

What the Reward Curve Didn't Show: Silent Failures in DeepMimic-Style Imitation on the Unitree H1

Bhargav Oza
Independent Researcher, Rajkot, India
ozabhargav41@gmail.com

Abstract

We reimplement DeepMimic-style motion imitation from scratch for the full 19-DOF Unitree H1 humanoid in MuJoCo MJX and JAX, and report the failures we met on the way. Many of them were *silent*: training ran, the reward curve rose, and the robot still did the wrong thing. We describe these failures, grouped into action-space, coordinate-frame, data-pipeline, reward-code, training-loop and simulation-setup errors, together with the evidence that exposed each one. Our main observation is that symptoms which look like reward-design problems in motion imitation are often action-space or data-pipeline bugs, and the aggregate reward cannot tell the two apart. Examples include an action scale that made the reference ankle pose unreachable (fixing it raised walking speed from 0.74 to 1.26 m/s), end-effector and centre-of-mass terms that, computed in the world frame as in the original DeepMimic, lost their signal as soon as the pelvis drifted, a centre-of-mass reward that was too flat to give a learning signal, and a sign typo that turned a penalty into a bonus and taught the robot to hop. After the fixes, the plain DeepMimic reward, with no extra path, velocity-command or symmetry terms, produces a policy that walks the full 14.7-minute reference clip in simulation without a single fall, with at most 24 cm of sideways offset from the reference path. We also report a small ablation of termination penalties and a per-joint diagnostic protocol that found the bugs where reward curves did not. This is a case study, not a new method.

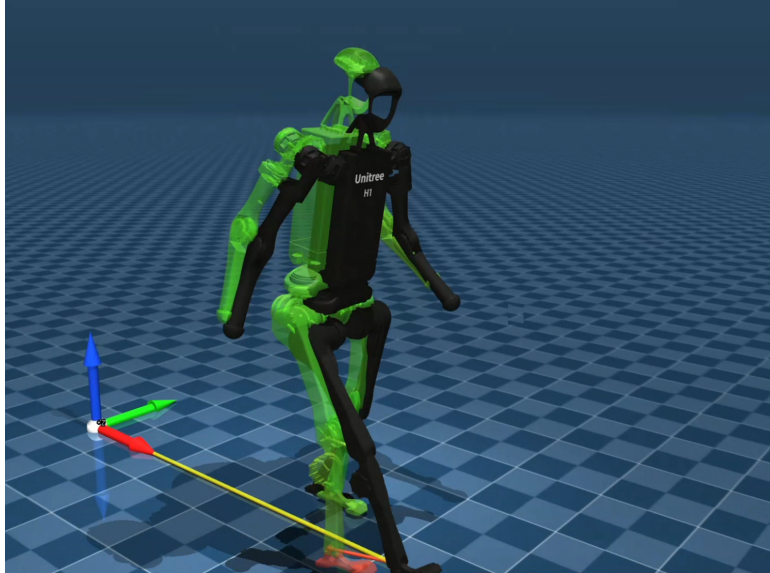


Figure 1: The trained policy (black) walking alongside the reference motion (transparent green) in the MuJoCo viewer.

1 Introduction

Motion imitation with a tracking reward, popularised by DeepMimic [1], is a standard way to get natural locomotion from a physics-based character or robot. The idea is simple: give the policy a reference motion and reward it, at every step, for matching the reference pose. Mature open-source implementations exist, including a full DeepMimic setup for the Unitree H1 in MJX in LocoMuJoCo [2].

This paper is about what happens when you build such a pipeline yourself. We reimplemented DeepMimic for the full 19-DOF Unitree H1 in MuJoCo MJX [3] and JAX [4], training 4,096 parallel environments at about 8,800 environment steps per second on a single consumer GPU (RTX 4070 Ti). The final policy walks well. The more useful result, we think, is the list of failures we met before getting there, and the fact that most of them did not show up in the reward curve.

We call a failure **silent** when training keeps running and the reward keeps looking healthy, but the robot’s behaviour is wrong. A clear example from this project: a double-negative typo (`--0.30`) turned the airborne penalty into a bonus. The policy learned to hop, spending 17% of the time with both feet off the ground, while the total reward stayed positive.

