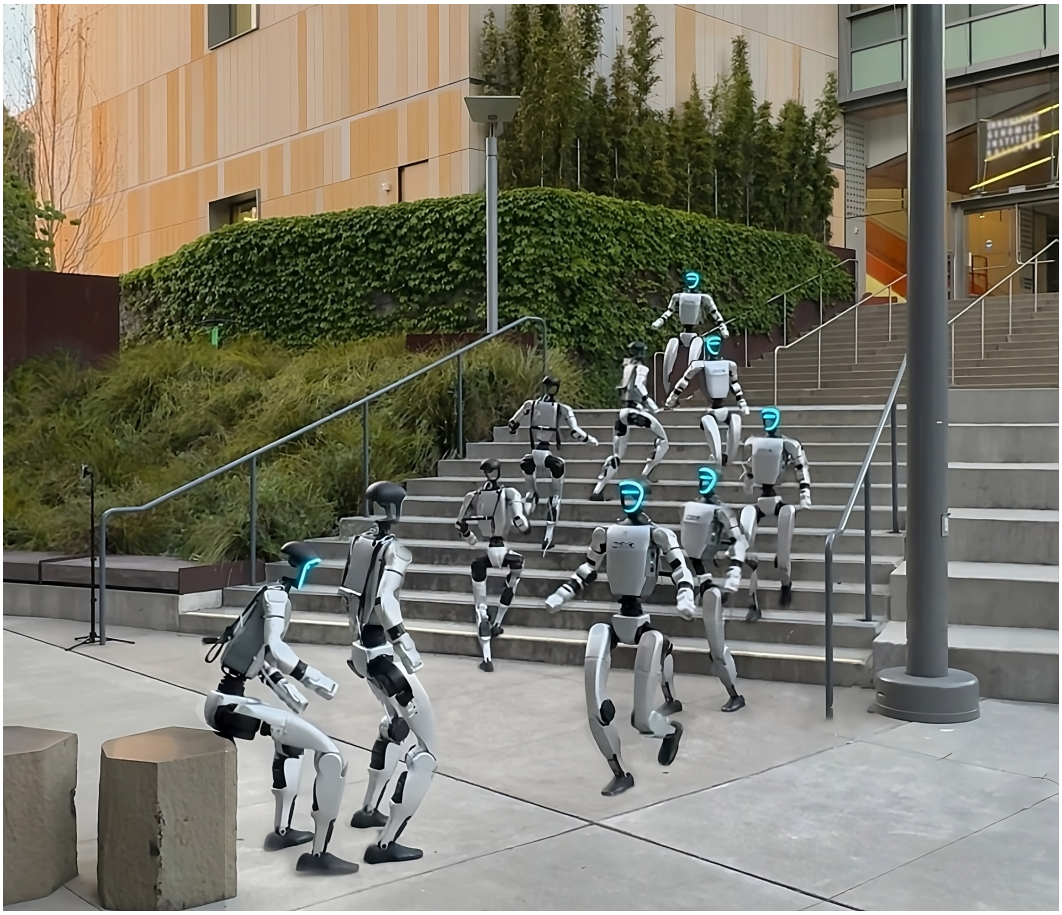


Figure 1: **VIDEOMIMIC** is a real-to-sim-to-real pipeline that converts monocular videos into transferable humanoid skills, letting robots learn context-aware behaviors (terrain-traversing, stairs-climbing, sitting) in a single policy. Video results are available on our webpage: https://videomimic.net.

Abstract: How can we teach humanoids to climb staircases and sit on chairs using the surrounding environment context? Arguably, the simplest way is to *just show them*—casually capture a human motion video and feed it to humanoids. We introduce VIDEOMIMIC, a real-to-sim-to-real pipeline that mines everyday videos, jointly reconstructs the humans and the environment, and produces whole-body control policies for humanoid robots that perform the corresponding skills. We demonstrate the results of our pipeline on real humanoid robots, showing robust, repeatable contextual control such as staircase ascents and descents, sitting and standing from chairs and benches, as well as other dynamic whole-body skills—all from a single policy, conditioned on the environment and global root commands. VIDEOMIMIC offers a scalable path towards teaching humanoids to operate in diverse real-world environments.

*These authors contributed equally. Correspondence to: allshire@berkeley.edu, hongsuk@berkeley.edu.

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

1 Introduction

How do we learn to interact with the world around us—like sitting on a chair or climbing a staircase? We watch others perform these actions, try them ourselves, and gradually build up the skill. Over time, we can handle new chairs and staircases, even if we have not seen those exact ones before. If humanoid robots could learn in this way—by observing everyday videos—they could acquire diverse contextual whole-body skills without relying on hand-tuned rewards or motion-capture data for each new behavior and environment. We refer to this ability to execute environment-appropriate actions as contextual control.

We introduce VIDEOMIMIC, a real-to-sim-to-real pipeline that turns monocular videos—such as casual smartphone captures—into transferable skills for humanoids. From these videos, we jointly recover the 4D human-scene geometry, retarget the motion to a humanoid, and train an RL policy to track the reference trajectories. We then distill the policy into a single unified policy that observes only proprioception, a local height-map, and the desired root direction. This distilled policy outputs low-level motor actions conditioned on the terrain and body state, allowing it to execute appropriate behaviors—such as stepping, climbing, or sitting—across unseen environments without explicit task labels or skill selection.

We develop a perception module that reconstructs 3D human motion from a monocular RGB video, along with aligned scene point clouds in the world coordinate frame. We convert the point clouds into meshes and align them with gravity to ensure compatibility with physics simulators. The global motion and local poses are retargeted to a humanoid with constraints that ensure physical plausibility, accounting for the embodiment gap. The mesh and retargeted data seed a goal-conditioned DeepMimic [1]-style reinforcement-learning phase in simulation: we warm-start on MoCap data, then train a single policy to track motions from multiple videos in their respective height-mapped environments while randomizing mass, friction, latency, and sensor noise for robustness. Once our tracking policy is trained, we distill it using DAgger [2] to a policy that operates without conditioning on target joint angles. The new policy observes proprioception, an 11×11 height-map patch centered on the torso, and the vector to the goal in the robot's local reference frame. PPO fine-tuning under this reduced observation set yields a generalist controller that, given height-map and root direction at test time, selects and smoothly executes context-appropriate actions such as stepping, climbing, or sitting. In particular, every step of our policy relies only on observations available at real-world deployment, making it immediately runnable on real hardware.

Our approach bridges 4D video reconstruction and robot skill learning in a single, data-driven loop. Unlike earlier work that recovers only the person or the scene in isolation, we jointly reconstruct both at a physically meaningful scale and represent them as meshes and motion trajectories suitable for physics-based policy learning. We train our approach on 123 monocular RGB videos, which will be released. We validate the approach through deployment on a real Unitree G1 robot, which shows generalized humanoid motor skills in the context of surrounding environments, even on unseen environments. We will release the reconstruction code, policy training framework, and the video dataset to facilitate future research.

2 Related Work

Learning Skills on Legged Robots. Recent progress in legged-robot motor skills follows two complementary streams. Reward-based methods use model-free RL in simulation, shaping behavior with handcrafted objectives that mix task terms (e.g., velocity tracking) and motion-naturalness regularizers; thanks to massive parallel

Method	Env. Real-to-Sim	Context. Ctrl	Real Robot
DeepMimic / SfV [1, 3]	✗	✗	✗
Egocentric Loco [4]	✗	✓	✓
ASAP [5]	✗	✗	✓
Humanoid Loco. [6]	✗	✗	✓
H2O / ExBody2 [7, 8]	✗	✗	✓
Parkour [9, 10]	✗	✓	✓
VideoMimic (Ours)	✓	✓	✓

Table 1: **Comparison of methods across different features.** VIDEOMIMIC transfers both human motion and scene geometry from real videos to simulation, learns context-aware control in simulation, and successfully deploys the resulting policy on real-world environments.

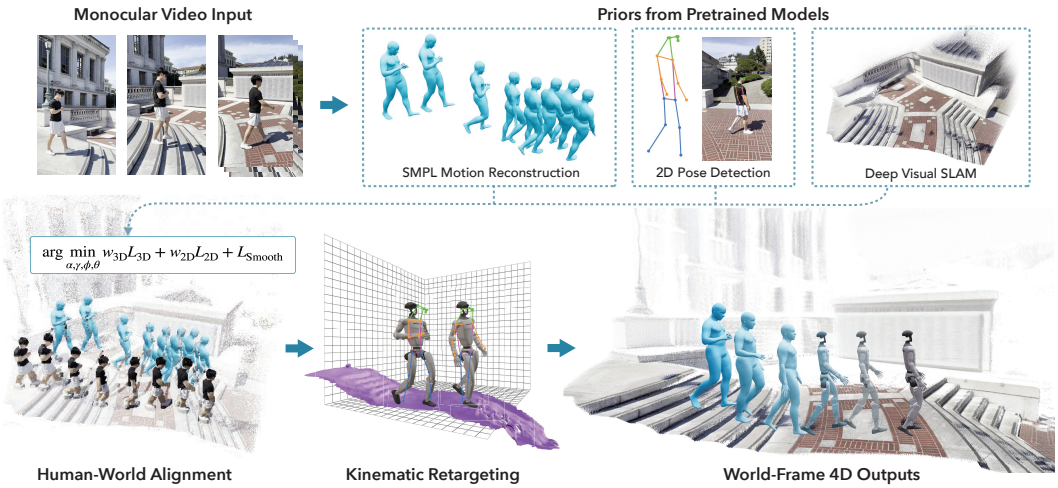


Figure 2: **VideoMimic Real-to-Sim.** A casually captured phone video provides the only input. We first reconstruct per-frame human motion and 2D keypoints, along with a dense scene point cloud. An efficient optimization jointly aligns the motion and point cloud, recovers statistically accurate metric scale using a human height prior, and registers the human trajectory based on human-associated points. The point cloud is then converted to a mesh, aligned with gravity, and the motion is retargeted to a humanoid in the reconstructed scene. This yields world-frame trajectories and simulator-ready meshes that serve as inputs for policy training.

physics engines [11, 12], this paradigm has produced agile locomotion on quadrupeds and humanoids without motion data. However, each new behavior demands tuning of user-defined rewards and environment scripting [13, 14, 15, 4, 6, 9, 16, 17]. Data-driven methods instead imitate reference motion, originally MoCap clips or monocular video, training a simulated character to track them and porting the idea to robots [1, 3, 18, 19, 7, 8]. For example, recent work [20, 21] frames legged locomotion as a next- token prediction task and pre-trains a policy on human data in kinematic space, showing strong performance. While imitation bypasses reward engineering [5], existing works typically assume flat ground or manually designed setups, limiting context-aware whole-body control; even animation systems that model human-scene contact rely on instrumented MoCap stages and thus lack scalability [22, 23]. Our system conditions on visual observations, the local height-maps, and learns environment-aware skills such as stair-climbing and chair-sitting directly from monocular RGB videos. Joint 4D human–scene reconstruction provides physically consistent reference motions, which RL distills into policies that transfer to a real humanoid (Table 1).

