

Visual Imitation Enables Contextual Humanoid Control

Anonymous Author(s)

Affiliation

Address

email

1 This document provides supplementary material for our paper “Visual Imitation Enables Con-
2 textual Humanoid Control.” Section A details the Real-to-Sim pipeline, Section B explains the
3 policy-learning procedure, and Section C describes the real-world robot deployment.

4 A Real-to-Sim Details

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

14 A.1 Notations

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

19 **Human.** We represent reconstructed humans from a video using the SMPL model [1]. SMPL is a
20 differentiable function mapping pose parameters $\theta \in \text{SO}(3)^J$ and shape parameters $\beta \in \mathbb{R}^B$ to a
21 mesh with J joints. The mesh is positioned in the world coordinate system by global orientation
22 $\phi \in \text{SO}(3)$ and translation $\gamma \in \mathbb{R}^3$. Thus, at frame t , a human is defined by:

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

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

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

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

27 **Scene.** The scene is represented using dense per-frame depth and camera output by MegaSaM
28 [2] or MonST3R [3]. Specifically, we use MegaSaM for scenes with richer textures where
29 the correspondence-based Bundle-Adjustment is more reliable. For textureless scenes, we adopt



Figure 1: **Before and after optimization.** We visualize human trajectories and scene point clouds before and after optimization, showing each from two different viewpoints.

MonST3R and its depth-conditioned variant [4] 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 $S_{i,j}$ for pixel (i,j) is obtained by unprojecting the points:

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

A.2 Objectives for Joint Human–Scene Reconstruction

As shown in Fig. 1a, 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 Fig. 1b, and aids 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:

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

$$L_{3D} = \sum_t \|J_{3D}^t - \tilde{J}_{3D}^t\|_1, \quad (4)$$

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

$$L_{\text{Smooth}} = \lambda_\gamma \sum_t \|\gamma^t - \gamma^{t-1}\|_2^2 + \lambda_\theta \sum_t \|\theta^t - \theta^{t-1}\|_2^2, \quad (6)$$

where $\Pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ 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 [5]. 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 [6] provides the roll–pitch angles for that frame; the composite rotation $R_{\text{gravity,world}} = R_{y \uparrow \rightarrow z} R_x(\pi) R_z(\rho) R_{\text{cam,world}}$ re-ori-ents 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 ± 2 m box around the SMPL joints per frame. A 0.1 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 [7] 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 (~ 5 s for gravity alignment and ~ 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 [8, 9], 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 $(\phi_R^{1:T}, \gamma_R^{1:T})$, 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. [10], we implement a kinematic tree-based motion transfer cost $L_{\text{motion}}^t = L_{\text{position}}^t + L_{\text{angle}}^t$:

$$L_{\text{position}}^t = \sum_{i \neq j} m_{ij} \left\| \Delta_{ij}^{\text{SMPL}}(t) - s \cdot \Delta_{ij}^{\text{G1}}(t) \right\|_2^2, \quad (7)$$

$$L_{\text{angle}}^t = \sum_{i \neq j} m_{ij} \left(1 - \langle \Delta_{ij}^{\text{SMPL}}(t), \Delta_{ij}^{\text{G1}}(t) \rangle \right), \quad (8)$$

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 [11] is used to get contact signals.

The skating cost is defined as:

$$L_{\text{skating}}^t = \sum_{\text{foot} \in \{\text{L}, \text{R}\}} \left\| p_{\text{foot}}^t - p_{\text{foot}}^{t-1} \right\| + \left\| p_{\text{ankle}}^t - p_{\text{ankle}}^{t-1} \right\|, \quad (9)$$

where $p_{\text{foot}}^t = \mathbf{T}_{\text{root,world}}(\phi_R^t, \gamma_R^t) \mathbf{T}_{\text{foot,root}}(q^t)$ 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 [12].

A.4 Evaluation Details

For quantitative evaluation of our real-to-sim pipeline, we conduct experiments on the SLOPER4D dataset [13]. Following established protocols [14, 15, 16], we assess human trajectory reconstruction