Prior work has shown that implementation details strongly affect deep RL results [5–7]. Our case study adds a robotics-specific version of that lesson for motion imitation: the reward is a weighted sum of many terms, each term has its own coordinate frame and scale, and the robot’s action space and controller sit between the policy and the reference. Each of these is a place where a bug can hide behind a rising reward.

Contributions.

- A catalogue of silent failures from a from-scratch DeepMimic reimplementation on the Unitree H1, grouped by where they occur, with the symptom, root cause and effect of the fix (Section 4).
- The observation that reward-design-looking symptoms (drift, feet not clearing the ground, arms not moving) were often caused by the action space or the data pipeline, which the aggregate reward could not distinguish.
- A per-joint diagnostic protocol that found these bugs where reward curves did not (Section 5).
- Evidence that, once these bugs are fixed, the plain DeepMimic reward is enough for straight, stable walking, and a small ablation of termination penalties (Section 6).

2 Related Work

Motion imitation. DeepMimic [1] combines a pose, velocity, end-effector and centre-of-mass tracking reward with reference state initialisation (RSI) and early termination. Later work extends tracking to large motion libraries, e.g. PHC [8]. LocoMuJoCo [2] provides an imitation-learning benchmark for locomotion with retargeted mocap for several robots, including H1, and an MJX DeepMimic baseline. We use LocoMuJoCo’s H1 walking clip as our reference. More recent systems build full motion-tracking pipelines and frameworks. BeyondMimic [9] tracks highly dynamic human motions on real humanoid hardware and distils the trackers into a guided-diffusion controller, and MimicKit [10] collects common motion-imitation methods in one open-source framework, motivated by the observation that effective motion-imitation systems depend on many small design details. Our goal is not to improve on these systems but to document the failure modes of building one.

Closest to our work is the empirical study of Amadio and Mingo Hoffman [11], which compares common design choices for general motion tracking, such as the motion-command representation, observation history, action representation and actuation profile, in a controlled framework on the Unitree G1. Their study asks which *correct* design choices matter; ours asks which *implementation errors* go unnoticed. The two are complementary: a design-choice comparison assumes a correct pipeline, and silent bugs of the kind we report could change the outcome of such a comparison without any visible sign.

Massively parallel legged-robot RL. GPU-parallel simulation made it practical to train legged locomotion policies in minutes [12]. Isaac Lab (formerly Orbit) [13] provides reference configurations for the H1, including PD gains and action scaling, which we adopt.

Reproducibility and implementation details. Henderson et al. [5] showed that deep RL results depend heavily on seeds and implementation choices. Engstrom et al. [6] showed that code-level details explain much of PPO’s advantage over TRPO, and Huang et al. [7] list the many implementation details of PPO. Our work is in the same spirit, but focuses on the robot-side and reward-side details of motion imitation rather than the RL algorithm. Bouchoucha et al. [14] note that faults in deep RL programs often cause wrong behaviour without any explicit error, and propose RLExplorer, which monitors training traces to diagnose such faults automatically. Our diagnostic protocol (Section 5) is manual and specific to motion imitation: it compares the robot with the reference, joint by joint.

3 Setup

3.1 Robot, simulator and reference motion

We control all 19 actuated joints of the Unitree H1 (10 leg, 1 torso, 8 arm) using the MuJoCo Menagerie model [15]. The model has no touch or IMU sensors, so we added foot and thigh touch sensors and a torso IMU. Physics runs at 200 Hz ($\Delta t_{\text{sim}} = 0.005$ s) and the policy at 50 Hz (4 physics substeps per policy step).

The reference is the H1 walking clip from LocoMuJoCo: 35,198 frames at 40 Hz (about 880 s, or 14.7 minutes), always walking in $+x$. We remap the joints from OpenSim naming to the H1 MJCF order, resample to 50 Hz by linear interpolation (43,997 frames), and compute joint and root velocities by finite differences. Reference end-effector and centre-of-mass (CoM) positions are precomputed once with forward kinematics (one `mj_forward` per frame) and stored as arrays, so the reward only needs a lookup at training time. H1 has no hand body; the arm ends at the elbow, so we use the elbow links as the hand end-effectors.

3.2 Observation and action

The 90-dimensional observation contains: root linear velocity in the body frame (3), IMU angular velocity (3), projected gravity (3); the reference leg, torso and arm joint positions for the current frame as offsets from the default pose (10+1+8); the reference forward and lateral root velocity 0.1 s ahead, rotated into the robot’s body frame (2); the wrapped yaw error to the reference (1); the offset of the robot’s root from the reference root in the robot’s heading frame (2); current joint positions and velocities for legs, torso and arms (10+10+1+1+8+8); and the previous action (19).