Human and Scene Reconstruction from Images and Videos. Early monocular-video methods regress pose and shape of humans [24] in a camera-relative frame with deep networks [24, 25, 26], which suffices for rendering, action recognition, or single-person tracking [27, 28, 29, 30, 31] but leaves the global trajectory—and thus context-aware dynamics—undefined; pioneers like SfV hand-tuned a global scale and even assumed a static camera, limiting generality. Recent methods combine human motion priors with SfM/SLAM to recover metric trajectories [32, 33, 34], yet still model only the person and camera. Advances in general scene parsing [35, 36, 37] have enabled joint human-scene reconstruction that resolves scale via multi-view cues or learned priors [38, 39], but these systems have not been validated on robots. Parallel work injects physics constraints in post-processing or simulation [40, 41, 42, 43, 44], trading scalability for realism. Our pipeline unifies these threads: it simultaneously estimates metric human motion and surrounding geometry from in-the-wild videos—without MoCap, pre-scanned scenes, or reward engineering—and outputs simulator-ready trajectories that respect contacts and collisions, enabling scalable learning of whole-body humanoid skills.

3 Real-to-Sim Data Acquisition

Our real-to-sim pipeline proceeds as summarized in Figure 2. We extract per-frame human poses and a raw scene point cloud from the input video (Sec. 3.1); jointly optimize them to obtain metrically

aligned human trajectories and scene geometry (Sec. 3.2); apply gravity alignment and convert the filtered point cloud into a lightweight mesh (Sec. 3.3); and retarget the refined trajectories to the humanoid under joint-limit, contact, and collision constraints (Sec. A.3). The resulting motion–mesh pairs are ready for policy learning in Sec. 4.

3.1 Preprocessing

We preprocess monocular RGB videos with off-the-shelf state-of-the-art human pose estimation and SfM methods. First, people are detected and associated across frames using Grounded SAM2 [45, 46]. For each detected person, we recover per-frame 3D SMPL [24] parameters with VIMO [47], obtaining per-frame local pose θ^t, shape β, and SMPL 3D joints $J^t_{\text{3D}} \in \mathbb{R}^{J\times 3}$. We detect 2D keypoints J^t_{2D}, i.e., body joint pixel positions, with ViTPose [48]. Foot contact is regressed by BSTRO [49]. For scene reconstruction, we obtain the world point cloud from either MegaSaM [35] or MonST3R [36], which is parameterized as per-frame depth D^t, camera pose $[R^t|t^t]$, and a shared camera intrinsic matrix K. Note that the resulting point cloud is not metrically accurate.

To coarsely position the person in the world frame, we follow the initialization strategy of SLAHMR [32], using (i) the camera focal length predicted by SfM and (ii) the ratio between the average 2D limb length from the ViTPose detections $\tilde{J}^t_{\text{2D}}$ and the corresponding metric scale 3D limb length in J^t_{3D}, we estimate a similarity factor per frame that yields a coarse global trajectory $(\phi^{t_0}, \gamma^{t_0})$. Separately, we also lift $\tilde{J}^t_{\text{2D}}$ to 3D by un-projecting each pixel (u, v) with its depth $D^t_{u,v}$ and intrinsics K from SfM: $\tilde{J}^t_{\text{3D}} = K^{-1}[u, v, 1]^\top D^t_{u,v}$. The lifted joints are then used to jointly optimize human poses and scene geometry scale, as described in the following section.

3.2 Joint Human–Scene Reconstruction

Our pipeline jointly optimizes the human trajectory and the scene scale. The variables are the humans' global translations $\gamma^{1:T}$, global orientations $\phi^{1:T}$, local poses $\theta^{1:T}$, and the scene point-cloud scale α. Because MegaSam pointclouds are scale-ambiguous, the metric human height prior in the SMPL body models serves as the metric reference, while the lifted joints $\tilde{J}^t_{\text{3D}}$ refine both the global trajectory and the local pose. We therefore solve for α simultaneously, reconciling any residual mismatch between the human-derived scale and the scene geometry.

Inspired by He et al. [7], we optionally run a scale-adaptation pass that searches for an SMPL shape $\beta^\star$ whose height and limb proportions match those of the G1 robot, prior to joint human-scene optimization. The SMPL joints are then extracted from the reshaped mesh. This use of a prefitted G1-scale SMPL β effectively rescales the scene geometry to G1 size, improving the feasibility of humanoid motion—e.g., enabling actions like running or climbing over large obstacles—and facilitating reference motion learning. For real-world deployment, we skip this step and operate directly on the original metric-scale scene.

The objective combines joint-distance losses in 3D (L_{3D}), computed as the L1 distance between $\tilde{J}^t_{\text{3D}}$ and J^t_{3D}, and 2D projection losses (L_{2D}), along with a temporal smoothness regularizer (L_{Smooth}) that discourages frame-to-frame jitter:

$$\arg\min_{\alpha,\gamma,\phi,\theta} \; w_{\text{3D}} L_{\text{3D}} + w_{\text{2D}} L_{\text{2D}} + L_{\text{Smooth}}.$$

We optimize this objective with a Levenberg–Marquardt solver implemented in JAX [50]. Running on an NVIDIA A100 GPU, the optimizer processes a 300-frame sequence in approximately 20 ms after compilation. Please refer to Sec. A for more details.

3.3 Generating Simulation-Ready Data

To deploy the monocular reconstruction in a physics engine, we (i) align it with real-world gravity using GeoCalib [51] and (ii) convert the noisy, dense point cloud into a lightweight mesh that imposes meaningful geometric constraints and supports memory-efficient parallel training. We use NKSR [52] for meshification. Please see Sec. A.3 for more details.

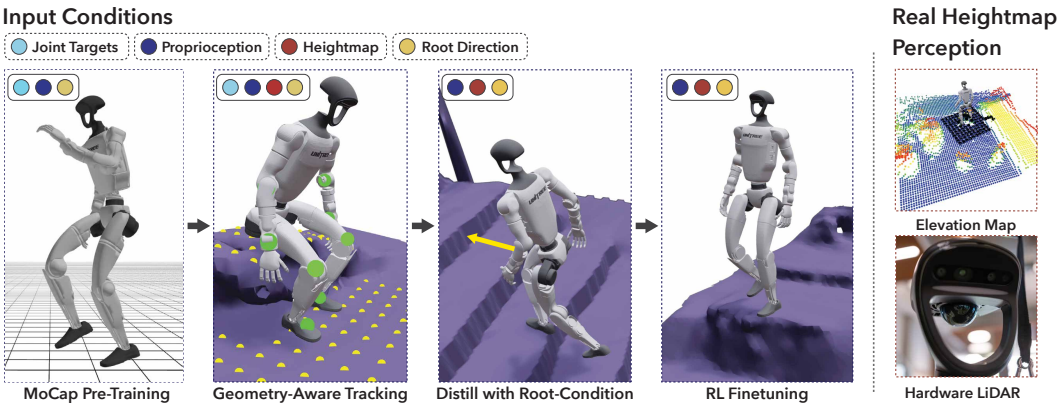


Figure 3: **Policy training in sim.** Our pipeline of training RL starts with a dataset of Motion Capture trajectories. We then inject a heightmap observation and track whole-body reference trajectories from our videos in various environments. We proceed to distill a policy conditioned only on the root position of the robot. We then finetune this policy directly with RL using the same reduced observation set. Our pipeline is motivated by three goals: (a) producing motions that are fast and faithful to the original video demonstrations; (b) ensuring observations are available in real-world settings; and (c) training a generalist policy that distills knowledge from all video demonstrations into a single model applicable beyond the training set.

4 Policy Learning

Given the kinematic reference from our clips and scenes, our policy learning pipeline produces a context-conditioned policy that can perform skills from the references when prompted by the appropriate environmental context. Figure 3 gives an overview of our pipeline, detailed below.

Policy Learning. We use Proximal Policy Optimization [53] implementation from Rudin et al. [54] for training our policy. Our learning takes place in the IsaacGym simulator [12]. Please refer to Sec. B for more details on our formulation and hyperparameters.

Observations. Our policies are conditioned on both proprioceptive and target-related observations. The proprioceptive inputs include a history of the robot's joint positions (q), joint velocities ($\dot{q}$), angular velocity (ω), projected gravity vector (g), and previous actions (a^{t-1}); we use a history length of 5 in practice. In addition, the policy receives local target observations: the target joint angles, target root roll and pitch, and the desired root direction, specified by relative x-y offset and yaw angle between the robot's current root position and the target root, all expressed in the robot's local frame. For policies conditioned on heightmaps, we further provide an elevation map around the torso. This is represented as an 11×11 grid sampled at 0.1m intervals, which captures local terrain geometry. Finally, the critic receives additional privileged observations, which are detailed in Table 3.

Batched Tracking. Our system utilizes a batched variant of DeepMimic [1] in order to learn to imitate motions using RL. We implement Reference State Initialization [1] in addition to motion load balancing similar to Tessler et al. [55], upweighting motions with a lower success rate.