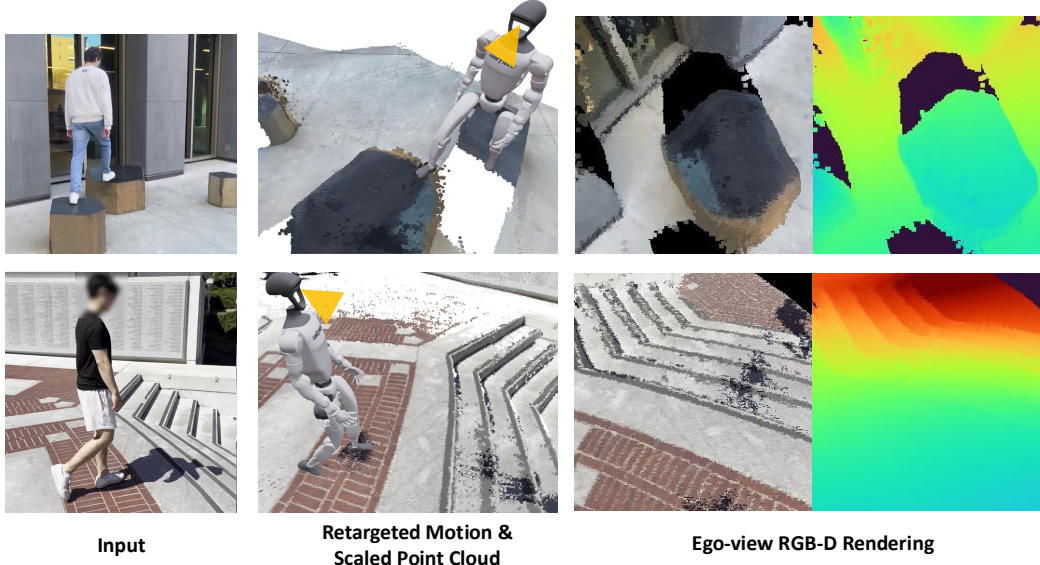


Figure 2: **Ego-view RGBD-rendering.** While our policy does not use RGB conditioning for now, we demonstrate the potential of our reconstruction for ego-view rendering. We rasterize the metrically-scaled point cloud into color and depth images by projecting every 3D point onto the image plane of a virtual 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.

87 using World-frame Mean Per Joint Position Error (W-MPJPE) and World-frame Aligned MPJPE
88 (WA-MPJPE), and evaluate scene geometry reconstruction using the Chamfer Distance. Specifi-
89 cally, for human trajectory evaluation, we first slice each sequence into 100-frame segments. W-
90 MPJPE is then computed by aligning only the first two frames to the ground truth, emphasizing
91 global consistency, while WA-MPJPE aligns the entire segment to evaluate local trajectory accu-
92 racy. For geometry evaluation, we compute the Chamfer Distance (in meters) between the predicted
93 pointcloud and LiDAR points within the RGB camera’s field of view. For benchmarking, we use a
94 subset of SLOPER-4D that contains only those sequences in which SAM2 tracking—human detec-
95 tion plus cross-frame association—succeeds. This yields two sequences each of running, walking,
96 and stair ascent/descent. Results for WHAM and TRAM are reproduced with their official code.

97 A.5 Ego-view Rendering

98 We leverage our metrically-scaled 4D reconstruction to render ego-view RGB-D frames by project-
99 ing every 3D point into a virtual camera at the Unitree G1 head sensor. This first-person perspective
100 is not consumed by the policy in the present work, yet it demonstrates the pipeline’s future poten-
101 tial. Our reconstructions can supply realistic visual observations to future perception-conditioned
102 policies, enabling active vision and semantic scene understanding directly from the robot’s view-
103 point. A key limitation, however, is that monocular capture leaves many surfaces unobserved; when
104 rasterized, these occluded regions manifest as holes or gaps in the rendered images. Bridging these
105 gaps is an exciting direction for follow-up work: modern data-driven novel-view-synthesis models
106 can hallucinate plausible geometry and appearance for invisible surfaces, promising photorealistic
107 egocentric renderings that would further narrow the sim-to-real perceptual gap for visuomotor
108 policies.

B Policy Learning

B.1 RL Setup and Training

We use the PPO algorithm [17]. 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 1 details the actor and critic observations. We note that, unlike works in graphics [18, 19], the actor gets all unprivileged observations that are all available on the real robot. Furthermore, the target root reference during train time is fed into the global frame to ensure it is aligned with the terrain. Thus, unlike [20, 21], 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.

Input	Dim.	Actor (MPT)	Actor (tracking)	Actor (distill/ft.)	Critic
Base angular velocity (last 5)	$5 \times 3 = 15\text{D}$	✓	✓	✓	✓
Base linear velocity (last 5)	$5 \times 3 = 15\text{D}$	✓	✓	✓	✓
Projected gravity (last 5)	$5 \times 3 = 15\text{D}$	✓	✓	✓	✓
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)	$5 \times 2 = 10\text{D}$	✓	✓	✓	✓
Local frame yaw to target (last 5)	$5 \times 1 = 5\text{D}$	✓	✓	✓	✓
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 1: **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 [22], however following [23] 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

133 limit in the actor output:

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

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

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

137 B.4 Simulation Parameters

138 We run Isaac Gym [24] at 200Hz with control decimation 4, giving an effective policy Δt of 0.02s.
 139 We set the maximum depenetration velocity especially low – to 0.1m/sec – to prevent large velocities
 140 being applied in the occasional case of interpenetration with terrain during the reference motion.