The 19-dimensional action is a joint position target. Following the Isaac Lab H1 configuration [13], the action is clipped to $[-3, 3]$, scaled by 0.5 and added to the default (bent-knee) pose. The target is then clipped to the joint limits and tracked by a PD controller with zero velocity target, run at every physics substep:

$$q^{\text{des}} = \text{clip}(q^{\text{default}} + 0.5 \cdot \text{clip}(a, -3, 3), q^{\text{lo}}, q^{\text{hi}}), \quad (1)$$

$$\tau = \text{clip}(K_p(q^{\text{des}} - q) - K_d \dot{q}, -\tau^{\text{max}}, \tau^{\text{max}}). \quad (2)$$

Gains and limits are listed in Table 1.

Table 1: PD gains (Isaac Lab H1 configuration), actuator effort limits from `h1.xml`, and default pose.

Joint	K_p	K_d	Effort limit (N m)	Default (rad)
Hip yaw	150	5	200	0.0
Hip roll	150	5	200	0.0
Hip pitch	200	5	200	-0.4
Knee	200	5	300	0.8
Ankle	40	2	40	-0.4
Torso	200	5	200	0.0
Shoulder pitch / roll	40	10	40	0.0
Shoulder yaw / elbow	40	10	18	0.0

3.3 Reward

The tracking reward is the four DeepMimic terms, with the CoM weight lowered from 0.10 to 0.05:

$$r^{\text{track}} = 0.65 r_{\text{pose}} + 0.10 r_{\text{vel}} + 0.15 r_{\text{ee}} + 0.05 r_{\text{com}}. \quad (3)$$

Each term has the form $r_k = \exp(-\alpha_k e_k)$, where e_k is a *summed* squared error: e_{pose} over the 19 joint angles (hip pitch and knee weighted $\times 2$), e_{vel} over the 19 joint velocities, e_{ee} over the feet and elbows, and e_{com} over the whole-body CoM. Unlike DeepMimic, which uses world positions, we measure end-effector and CoM positions in the pelvis frame, for both the robot and the reference (Section 4.2). The full per-step reward is

$$r = 2.0 r^{\text{track}} + r_{\text{energy}} + r_{\text{limit}} + r_{\text{smooth}} + r_{\text{contact}} + r_{\text{fall}} + r_{\text{div}}, \quad (4)$$

with the regularisers and penalties in Table 2. There is deliberately no alive bonus. The α values are listed in Table 3.

Table 2: Regularisers and termination penalties. Overshoot is the distance a joint has moved into the last 10% of its range (zero elsewhere). Fall and divergence penalties can both apply on the same step.

Term	Definition
r_{smooth}	$-0.01 \sum_j (a_{t,j} - a_{t-1,j})^2$
r_{energy}	$-10^{-5} \sum_j \tau_j \dot{q}_j $, using the applied (clipped) torques
r_{limit}	$-0.01 \sum_j \text{overshoot}_j^2$
r_{contact}	$-10^{-6} \sum_{\text{feet}} \max(0, F - 500 \text{ N})^2$
r_{fall}	-0.8 (Phase 0), -1.5 (Phase 1), on the step the robot falls
r_{div}	-0.5 (Phase 0), -2.5 (Phase 1), on the step the robot diverges

Table 3: Tracking-term sharpness α per curriculum phase, compared with the values that failed for us.

Term	Phase 0	Phase 1	Value that failed
r_{pose}	2.0	2.5	–
r_{vel}	0.03	0.04	0.1 (reward near 0)
r_{ee}	30	40	–
r_{com}	3000	4000	10 (no learning signal)

3.4 Termination and curriculum

An episode ends when the robot falls (pelvis height below 0.6 m, or tilt $\|g_{xy}\| > 0.7$, about 44°), diverges from the reference, times out (500 steps, 10 s) or reaches the end of the clip. Divergence has three checks: RMS joint tracking error over all 19 joints, sideways (y) and ahead/behind (x) offset of the pelvis from the reference pelvis, and heading error (Table 4). The sideways limit is stricter than the forward one, since small speed differences are acceptable but sideways drift is not.