Rewards. Our RL reward is designed entirely around data-driven tracking terms—specifically, link and joint positions, joint velocities, and foot-contact signals—so that raw demonstrations can be translated into physically executable motions with minimal hand-tuning. We have two objectives: (1) reducing reliance on manually crafted priors that are typically introduced through reward engineering, and (2) ensuring physical feasibility of the resulting motions. These two goals can conflict: because the reference trajectories are purely kinematic data from humans, exact tracking may result in non-physical motion. We therefore introduce an action-rate penalty along with several other penalty criteria designed to discourage exploiting simulator physics. Full formulations and weights are detailed in Sec. B.1. We train our policy over the stages described below.

Stage 1: MoCap Pre-Training. MoCap pre-training lets a policy learn challenging skills from noisy video reconstruction while keeping hand-crafted priors to a minimum and bridging the human-to-

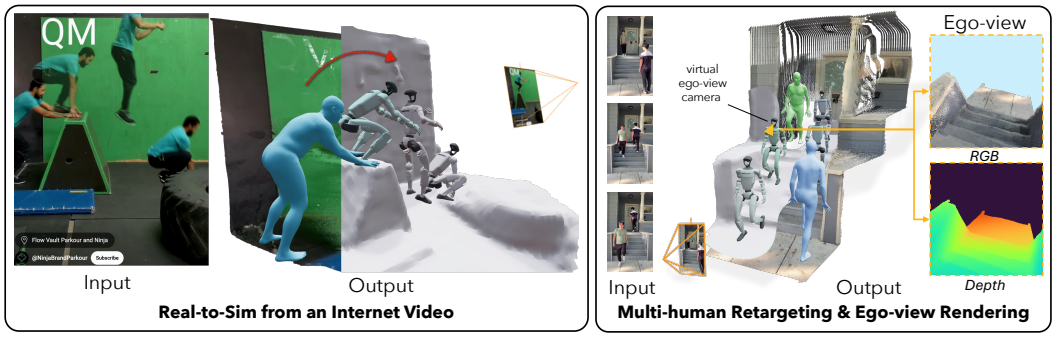


Figure 4: **Versatile capabilities of our Real-to-Sim pipeline.** VideoMimic enables (i) robust tracking of Internet videos with challenging motion and diverse environments, (ii) simultaneous reconstruction and retargeting of multiple humans, and (iii) ego-view RGB-D rendering for embodied perception—though not used in our current policy, it highlights the framework's broader applicability across inputs and tasks.

robot embodiment gap. Earlier work tackles this either by sampling start poses with multi-agent RL [3] or by having a privileged simulator imitate the motion [5]. Radosavovic et al. [21] and Singh et al. [56] instead employed a form of kinematic pre-training on human data. We adopt a simpler yet effective strategy: first pre-train the policy on MoCap trajectories, then fine-tune it on our reconstructed video data—both stages use reinforcement learning in a physics simulator. Even the MoCap-only policy can be deployed directly on the real robot as shown in Fig 9. We used LAFAN motion capture data [57] retargeted to Unitree G1. For the pretrained policies, the conditioning the policy receives is the target joint angles, target root roll/pitch, and desired root direction.

Stage 2: Scene-Conditioned Tracking. After MoCap pretraining, we initialize the policy from MPT checkpoints and introduce scene awareness by conditioning on the environment heightmap. The heightmap is integrated via a projection into the MPT policy's latent space residually with an initial weight of 0. We then randomly sample motions and perform DeepMimic-style tracking across reconstructed terrains. During this stage, the policy continues to receive motion-specific tracking conditioning, including target joint angles, root roll/pitch, and desired root directions.

Stage 3: Distillation. Following the stage of batched tracking, we distill via DAgger [2] to a policy that does not observe target joint angles or root roll/pitch observations. We are then able to use the desired root directions observations as conditioning signals to control the robot's position, which can be fed either from a joystick or potentially a path provided by a high-level controller. In this way, our framework unifies the previously separate approaches of joystick tracking and global reference following. Our distilled policy benefits from the fact that the teacher is also trained with observation randomization, hence it learns actions under some uncertainty, which would not be the case if we started by training with full body observations; this has been shown to be helpful in other contexts with policy learning [58].

Stage 4: Under-conditioned RL Finetuning. After distilling our policies to be exclusively conditioned on the root of our trajectories, we perform another round of RL. This is because behaviours which are learned conditioned on target joints may be sub-optimal for policies which are not conditioned on such targets. In practice, we found that this can significantly boost performance as compared to distilled policies. It also makes it possible to add lower-quality reference motions to the reference set since removing targets from the actor in effect makes it a "data-driven" reward signal with an under-constrained actor which is able to follow references appropriate to context.

5 Results

We demonstrate that humanoid robots can learn context-aware skills in diverse environments by imitating everyday human videos. We first evaluate the robustness of our reconstruction pipeline against baselines. Next, we demonstrate its versatility, highlighting its potential impact on future research. We then detail our curated video dataset. Finally, we ablate the MPT component and present demonstrations successfully transferred from simulation policies to a physical robot.

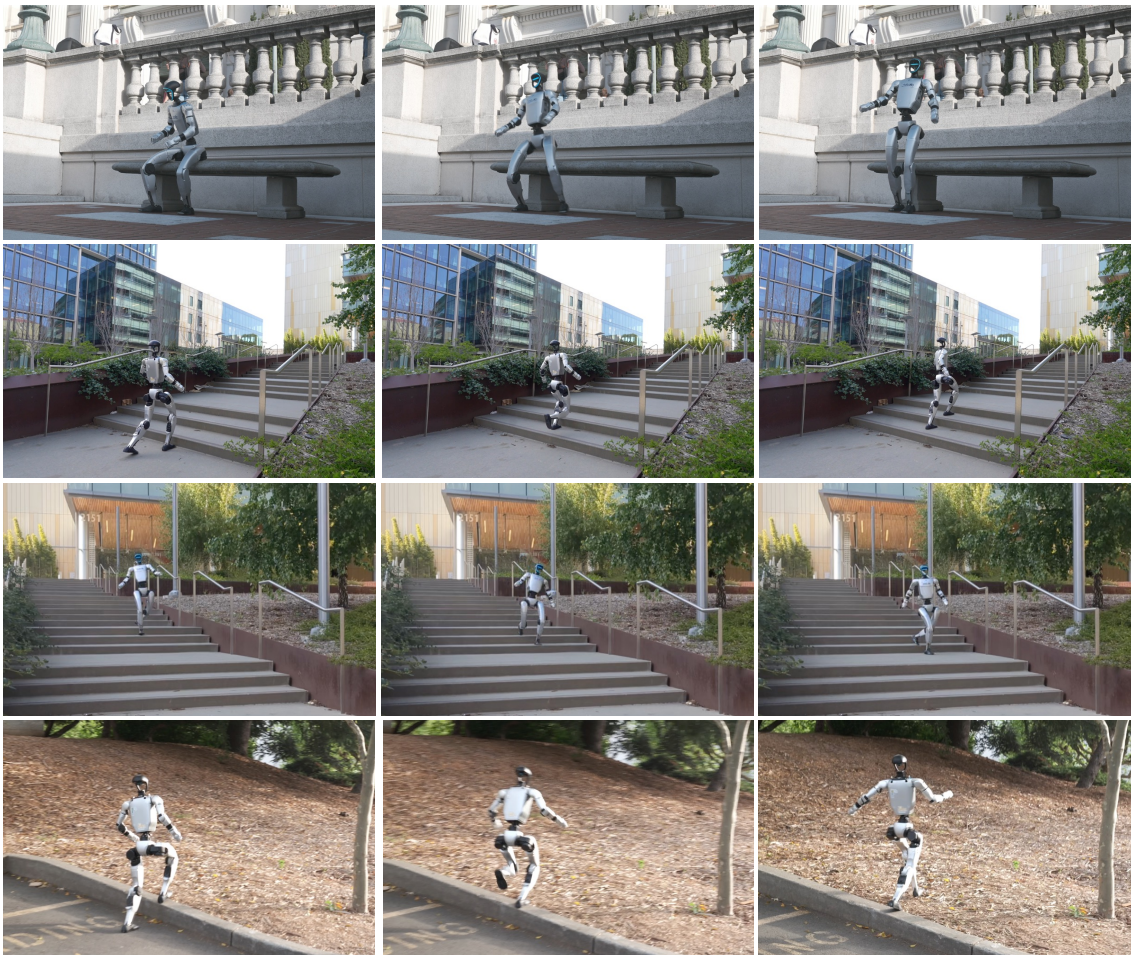


Figure 5: **The policy performing various skills on the real robot:** traversing complex terrain, standing, and sitting. All these skills are in a single policy, which decides what to do based on the context of its heightmap and joystick direction input. *Top row*: the policy stands from a seated position after sitting down. *Second row*: the policy walks up a flight of stairs. *Third row*: the policy walks down a flight of stairs. *Bottom row*: the policy walks over a kerb and onto a rough terrain. Please find the video results on our webpage.

5.1 Reconstruction and Data

Evaluation. We evaluate the robustness of our reconstruction pipeline on a subset of the SLOPER4D dataset [59], assessing both human trajectory and scene geometry reconstruction.

We compare our method against baselines following the standard evaluation protocol [60, 47]. As summarized in Table 2, our method consistently achieves the best performance, outperforming prior work in both human trajectory accuracy (WA/W-MPJPE) and scene geometry (Chamfer Distance). Refer to Sec. A.4 for more evaluation details.

Methods	WA-MPJPE	W-MPJPE	Chamfer Distance
WHAM* [60]	189.29	1148.49	-
TRAM [47]	149.48	954.90	10.66
Ours	**112.13**	**696.62**	**0.75**

Table 2: **Comparison of Reconstruction.** *: WHAM does not recover the environment.

Versatility. Figure 4 and our webpage's Viser [61] visualizations highlight the breadth of our reconstruction pipeline, showcasing (i) robust environment reconstruction from an Internet video involving dynamic human-scene interaction, (ii) multi-human reconstruction and retargeting. Furthermore, the dense point cloud reconstruction enables ego-view RGB-D rendering via simple rasterization. While not used in our current policy, this offers a promising direction for future work—especially given the challenges of rendering naturalistic images in simulation.