141 B.4.1 Dealing with Terrains in the Simulator

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

147 B.5 Reward Formulation

148 Our concern above all is to have a system that can take in data and produce new physical motions;
 149 hence, our reward is made up mostly of data-driven terms — tracking link and joint positions and
 150 velocities, as well as matching feet contact, with minimal re-weighting. In reward design, We have
 151 two objectives: firstly, to minimize the strength of manually-designed priors injected into RL track-
 152 ing via reward engineering.¹ Secondly, to produce physically feasible motions. Note that the two
 153 objectives are in tension as since the kinematic data comes from humans, perfect tracking would
 154 result in nonphysical motions. We also have several penalty criteria, namely action rate penalties,
 155 and two penalties to discourage exploiting simulator physics. Table 2 details the reward terms used
 156 for policy learning.

157 B.6 Termination Criteria

158 Our termination criteria depend on the maximum error of the tracked link joints in the robot. The
 159 episode terminates if the Cartesian error exceeds the termination threshold during any step in train-
 160 ing. During MPT, we set this to 0.3. When doing the tracking stage over terrains, we set this to 0.5
 161 (higher is better on sometimes noisy references from vision). During RL finetuning, we set this to
 162 1.2. The reason for the high threshold in finetuning is that during RL, we care a lot about ensuring
 163 both recovery behaviours and diversity which are not seen during normal DeepMimic-style training
 164 on a dataset of the size we are using (and indeed, with too strict a tolerance recovery behaviours
 165 will not be seen at all). Hence, a loose tolerance on termination while still using the same tracking
 166 reward helps to maintain the essence of the motions in the data while still achieving strong recovery
 167 behaviours.

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

Table 2: **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.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 3.

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 [25] 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 4.

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 [26].

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 3 shows two examples of this.

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 <i>offset</i> 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 <i>white</i> 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 3: **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.

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

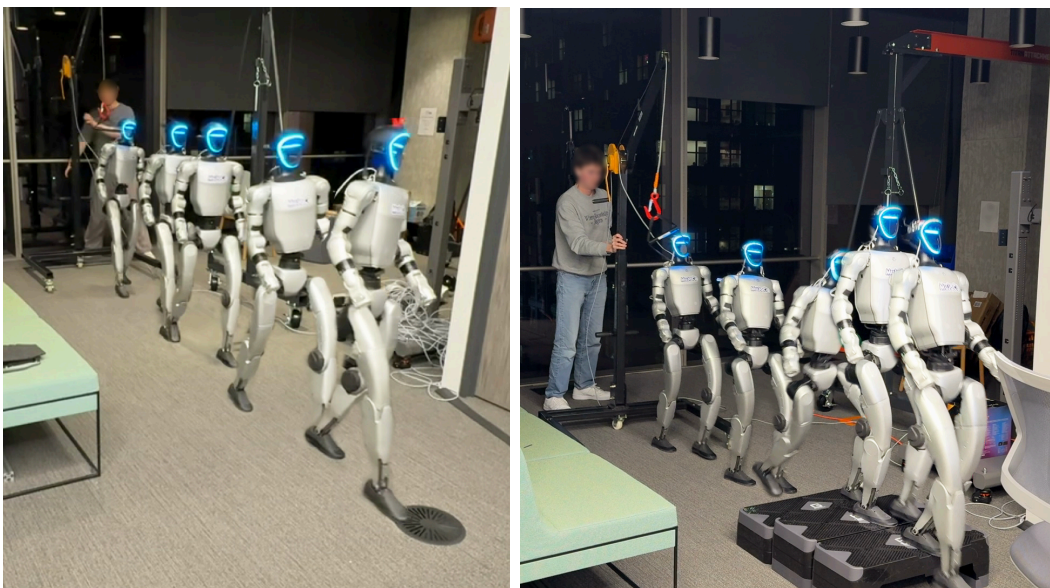


Figure 3: **Evaluation of policies at multiple stages.** Due to the multi-stage nature of our pipeline, we evaluate the performance on real at multiple stages before going out into the real world. *Left:* trying our MoCap Pre-Trained policy. *Right:* Trying the first generalist in a lab environment tracking a pre-defined root trajectory.

References

- [1] 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.
- [2] 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.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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>.
- [9] 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>.
- [10] 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.
- [11] 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.
- [12] 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>.
- [13] 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.
- [14] 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.
- [15] 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.

- [16] 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.
- [17] 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>.
- [18] 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/10.1145/3197517.3201311>.
- [19] 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>.
- [20] 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>.
- [21] 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>.
- [22] 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>.
- [23] D. Makoviichuk and V. Makoviychuk. rl-games: A high-performance framework for reinforcement learning. https://github.com/Denys88/rl_games, May 2021.
- [24] 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>.
- [25] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal. Robust motion in-betweening. 39 (4), 2020.
- [26] 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>.