Training uses two curriculum phases, switched by PPO update count: Phase 0 before update 1000 and Phase 1 after. Phase 1 uses sharper α values, tighter divergence limits and larger penalties. Each environment picks up the new phase when it next resets.

3.5 Policy and training

Actor and critic are separate MLPs with hidden layers [512, 256, 128] and ELU activations; the action standard deviation is a learnable, state-independent parameter initialised at $\log \sigma = -1.4$. We train with PPO [16] and GAE [17]: 4,096 environments, 24-step rollouts, 5 epochs, 32 minibatches, $\gamma = 0.99$, $\lambda = 0.95$, clip $\epsilon = 0.2$, value coefficient 0.5, entropy coefficient 0.005 and gradient-norm clipping at 0.5. The learning rate starts at 3×10^{-4} and adapts to a KL target of 0.02: it is multiplied by 0.7 when the approximate KL exceeds

Table 4: Divergence termination limits per phase.

Condition	Phase 0	Phase 1
Joint-tracking RMS error	$> 0.50 \text{ rad } (\approx 29^\circ)$	$> 0.35 \text{ rad } (\approx 20^\circ)$
Sideways pelvis offset (y)	$> 0.30 \text{ m}$	$> 0.20 \text{ m}$
Ahead/behind pelvis offset (x)	$> 0.50 \text{ m}$	$> 0.40 \text{ m}$
Heading error	$> 0.30 \text{ rad } (\approx 17^\circ)$	$> 0.20 \text{ rad } (\approx 11^\circ)$

twice the target and by 1.3 when it falls below half the target, within $[10^{-6}, 5 \times 10^{-4}]$. Episodes start from a random frame of the clip (RSI), using the reference pose with $\pm 0.05 \text{ rad}$ uniform joint noise and zero joint velocities. Every 180 policy steps (3.6 s) the robot receives a random horizontal velocity push of up to 0.5 m/s per axis.

4 Silent Failures

Table 5 summarises the failures. We describe the most instructive ones below. In almost every case, the reward curve either kept rising or settled at a plausible value, so the failure was only visible by watching the robot or by comparing it to the reference joint by joint.

4.1 An unreachable reference: the action scale

The first action scale was computed per joint as $0.8 \cdot \min(q^{\text{default}} - q^{\text{lo}}, q^{\text{hi}} - q^{\text{default}})$. This looks safe, since it keeps targets inside the joint limits, but it makes the reachable range symmetric around the default pose and as small as the *shorter* side. For the ankle, the reference needs about $+15^\circ$ of dorsiflexion, which this range did not allow. The feet could not physically clear the ground, so the policy shuffled. The same truncation pinned the shoulder roll and elbow at their limits, which looked like “the arms are not learning”.

Both symptoms invite reward changes: a foot-clearance term for the feet, more arm weight for the arms. Neither would have helped, because the reference pose was outside the action space. Replacing the scale with the Isaac Lab setting (scale 0.5, actions clipped to ± 3 , targets clipped to the joint limits) raised walking speed from 0.74 to 1.26 m/s with no reward change.

4.2 Coordinate frames: world vs. pelvis

DeepMimic defines the end-effector reward on the 3D world positions of the hands and feet, and compares the character’s centre of mass directly with that of the reference [1]; only its *observations* are expressed in the character’s root frame. We first did the same. In our setup, the robot’s pelvis drifted away from the reference pelvis within the first second of every episode, and both terms dropped to near zero and stayed there. With the paper’s weights this silently removed 25% of the tracking reward. The total reward still rose, driven by r_{pose} , so nothing looked broken.

We therefore moved both terms to the pelvis frame: the robot’s positions are measured relative to its own pelvis, and the reference positions relative to the reference pelvis. This keeps a useful signal, but it is a trade-off. A pelvis-relative reward no longer says where the robot is in the world, so the reward alone no longer penalises drifting away from the reference path. We put that information back in two other places: the observation, which includes the heading error and the path offset to the reference (Section 3.2), and the path and heading terminations (Section 3.4). The same pattern appeared in a foot-clearance term that used world-frame foot height: when the pelvis rode low, every foot looked low.