Video Data. We curated 123 casually recorded smartphone videos of people performing everyday activities in diverse indoor and outdoor settings, including sitting, standing up from furniture, walking up/down stairs (even backwards), and stepping onto blocks. See Sec. B.8 for more details.

MPT ablation. We ablate the impact of pretraining on motion capture data. MPT has multiple effects: first, reference motions are noisy and thus harder to learn to track tabula rasa. Second, initial positions of the robot are often not entirely statically stable or may have some interpenetrations with the scene. Hence, MPT can help stabilize learning during the initial phases, whereas a policy from scratch may not even be able to learn how to balance. As shown in Figure 6, removing MPT significantly hinders the policy's ability to learn effective behaviors.

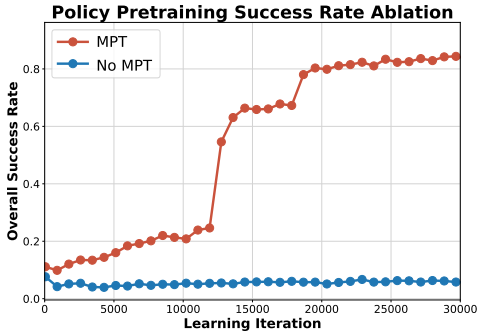


Figure 6: **Impact of MoCap pre-training (MPT).** Pretraining the policy on motion-capture data facilitates learning on video captures despite noisy references.

5.2 Real-world Deployment

Setup. We deploy our controller on a 23-DoF *Unitree G1* humanoid and run it onboard at 50 Hz. Following [62], we set relatively low joint gains, $K_p = 75$, to avoid excessively fast or overly stiff behaviour—which helps to avoid excessively violent contact when the robot makes heavy contact with objects such as chairs or stairs. LiDAR odometry and height-maps are computed in real time using Fast-lio2 [63] and probabilistic terrain mapping [64, 17]. The odometry is used to align the local height-map observations into the robot's frame. We feed joystick targets from a human operator. Including policy running, all operations are run onboard. We found two critical ingredients for successful motion deployment through iterative sim-to-real trials: (i) relaxing the episode-termination tolerances with respect to the reference motion, and (ii) injecting realistic physics perturbations during training. Full details are given in Sec. C.

Real-world evaluation. Figure 5 and the accompanying video showcase the policy executing a wide range of whole-body behaviors on the Unitree G1. Without any task-specific tuning, the same network—driven only by proprioception and a noisy LiDAR height-map that provides a full 360° view around the torso—climbs and descends indoor and outdoor staircases, traverses steep earthen slopes and rough vegetation, and reliably sits down on or stands up from chairs and benches. The controller is surprisingly resilient: after unexpected foot slides while descending stairs, it recovers by momentarily hopping on a single leg before regaining nominal gait.

To the best of our knowledge, this is the *first* real-world deployment of a *context-aware* humanoid policy learned from monocular human videos, jointly demonstrating perceptive locomotion and environment-prompted whole-body skills such as sitting, standing, and climbing stairs. Additional qualitative results are available on the project webpage.

6 Conclusion

We introduced VIDEOMIMIC, a real-to-sim-to-real pipeline that converts everyday human videos into environment-conditioned control policies for humanoids. The system (i) reconstructs humans and surrounding geometry from monocular clips, (ii) retargets the motion to a kinematically feasible humanoid, and (iii) uses the recovered scene as task terrain for dynamics-aware RL. The result is a *single* policy that delivers robust, repeatable contextual control—e.g., stair ascents/descents and chair sit-stand—all driven only by the environment geometry and a root direction command. VIDEOMIMIC offers a scalable path for teaching humanoids contextual skills directly from videos. We expect future work to extend the system to richer human–environment interactions, multi-modal sensor-based context learning, and multi-agent behavior modeling, among other directions.

7 Limitations

Our pipeline delivers encouraging real-world results, yet several practical weaknesses remain.

Reconstruction. Monocular 4D human–scene recovery is still brittle in the wild. Camera pose drift in MegaSaM often yields duplicate "ghost" layers of the same surface. Due to its inability to refine the dynamic points, the dynamic points from the person are mistakenly fused into the static point cloud or inaccurately placed (e.g., feet buried beneath the environment). In particular, we found that MegaSaM performs poorly on images with low texture. Depth filtering and spatio-temporal subsampling remove many outlier points, but aggressive thresholds leave holes that hinder meshing. NKSR mitigates noise, yet may oversmooth fine geometry (e.g., narrow stair treads); such high-frequency details are crucial for robot control, and we discard videos where these details are missing after reconstruction. Also, during point-to-mesh conversion, spiky artifacts may appear due to stray points.

Retargeting. The kinematic optimizer assumes every reference pose can be made feasible once scaled to the robot. In cluttered scenes, this is not always true, and conflicting costs—strict foot-contact matching versus collision avoidance—can trap the solver in poor local minima that the RL controller must subsequently "clean up."

Sensing and policy input. At test time, the controller receives only proprioception and an 11×11 LiDAR height-map. This coarse grid is adequate for terrain and chairs but lacks the resolution for precise contacts, manipulation, or reasoning about overhanging obstacles. Incorporating richer perceptual inputs—such as RGB-D data or learned occupancy grids—would likely broaden the method's applicability and improve its semantic understanding of the environment.

Simulation fidelity. We assume the scene can be represented as a single rigid mesh. Scaling to articulated or deformable objects will require more expressive simulators and object-level reconstruction pipelines—open problems for future work.

Data scale and motion quality. The distilled policy is trained on only 123 video clips and occasionally relies on recovery behaviors, leading to jerky motions. Larger, more diverse video corpora and iterative real-world fine-tuning should improve smoothness and robustness.

Moving beyond these limitations—through better dynamic static separation, hole-resistant meshing, adaptive retargeting costs, richer perception, and larger datasets—is a key direction for future work.

Acknowledgments

We thank Brent Yi for his guidance with the excellent 3D visualization tool we use, Viser. We are grateful to Ritvik Singh, Jason Liu, Ankur Handa, Ilija Radosavovic, Himanshu Gaurav-Singh, Haven Feng, and Kevin Zakka for helpful advice and discussions during the paper. We thank Lea Müller for helpful discussions at the start of the project. We thank Zhizheng Liu for helpful suggestions on evaluating human and scene reconstruction. We thank Moji Shi and Huayi Wang for advice on using LiDAR heightmap input. We thank Eric Xu, Matthew Liu, Hayeon Jeong, Hyunjoo Lee, Jihoon Choi, Tyler Bonnen, and Yao Tang for their help in capturing and featuring in the video clips used in this project. We acknowledge support from the BAIR Humanoid Intelligence Center. Chung Min Kim is supported by the NSF Research Fellowship Program, Grant DGE 2146752. Pieter Abbeel holds concurrent appointments as a Professor at UC Berkeley and as an Amazon Scholar. This paper describes work performed at UC Berkeley and is not associated with Amazon. This project was funded in part by NSF: CNS-2235013, IARPA DOI/IBC No. 140D0423C0035, ONR MURI awards N00014-21-1-2801 and W911NF-23-1-0277, and Bakar fellows.

References

[1] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. DeepMimic. *ACM Transactions on Graphics*, 37(4):1–14, jul 2018. doi:10.1145/3197517.3201311. URL https://doi.org/10.1145%2F3197517.3201311.

[2] S. Ross, G. J. Gordon, and J. A. Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010. URL http://arxiv.org/abs/1011.0686.

[3] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine. SFV: reinforcement learning of physical skills from videos. *CoRR*, abs/1810.03599, 2018. URL http://arxiv.org/abs/1810.03599.

[4] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged locomotion in challenging terrains using egocentric vision, 2022. URL https://arxiv.org/abs/2211.07638.

[5] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan, Z. Yi, G. Qu, K. Kitani, J. Hodgins, L. J. Fan, Y. Zhu, C. Liu, and G. Shi. Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills, 2025. URL https://arxiv.org/abs/2502.01143.

[6] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Real-world humanoid locomotion with reinforcement learning, 2023. URL https://arxiv.org/abs/2303.03381.

[7] T. He, Z. Luo, W. Xiao, C. Zhang, K. Kitani, C. Liu, and G. Shi. Learning human-to-humanoid real-time whole-body teleoperation, 2024. URL https://arxiv.org/abs/2403.04436.

[8] M. Ji, X. Peng, F. Liu, J. Li, G. Yang, X. Cheng, and X. Wang. Exbody2: Advanced expressive humanoid whole-body control, 2025. URL https://arxiv.org/abs/2412.13196.

[9] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots, 2023. URL https://arxiv.org/abs/2306.14874.

[10] Z. Zhuang, S. Yao, and H. Zhao. Humanoid parkour learning, 2024. URL https://arxiv.org/abs/2406.10759.

[11] J. Hwangbo, J. Lee, and M. Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018. URL www.raisim.com.

[12] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning. *CoRR*, abs/2108.10470, 2021. URL https://arxiv.org/abs/2108.10470.

[13] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *CoRR*, abs/1901.08652, 2019. URL http://arxiv.org/abs/1901.08652.

[14] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Sci. Robotics*, 5(47):5986, 2020. doi:10.1126/scirobotics.abc5986. URL https://doi.org/10.1126/scirobotics.abc5986.

[15] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: rapid motor adaptation for legged robots. *CoRR*, abs/2107.04034, 2021. URL https://arxiv.org/abs/2107.04034.

[16] H. Wang, Z. Wang, J. Ren, Q. Ben, T. Huang, W. Zhang, and J. Pang. Beamdojo: Learning agile humanoid locomotion on sparse footholds. In *Robotics: Science and Systems (RSS)*, 2025.

[17] J. Long, J. Ren, M. Shi, Z. Wang, T. Huang, P. Luo, and J. Pang. Learning humanoid locomotion with perceptive internal model. *arXiv preprint arXiv:2411.14386*, 2024.

[18] R. Yu, H. Park, and J. Lee. Human dynamics from monocular video with dynamic camera movements. *ACM Transactions on Graphics (TOG)*, 40(6):1–14, 2021.

[19] Z. Luo, J. Cao, J. Merel, A. Winkler, J. Huang, K. M. Kitani, and W. Xu. Universal humanoid motion representations for physics-based control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=OrOd8PxOO2.

[20] I. Radosavovic, S. Kamat, T. Darrell, and J. Malik. Learning humanoid locomotion over challenging terrain. *arXiv preprint arXiv:2410.03654*, 2024.

[21] I. Radosavovic, B. Zhang, B. Shi, J. Rajasegaran, S. Kamat, T. Darrell, K. Sreenath, and J. Malik. Humanoid locomotion as next token prediction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[22] M. Hassan, Y. Guo, T. Wang, M. Black, S. Fidler, and X. B. Peng. Synthesizing physical character-scene interactions. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–9, 2023.

[23] L. Pan, Z. Yang, Z. Dou, W. Wang, B. Huang, B. Dai, T. Komura, and J. Wang. Tokenhsi: Unified synthesis of physical human-scene interactions through task tokenization, 2025. URL https://arxiv.org/abs/2503.19901.

[24] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 851–866. 2023.

[25] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7122–7131, 2018.

[26] M. Kocabas, N. Athanasiou, and M. J. Black. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5253–5263, 2020.

[27] S. Peng, Y. Zhang, Y. Xu, Q. Wang, Q. Shuai, H. Bao, and X. Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9054–9063, 2021.

[28] G. Moon, T. Shiratori, and S. Saito. Expressive whole-body 3d gaussian avatar. In *European Conference on Computer Vision*, pages 19–35. Springer, 2024.

[29] J. Rajasegaran, G. Pavlakos, A. Kanazawa, and J. Malik. Tracking people by predicting 3d appearance, location and pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2740–2749, 2022.

[30] D. C. Luvizon, D. Picard, and H. Tabia. 2d/3d pose estimation and action recognition using multitask deep learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5137–5146, 2018.

[31] J. Rajasegaran, G. Pavlakos, A. Kanazawa, C. Feichtenhofer, and J. Malik. On the benefits of 3d pose and tracking for human action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 640–649, 2023.

[32] V. Ye, G. Pavlakos, J. Malik, and A. Kanazawa. Decoupling human and camera motion from videos in the wild. 2023.

[33] Y. Yuan, U. Iqbal, P. Molchanov, K. Kitani, and J. Kautz. Glamr: Global occlusion-aware human mesh recovery with dynamic cameras. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11038–11049, 2022.

[34] M. Kocabas, Y. Yuan, P. Molchanov, Y. Guo, M. J. Black, O. Hilliges, J. Kautz, and U. Iqbal. Pace: Human and camera motion estimation from in-the-wild videos. In *2024 International Conference on 3D Vision (3DV)*, pages 397–408. IEEE, 2024.

[35] Z. Li, R. Tucker, F. Cole, Q. Wang, L. Jin, V. Ye, A. Kanazawa, A. Holynski, and N. Snavely. Megasam: Accurate, fast, and robust structure and motion from casual dynamic videos. *arXiv preprint arXiv:2412.04463*, 2024.

[36] J. Zhang, C. Herrmann, J. Hur, V. Jampani, T. Darrell, F. Cole, D. Sun, and M.-H. Yang. Monst3r: A simple approach for estimating geometry in the presence of motion. *arXiv preprint arXiv:2410.03825*, 2024.

[37] Q. Wang, Y. Zhang, A. Holynski, A. A. Efros, and A. Kanazawa. Continuous 3d perception model with persistent state. *arXiv preprint arXiv:2501.12387*, 2025.

[38] L. Müller, H. Choi, A. Zhang, B. Yi, J. Malik, and A. Kanazawa. Reconstructing people, places, and cameras. *arXiv preprint arXiv:2412.17806*, 2024.

[39] Z. Liu, J. Lin, W. Wu, and B. Zhou. Joint optimization for 4d human-scene reconstruction in the wild. *arXiv preprint arXiv:2501.02158*, 2025.

[40] Y. Yuan, S.-E. Wei, T. Simon, K. Kitani, and J. Saragih. Simpoe: Simulated character control for 3d human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7159–7169, 2021.

[41] Y. Yuan, V. Makoviychuk, Y. Guo, S. Fidler, X. Peng, and K. Fatahalian. Learning physically simulated tennis skills from broadcast videos. *ACM Trans. Graph*, 42(4), 2023.

[42] N. Ugrinovic, B. Pan, G. Pavlakos, D. Paschalidou, B. Shen, J. Sanchez-Riera, F. Moreno-Noguer, and L. Guibas. Multiphys: multi-person physics-aware 3d motion estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2331–2340, 2024.

[43] S. Zhang, Y. Zhang, F. Bogo, M. Pollefeys, and S. Tang. Learning motion priors for 4d human body capture in 3d scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11343–11353, 2021.

[44] J. Li, S. Bian, C. Xu, G. Liu, G. Yu, and C. Lu. D &d: Learning human dynamics from dynamic camera. In *European Conference on Computer Vision*, pages 479–496. Springer, 2022.

[45] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. URL https://arxiv.org/abs/2408.00714.

[46] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, Z. Zeng, H. Zhang, F. Li, J. Yang, H. Li, Q. Jiang, and L. Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024.

[47] Y. Wang, Z. Wang, L. Liu, and K. Daniilidis. Tram: Global trajectory and motion of 3d humans from in-the-wild videos. *arXiv preprint arXiv:2403.17346*, 2024.

[48] Y. Xu, J. Zhang, Q. Zhang, and D. Tao. Vitpose: Simple vision transformer baselines for human pose estimation. *Advances in neural information processing systems*, 35:38571–38584, 2022.

[49] C.-H. P. Huang, H. Yi, M. Höschle, M. Safroshkin, T. Alexiadis, S. Polikovsky, D. Scharstein, and M. J. Black. Capturing and inferring dense full-body human-scene contact. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 13274–13285, June 2022.

[50] B. Yi, V. Ye, M. Zheng, Y. Li, L. Müller, G. Pavlakos, Y. Ma, J. Malik, and A. Kanazawa. Estimating body and hand motion in an ego-sensed world. *arXiv preprint arXiv:2410.03665*, 2024.

[51] A. Veicht, P.-E. Sarlin, P. Lindenberger, and M. Pollefeys. Geocalib: Learning single-image calibration with geometric optimization. In *European Conference on Computer Vision*, pages 1–20. Springer, 2024.

[52] J. Huang, Z. Gojcic, M. Atzmon, O. Litany, S. Fidler, and F. Williams. Neural kernel surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4369–4379, 2023.

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

[54] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 91–100. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/rudin22a.html.

[55] C. Tessler, Y. Guo, O. Nabati, G. Chechik, and X. B. Peng. Maskedmimic: Unified physics-based character control through masked motion inpainting. *ACM Transactions on Graphics (TOG)*, 2024.

[56] H. G. Singh, A. Loquercio, C. Sferrazza, J. Wu, H. Qi, P. Abbeel, and J. Malik. Hand-object interaction pretraining from videos, 2024. URL https://arxiv.org/abs/2409.08273.

[57] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal. Robust motion in-betweening. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 39(4), 2020.

[58] T. G. W. Lum, M. Matak, V. Makoviychuk, A. Handa, A. Allshire, T. Hermans, N. D. Ratliff, and K. V. Wyk. Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics, 2024. URL https://arxiv.org/abs/2407.02274.

[59] Y. Dai, Y. Lin, X. Lin, C. Wen, L. Xu, H. Yi, S. Shen, Y. Ma, and C. Wang. Sloper4d: A scene-aware dataset for global 4d human pose estimation in urban environments. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 682–692, 2023.

[60] S. Shin, J. Kim, E. Halilaj, and M. J. Black. WHAM: Reconstructing world-grounded humans with accurate 3D motion. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[61] B. Yi, C. M. Kim, J. Kerr, G. Wu, R. Feng, A. Zhang, J. Kulhanek, H. Choi, Y. Ma, M. Tancik, et al. Viser: Imperative, web-based 3d visualization in python. *arXiv preprint arXiv:2507.22885*, 2025.

[62] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, C. Sferrazza, Y. Tassa, and P. Abbeel. Mujoco playground, 2025. URL https://arxiv.org/abs/2502.08844.

[63] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang. Fast-lio2: Fast direct lidar-inertial odometry, 2021. URL https://arxiv.org/abs/2107.06829.

[64] P. Fankhauser, M. Bloesch, and M. Hutter. Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):3019–3026, 2018. doi:10.1109/LRA.2018.2849506.

[65] J. Lu, T. Huang, P. Li, Z. Dou, C. Lin, Z. Cui, Z. Dong, S.-K. Yeung, W. Wang, and Y. Liu. Align3r: Aligned monocular depth estimation for dynamic videos. *arXiv preprint arXiv:2412.03079*, 2024.

[66] T. He, Z. Luo, X. He, W. Xiao, C. Zhang, W. Zhang, K. Kitani, C. Liu, and G. Shi. Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning, 2024. URL https://arxiv.org/abs/2406.08858.

[67] T. Cheynel, T. Rossi, B. Bellot-Gurlet, D. Rohmer, and M.-P. Cani. Sparse motion semantics for contact-aware retargeting. In *ACM SIGGRAPH Conference on Motion, Interaction and Games (MIG)*, 2023.

[68] C. M. Kim, B. Yi, H. Choi, Y. Ma, K. Goldberg, and A. Kanazawa. Pyroki: A modular toolkit for robot kinematic optimization, 2025. URL https://arxiv.org/abs/2505.03728.

[69] Y. Li, Y. Lin, J. Cui, T. Liu, W. Liang, Y. Zhu, and S. Huang. Clone: Closed-loop whole-body humanoid teleoperation for long-horizon tasks, 2025. URL https://arxiv.org/abs/2506.08931.

[70] D. Makoviichuk and V. Makoviychuk. rl-games: A high-performance framework for reinforcement learning. https://github.com/Denys88/rl_games, May 2021.

A Real-to-Sim Details

Our goal is to endow a humanoid with whole-body skills—walking, climbing, sitting—that account for the surrounding geometry after watching a collection of monocular RGB videos of humans performing such skills. We assume (i) a monocular RGB video that clearly captures both the person and the scene; (ii) a static environment during the clip so human motion and terrain can be treated as rigid at training time; and (iii) known robot kinematics and joint limits, but no multi-view rig or motion-capture setup, depth sensors, or pre-scanned meshes. From the video, we jointly reconstruct metric-scale 4D human trajectories and dense scene geometry, align them to gravity, meshify scene point clouds, and retarget the kinematics to the robot while enforcing contacts and collisions. The resulting motion-and-mesh pairs serve as simulator-ready training clips.

A.1 Notations

Setup. Our method takes a monocular video sequence as input. We denote each video frame as I^t, with resolution $H \times W$. Given this input, along with initial estimates of camera parameters and human poses, our method jointly reconstructs global human motion trajectories and dense environmental geometry in a metric-scale 3D world.

Human. We represent reconstructed humans from a video using the SMPL model [24]. SMPL is a differentiable function mapping pose parameters $\theta \in \mathrm{SO}(3)^J$ and shape parameters $\beta \in \mathbb{R}^B$ to a mesh with J joints. The mesh is positioned in the world coordinate system by global orientation $\phi \in \mathrm{SO}(3)$ and translation $\gamma \in \mathbb{R}^3$. Thus, at frame t, a human is defined by:

$$\mathbf{P}^t = \phi^t, \theta^t, \beta^t, \gamma^t. \tag{1}$$

Camera. We assume a perspective camera model with intrinsics $K \in \mathbb{R}^{3\times3}$ and extrinsics defined by rotation $R \in \mathrm{SO}(3)$ and translation $t \in \mathbb{R}^3$. A 3D point $x^{\text{3D}} \in \mathbb{R}^3$ is first transformed into the camera frame and then projected onto the image plane as:

$$x_{\text{2D}} = \Pi\left(K\begin{bmatrix}R & t\end{bmatrix}\begin{bmatrix}x_{\text{3D}}\\ 1\end{bmatrix}\right), \tag{2}$$

where $\Pi : \mathbb{R}^3 \to \mathbb{R}^2$ denotes the perspective projection, defined as $\Pi\left(\begin{bmatrix}u\\v\\w\end{bmatrix}\right) = \left(\frac{u}{w}, \frac{v}{w}\right)$.

Scene. The scene is represented using dense per-frame depth and camera output by MegaSaM [35] or MonST3R [36]. Specifically, we use MegaSaM for scenes with richer textures where the correspondence-based Bundle-Adjustment is more reliable. For textureless scenes, we adopt MonST3R and its depth-conditioned variant [65] for better reconstruction results, although MegaSam generalizes better across a wider distribution of videos. To resolve scale ambiguity of the camera translation and scene geometry, we introduce a scaling parameter α. Given per-pixel depth $D_{i,j}$, the corresponding world coordinate $\mathcal{S}i, j$ for pixel (i, j) is obtained by unprojecting the points:

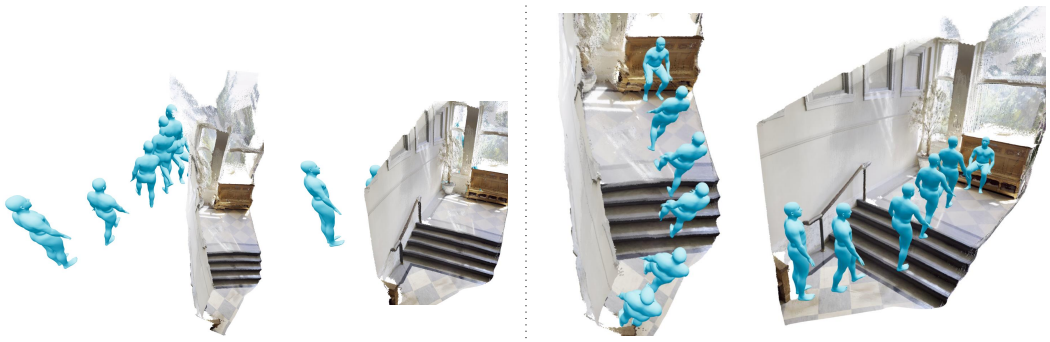
$$\mathcal{S}i, j = \alpha(R^\top[K^{-1}D_{i,j}[i, j, 1]^\top] - R^\top t). \tag{3}$$

A.2 Objectives for Joint Human–Scene Reconstruction

As shown in Figure 7a, the coarse initialization produces inaccurate human trajectories and an incorrect scene scale. Our optimization procedure refines these estimates by jointly optimizing the human's global translations ($\gamma^{1:T}$), global orientations ($\phi^{1:T}$), local poses ($\theta^{1:T}$), and the scene point cloud scale (α). The optimization aligns the human and world as shown in Figure 7b, and aids in generating simulation-ready data.

The objective combines joint-distance losses in 3D and through 2D projection with a temporal-smoothness regularizer, which discourages frame-to-frame jitter in both the root trajectory and the articulated pose:

$$\arg\min_{\alpha,\gamma,\phi,\theta} \; w_{\text{3D}}L_{\text{3D}} + w_{\text{2D}}L_{\text{2D}} + L_{\text{Smooth}}.$$



(a) Before (b) After

Figure 7: **Before and after optimization.** We visualize human trajectories and scene point clouds before and after optimization, showing each from two different viewpoints.

$$L_{\text{3D}} = \sum_t \left\| J^t_{\text{3D}} - \tilde{J}^t_{\text{3D}} \right\|_1, \tag{4}$$

$$L_{\text{2D}} = \sum_t \left\| \tilde{J}^t_{\text{2D}} - \Pi\Big(K \begin{bmatrix} R^t & t^t \end{bmatrix} \begin{bmatrix} J^t_{\text{3D}} \\ \mathbf{1} \end{bmatrix} \Big) \right\|_1, \tag{5}$$

$$L_{\text{Smooth}} = \lambda_\gamma \sum_t \left\| \gamma^t - \gamma^{t-1} \right\|_2^2 + \lambda_\theta \sum_t \left\| \theta^t - \theta^{t-1} \right\|_2^2, \tag{6}$$

where $\Pi : \mathbb{R}^3 \to \mathbb{R}^2$ denotes the perspective projection, defined as $\Pi\left(\begin{bmatrix} u \\ v \\ w \end{bmatrix}\right) = \left(\frac{u}{w}, \frac{v}{w}\right)$. We optimize this objective with a Levenberg–Marquardt solver implemented in JAX [50]. Running on an NVIDIA A100 GPU, the optimizer processes a 300-frame sequence in approximately 20 ms after compilation.

A.3 Generating Simulation Ready Data

Gravity alignment. We first estimate the gravity direction in the initial camera frame and define the transformation that converts the 3D reconstruction from world coordinates to a gravity-aligned coordinate system compatible with physics engines. GeoCalib [51] provides the roll–pitch angles for that frame; the composite rotation $R_{\text{gravity,world}} = R_{y\uparrow\to z} R_x(\pi) R_z(\rho) R_{\text{cam,world}}$ re-orients the reconstruction so that $+z$ points up, ensuring consistency with gravity in physics engines. The transformation is applied to both human keypoints and static scene geometry.

Pointcloud filtering. For each world pointcloud we discard noisy background points and human pointclouds with thresholding on depth gradient, rotate by $R_{\text{gravity,world}}$, scale, and crop to a $\pm 2\,\text{m}$ box around the SMPL joints per frame. A $0.1\,\text{m}$ voxel grid then keeps at most 20 samples per occupied cell, shrinking the pointcloud to about 5% of its original size without losing surface detail.

Meshification. We first surface it using NKSR [52] to obtain a coarse mesh. Top-down ray casting fills large holes via inverse-distance interpolation inside the convex hull; merging the infilled points with the originals and re-running NKSR produces the final mesh. The entire pipeline processes a 300-frame sequence in roughly 60 s ($\sim$ 5 s for gravity alignment and $\sim$ 55 s for filtering and meshing).

Humanoid Motion Retargeting. Given the human trajectories and environment meshes as input, our objective is to transform these motions to the *robot*'s embodiment. We approach the retargeting task as an optimization problem similar to previous work [7, 66], but instead use a Levenberg-Marquardt solver to handle the highly nonlinear landscape of the human-in-scene problem setting.

We solve for the following variables: the G1 joint angles $q^{1:T}$, its root poses $(\phi_{\text{R}}^{1:T}, \gamma_{\text{R}}^{1:T})$, and the set of per-link scale factors s between the two embodiments. Note that s is *constant* across timesteps, and for all distance-related costs we penalize x, y, and z components independently (i.e., no norm).

Inspired by Cheynel et al. [67], we implement a kinematic tree-based motion transfer cost $L^t_{\text{motion}} = L^t_{\text{position}} + L^t_{\text{angle}}$:

$$L^t_{\text{position}} = \sum_{i \neq j} m_{ij} \left\| \Delta^{\text{SMPL}}_{ij}(t) - s \cdot \Delta^{\text{G1}}_{ij}(t) \right\|_2^2, \tag{7}$$

$$L^t_{\text{angle}} = \sum_{i \neq j} m_{ij} \left(1 - \left\langle \Delta^{\text{SMPL}}_{ij}(t),\ \Delta^{\text{G1}}_{ij}(t) \right\rangle\right), \tag{8}$$

where Δ_{ij} is the position difference between joints i and j. m_{ij} denotes if the joints are immediate neighbors in the robot's kinematic chain — 1 if true, 0 if not. These two costs are weighed equally. These terms are regularized to be close to 1.0 and stay non-negative.

We also ground the robot with the environment through a set of contact costs (e.g., foot contact point matching, foot skating penalty) and collision costs (self- and world-collision avoidance). The robot is approximated as a set of capsules, and the world as a heightmap. The learning-based feet contact estimation [49] is used to get contact signals.

The skating cost is defined as:

$$L^t_{\text{skating}} = \sum_{\text{foot} \in \{\text{L,R}\}} \left\| p^t_{\text{foot}} - p^{t-1}_{\text{foot}} \right\| + \left\| p^t_{\text{ankle}} - p^{t-1}_{\text{ankle}} \right\|, \tag{9}$$

where $p^t_{\text{foot}} = \mathbf{T}_{\text{root,world}}(\phi^t_R, \gamma^t_R)\mathbf{T}_{\text{foot,root}}(q^t)$ is the position of the foot link in the world at time t.

We also include common robot costs (e.g., joint limits, temporal smoothness of root pose and robot joints) and a small regularization cost on the G1 robot's knee yaw joint for stable leg poses. Processing a 300-frame clip takes around 10 seconds on a single A100, using the PyRoki library [68].

A.4 Evaluation Details

For quantitative evaluation of our real-to-sim pipeline, we conduct experiments on the SLOPER4D dataset [59]. Following established protocols [39, 60, 47], we assess human trajectory reconstruction using World-frame Mean Per Joint Position Error (W-MPJPE) and World-frame Aligned MPJPE (WA-MPJPE), and evaluate scene geometry reconstruction using the Chamfer Distance. Specifically, for human trajectory evaluation, we first slice each sequence into 100-frame segments. W-MPJPE is then computed by aligning only the first two frames to the ground truth, emphasizing global consistency, while WA-MPJPE aligns the entire segment to evaluate local trajectory accuracy. For geometry evaluation, we compute the Chamfer Distance (in meters) between the predicted pointcloud and LiDAR points within the RGB camera's field of view. For benchmarking, we use a subset of SLOPER-4D that contains only those sequences in which SAM2 tracking—human detection plus cross-frame association—succeeds. This yields two sequences each of running, walking, and stair ascent/descent. Results for WHAM and TRAM are reproduced with their official code.

A.5 Ego-view Rendering

We leverage our metrically-scaled 4D reconstruction to render ego-view RGB-D frames by projecting every 3D point into a virtual camera at the Unitree G1 head sensor. This first-person perspective is not consumed by the policy in the present work, yet it demonstrates the pipeline's future potential. Our reconstructions can supply realistic visual observations to future perception-conditioned policies, enabling active vision and semantic scene understanding directly from the robot's viewpoint. A key limitation, however, is that monocular capture leaves many surfaces unobserved; when

Figure 8: **Ego-view RGBD-rendering.** While our policy does not use RGB conditioning for now, we demonstrate the potential of our reconstruction for ego-view rendering. We rasterize the metrically-scaled point cloud into color and depth images by projecting every 3D point onto the image plane of a virtual camera pitched to match the G1's head-mounted sensor. This first-person rendering gives the robot realistic observations of its surroundings, unlocking future work on active vision and semantic understanding, and can be injected directly into the simulator during RL training to couple perception and control.

rasterized, these occluded regions manifest as holes or gaps in the rendered images. Bridging these gaps is an exciting direction for follow-up work: modern data-driven novel-view-synthesis models can hallucinate plausible geometry and appearance for invisible surfaces, promising photorealistic egocentric renderings that would further narrow the sim-to-real perceptual gap for visuomotor policies.

B Policy Learning

B.1 RL Setup and Training

We use the PPO algorithm [53]. We use a discount factor of $\gamma = 0.99$, and a GAE parameter of $\lambda = 0.95$. We use an adaptive learning rate with a desired KL of 0.02. For adaptive learning rate, we use 1.2 as opposed to the usual 1.5 learning rate change, as we find that this leads to faster training. When doing RL from scratch, we set the learning rate (which is modified from this value by adaptive KL) at the start to $1e-3$ however when finetuning we find it is important to start from a very low learning rate of $2e-5$ to prevent early updates with a high learning rate from destroying the existing checkpoint when the distribution of data is biased early during an episode when all environments start. We use a max gradient norm of 1.0. 5 learning epochs are used per rollout, and our rollout length is 24. Our entropy coefficient is set to 0.0025. We use 2 x NVIDIA 4090 GPUs for training, each with 4096 environments (so an effective total of 8192 environments). We use MLPs with 4 layers of [1024, 512, 256, 128] dimensions for different layers.

B.2 Observations

Table 3 details the actor and critic observations. We note that, unlike works in graphics [1, 3], the actor gets all unprivileged observations that are all available on the real robot (see Sec. 5). Furthermore, the target root reference during train time is fed into the global frame to ensure it is aligned with the terrain. Thus, unlike [5, 8], we are able to train policies using a global root reference,

allowing our policies to correctly imitate references in the world frame over long horizons, which is critical for correctly imitating motions tied to a terrain.

Fast-lio2 [63] and probabilistic terrain mapping [64, 17] are used to derive heightmaps and a live odometry signal in real. This can be used to follow 2D commands in the global frame as we did in testing (see Figure 9) and as is done in a concurrent work [69] for the purpose of teleoperation. However, we note that we do not do this in the final version of the paper to avoid any hysteresis in the position commands, and instead multiply the joystick value by a fixed amount to obtain a position which is interpreted by the policy as an offset to the target position in the local frame.

Input	Dim.	Actor (MPT)	Actor (tracking)	Actor (distill/ft.)	Critic
Base angular velocity (last 5)	$5 \times 3 = 15$D	✓	✓	✓	✓
Base linear velocity (last 5)	$5 \times 3 = 15$D	×	×	×	✓
Projected gravity (last 5)	$5 \times 3 = 15$D	✓	✓	✓	✓
DoF positions (last 5)	$5 \times n_{\text{dof}}$	✓	✓	✓	✓
DoF velocities (last 5)	$5 \times n_{\text{dof}}$	✓	✓	✓	✓
Actions (last 5)	$5 \times n_{\text{act}}$	✓	✓	✓	✓
Local frame pos. to target (last 5)	$5 \times 2 = 10$D	✓	✓	✓	✓
Local frame yaw to target (last 5)	$5 \times 1 = 5$D	✓	✓	✓	✓
Target joint angles	n_{dof}	✓	✓	×	✓
Target root roll	1D	✓	✓	×	✓
Target root pitch	1D	✓	✓	×	✓
Terrain height (height-map)	121D	×	✓	✓	✓
Root height	1D	×	×	×	✓
Link heights	n_{links}	×	×	×	✓
Root orientation (quat)	4D	×	×	×	✓
Root position	3D	×	×	×	✓
Joint positions	n_{dof}	×	×	×	✓
Link positions	$3\,n_{\text{links}}$	×	×	×	✓
Link velocities	$3\,n_{\text{links}}$	×	×	×	✓
Feet contact flags	n_{feet}	×	×	×	✓
Feet target contact flags	n_{feet}	×	×	×	✓

Table 3: **Observations for each network**: Motion Capture Pretraining (MPT) actor, motion-tracking actor, distillation/finetuning actor, and critic. Here $n_{\text{dof}} = 23$ for the Unitree G1.

B.3 Actions

We use unbounded actions without activations on the actor output following [54], however following [70] we clip the actions to our chosen range prevent the policy from learning bang-bang style control earlier in training and add a "bounds loss" to the actor with the following formulation to enforce this limit in the actor output:

$$\tilde{\mu}_j = \text{clip}\big(\mu_j,\, -\epsilon,\, \epsilon\big), \tag{10}$$

$$\mathcal{L}_{\text{bounds}} = \frac{1}{D}\sum_{j=1}^{D} \big|\tilde{\mu}_j - \mu_j\big|, \tag{11}$$

where the μ_j are the policy actions per-dimension. We use a coefficient of 0.0005 for this loss. For the G1 humanoid, we limit the action magnitude to 8.0. During experiments, our initial action standard deviation is set to 0.8.

B.4 Simulation Parameters

We run Isaac Gym [12] at 200Hz with control decimation 4, giving an effective policy Δt of $0.02s$.

We set the maximum depenetration velocity especially low – to 0.1m/sec – to prevent large velocities being applied in the occasional case of interpenetration with terrain during the reference motion.

B.4.1 Dealing with Terrains in the Simulator

Due to an undocumented bug in IsaacGym, the simulator registers collisions between robots in different virtual and theoretically isolated "environments" but which occupy the same world-frame spatial volume. This leads to a dramatic increase in memory usage and a significant drop in simulator throughput. To mitigate this, when training on a single clip or a few clips, we duplicate the task terrain and spatially distribute robot spawns to reduce inter-robot collisions.

B.5 Reward Formulation

Our concern above all is to have a system that can take in data and produce new physical motions; hence, our reward is made up mostly of data-driven terms — tracking link and joint positions and velocities, as well as matching feet contact, with minimal re-weighting. In reward design, We have two objectives: firstly, to minimize the strength of manually-designed priors injected into RL tracking via reward engineering.[1] Secondly, to produce physically feasible motions. Note that the two objectives are in tension as since the kinematic data comes from humans, perfect tracking would result in nonphysical motions. We also have several penalty criteria, namely action rate penalties, and two penalties to discourage exploiting simulator physics. Table 4 details the reward terms used for policy learning.

Reward	Formula	Weight	k	Justification
Penalties / Regularization				
Action-Rate Penalty	$\lVert\mathbf{a}_t - \mathbf{a}_{t-1}\rVert^2$	-8.0	–	Can't move too quickly on real robot
Ankle Action Penalty	$\lVert\mathbf{a}_t\rVert^2$	-4.0	–	Ankles should stay near neutral
DOF-Pos Limits Penalty	$\sum$[overflow w.r.t. 0.98 range]	-50.0	–	Penalize joints outside 98 % of their range
Collision Penalty	$\sum 1\big[\lVert\mathbf{f}_c\rVert > 0.1\big]$	-1.0	–	Discourage collisions on penalized links
Contact–No–Velocity Pen.	$\lVert\mathbf{v}_{\text{feet}}\rVert_{(c>1)}$	-100	–	Penalize stationary foot contact
Tracking Rewards				
Joint-Pos Tracking	$\exp\big(-k\,\lVert\mathbf{q} - \mathbf{q}^*\rVert^2\big)$	120	2.0	Follow reference joint angles
Joint-Vel Tracking	$\exp\big(-k\,\lVert\dot{\mathbf{q}} - \dot{\mathbf{q}}^*\rVert^2\big)$	24	0.01	Follow reference joint velocities
Root Ori Tracking	$\exp\big(-k\,\theta_{\text{root}}\big)$	15	3.0	Keep base orientation on track
Torso Pos Tracking	$\exp\big(-k\,\lVert\mathbf{p}_{\text{torso}} - \mathbf{p}^*\rVert^2\big)$	15	50.0	Precise torso translation
Torso Ori Tracking	$\exp\big(-k\,\theta_{\text{torso}}\big)$	15	3.0	Precise torso attitude
Link Pos Tracking	$\exp\big(-k\,\lVert\mathbf{p}_{\text{links}} - \mathbf{p}^*\rVert^2\big)$	30	5.0	Track 13 key link positions
Link Vel Tracking	$\exp\big(-k\,\lVert\dot{\mathbf{p}}_{\text{links}} - \dot{\mathbf{p}}^*\rVert^2\big)$	5	0.1	Track link velocities
Feet Contact Match	$\sum \mathbb{1}\big[c_t^{(i)} = c_t^{*(i)}\big]$	1	–	Match contacts from reference motion (BSTRO)
Feet Air-Time Bonus	$\sum_i (t_{\text{air}}^{(i)} - k)\ \mathbb{1}_{\text{first\_contact}}^{(i)}$	2000	0.25	Reward long, ballistic steps
Reset	Condition	Weight	k	
Termination Penalty	episode end (failure)	-500	–	Harsh penalty on failure events
Alive Reward	added every step	300	–	Incentive to stay alive

Table 4: **Reward terms**, corresponding weights, and scaling factors k. These rewards remain the same across training phases. However, we anneal the action rate and ankle action penalty from 0.2 and 0.0, respectively, to the given values in the table.

B.6 Termination Criteria

Our termination criteria depend on the maximum error of the tracked link joints in the robot. The episode terminates if the Cartesian error exceeds the termination threshold during any step in training. During MPT, we set this to 0.3. When doing the tracking stage over terrains, we set this to 0.5 (higher is better on sometimes noisy references from vision). During RL finetuning, we set this to 1.2. The reason for the high threshold in finetuning is that during RL, we care a lot about ensuring both recovery behaviours and diversity which are not seen during normal DeepMimic-style training on a dataset of the size we are using (and indeed, with too strict a tolerance recovery behaviours will not be seen at all). Hence, a loose tolerance on termination while still using the same tracking

[1] The more such human priors that are added, the less general our motion tracking system will be. This is the inverse of classical sim-to-real RL pipelines, which largely rely on a practitioner's "reward hacking" to get desired behaviours.

reward helps to maintain the essence of the motions in the data while still achieving strong recovery behaviours.

B.7 Domain Randomization

We add various domain randomizations in the simulation to simulate the impacts of various unmodeled physical effects in the simulator, which leads to more robust policies. These are detailed in Table 5.

Category	Parameter	Value / Range	Comment
Dynamics randomization			
DoF friction	μ_{DoF}	$\mathcal{U}[0,\ 0.02]$	Different per environment.
Random pushes (xy)	$\Delta v_{\text{push},xy}$	$\mathcal{U}[-0.25,\ 0.25]$ m/s	Additive to the robot's reference state or current velocity.
Random pushes (z)	$\Delta v_{\text{push},z}$	$\mathcal{U}[-0.10,\ 0.10]$ m/s	Additive to the robot's reference state or current velocity.
Random push interval	T_{push}	10 s	Interval between random pushes.
Observation *offset* noise (additive bias, re-sampled from gaussian per episode with given standard deviation)			
Gravity bias	Δg	0.01g	Fixed bias added to gravity vector for an episode in policy input.
DoF-position bias	Δq	0.005rad	Fixed bias added to each joint position in policy input.
Observation *white* noise (re-sampled every step from Gaussian with divergent standard deviation)			
Relative odom (x, y) to target	σ_{xy}	0.01 m	Resampled every step.
Relative odom yaw to target	σ_{ψ}	0.01 rad	
DoF positions	σ_q	0.01 rad	
DoF velocities	$\sigma_{\dot{q}}$	1.5 rad/s	
Linear base velocity	σ_v	0.1 m/s	
Angular base velocity	σ_{ω}	0.2 rad/s	
Gravity (per-axis)	σ_g	0.05 g	
Odom update-rate randomization			
Update frequency (steps)	n_{odom}	$\mathcal{U}\{2, 6\}$	Odometry observation is held constant for n env-steps to mimic drop-outs.
Height-map sensing noise			
White noise (cells)	σ_h	0.02 m	Gaussian sampled per-cell and per-step.
Offset noise (cells)	Δ_h	0.02 m	Height-bias (applied to all cells, Gaussian sampled per episode).
Roll / Pitch noise	$\sigma_{\phi}, \sigma_{\theta}$	0.04 rad	Projection frame tilt, Gaussian per-env.
Yaw noise	σ_{ψ}	0.08 rad	Map frame rotation, Gaussian per-env.
Sensor delay	$d_{\max}$	$\mathcal{U}\{0, 3\}$	Frames of latency before height-map update.
Update frequency	n_{map}	$\mathcal{U}\{1, 5\}$	Heightmap refresh period (steps).
Bad-distance probability	p_{bad}	0.01	Chance a cell is replaced by a random value.

Table 5: **Domain-randomization and noise settings used during training.** "Offset" terms are fixed per episode; "white" terms are resampled each simulation step. Update-frequency variables simulate low-rate sensors by holding observations constant for a random number of steps.

B.8 Training Data Distribution

After the MPT stage, when training our pipelines on our own data, we train on 123 clips of human motion data collected from our pipeline. We also include during the training process the 10 clips of flat terrain walking data from LaFan [57] that we used during MPT. Since we sample to be class balanced per-clip, this means we train 90% on our own data and 10% on LaFan data. The motion distribution of our 123 collected clips is reported in Table 6.

Category	Keyword examples	# Sequences
Climbing up stairs	"climb up", "walk up stairs (backwards)", "step-up platform / block"	39
Climbing down stairs	"climb / walk down stairs", "down stairs backwards / sideways"	36
Both up & down stairs	"up then down", "mixed up / down", "two people opposite directions"	13
Standing (only)	"standing up"	7
Sitting (only)	"sit down", "sitting on sofa / chair / bench"	12
Both sitting & standing	"sit then stand"	5
Terrain traversing	"stepping stones / chairs", "side-walk", "step on board / block"	11
Total		**123**

Table 6: **Categorization and distribution of our 123 self-collected training clips.**

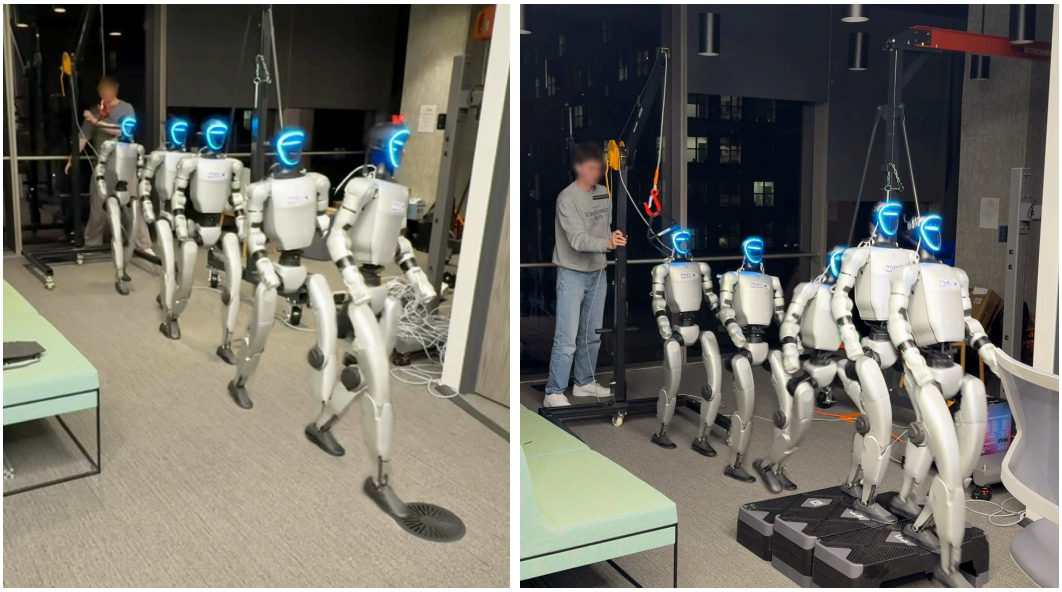


Figure 9: **Evaluation of policies at multiple stages.** Due to the multi-stage nature of our pipeline, we evaluate the performance on real at multiple stages before going out into the real world. *Left*: trying our MoCap Pre-Trained policy. *Right:* Trying the first generalist in a lab environment tracking a pre-defined root trajectory.

C Robot Deployment

We implement deployment code in C++ using ROS and the Unitree SDK 2 to enable fast running on the onboard Jetson Orin NX at 50Hz. We use gains of $Kp = 75$ and $Kd = 2$ for all joints except the ankle, where we use $Kp = 20$ and $Kd = 0.1$ and $Kd = 0.2$ for roll and pitch, respectively, following [62].

C.1 Stages of Deployment in Real

We employed a progressive evaluation approach to deploying our policies, to gradually debug capabilities in the real world. The first stage was being able to track motion capture data from MPT policies. We then distilled motion capture policies trained on a large dataset to test various approaches to distillation; this is how we found that root position conditioning works better than root velocity. Finally, we started to deploy heightmap conditioned distilled policies from our full pipeline. Figure 9 shows two examples of this.