4.3 A reward term that looks solved: α_{com}

After moving to the pelvis frame (Section 4.2), we first kept DeepMimic’s $\alpha_{\text{com}} = 10$. r_{com} then sat at about 0.9–0.97 from the very first update, while the robot visibly drifted sideways. The term looked solved, but it was simply flat. Because the error is measured in the pelvis frame, it is small, of the order of centimetres. For a 3 cm error, $\exp(-10 \cdot 0.03^2) \approx 0.991$: almost no difference from a perfect match, so almost no gradient.

Table 5: Silent failures found during the reimplementaion, grouped by where they occur.

Group	Visible symptom	Root cause	Fix / effect
Action space	Feet barely clear the ground; slow gait	Action scale $\min(q^{\text{def}} - q^{\text{lo}}, q^{\text{hi}} - q^{\text{def}}) \times 0.8$ cut off the side of the range the reference used; ankle could not reach the reference $+15^\circ$ dorsiflexion	Isaac Lab scale 0.5 with ± 3 clip; speed $0.74 \rightarrow 1.26$ m/s
Action space	Arms do not move	Same truncation pinned shoulder roll and elbow at their limits	Same fix
Action space	Weak stance-width control	Hip-roll effort limit 150 N m vs. ± 200 in <code>h1.xml</code>	Set to 200
Action space	Arm commands have no effect	Arm slice off by one (17:25), writing arm targets to the torso slot	18:26
Action space	Arm torque saturated 40–50% of the time	Arm $K_p = 60$	$K_p = 40$
Coordinate frames	$r_{\text{ee}}, r_{\text{com}}$ near 0 within 1 s of every episode	World-frame positions, as in DeepMimic; any root drift destroyed the signal (25% of reward weight with the paper’s weights)	Pelvis-frame positions; path information moved to observations and terminations
Coordinate frames	Velocity target inconsistent	Reference linear velocity in world frame, robot velocity in body frame	Rotate reference into robot body frame
Coordinate frames	Foot-clearance term penalised the whole leg	World-frame foot height; a low pelvis made every foot look low	Height relative to root
Reward scale	Steady sideways drift while $r_{\text{com}} \approx 0.9$ –0.97	$\alpha_{\text{com}} = 10$ too flat to give a gradient	$\alpha_{\text{com}} = 3000 \rightarrow 4000$
Reward scale	$r_{\text{vel}} \approx 0$ throughout	$\alpha_{\text{vel}} = 0.1$ with summed error	$\alpha_{\text{vel}} = 0.03 \rightarrow 0.04$
Reward code	Robot hops (17% flight time), reward positive	Typo --0.30 turned the airborne penalty into a bonus	Fixed sign
Reward code	Both legs lean the same way	Symmetry term $(\text{roll}_L - \text{roll}_R)^2$ with both hip-roll axes "1 0 0" rewards same-direction lean	Penalise asymmetry of tracking error
Reward code	Energy term inconsistent with applied torque	Torques recomputed without the default-pose offset	Use applied torques
Reward code	“Action-rate” term pulls toward default pose	Term actually penalised $\sum a^2$, not Δa	Use r_{smooth}
Reward code	One fall dominates a batch; value fit unstable	Termination penalty -50 vs. per-step reward ≈ 1	Small phase-dependent penalties
Data pipeline	Heading kick right after reset	Finite-difference velocities invalid at frames 0–1	Never start before frame 10
Data pipeline	Robot falls on step 1; looked like “needs more training” for 2,700+ updates	RSI noised joint positions but kept reference joint velocities; PD spike on first step	Zero joint velocities at reset
Training loop	Inflated fall rate, weak signal	Fallen robots simulated until end of the 24-step rollout	Reset inside the rollout
Training loop	Reward diverges with no error	Auto-reset did not reset <code>qpos/qvel</code> ; robots restarted fallen	Masked full-state reset
Training loop	Exploration never changes	$\log \sigma$ stored as a Python float, not a parameter	Learnable parameter
Training loop	No pushes; later identical pushes in all envs	Push interval 750 > episode length 500; RNG keyed only on step count	Interval 180; key on step count and frame index
Simulation	Policy behaves differently in the viewer	Training model had MJX collision fixes, viewer model did not	Same model in both

With $\alpha_{\text{com}} = 3000$ the same error gives $\exp(-2.7) \approx 0.07$, and a 1 cm error gives $\exp(-0.3) \approx 0.74$, so the term now separates good and bad CoM tracking. In training, r_{com} then rises from below 0.7 to about 0.9 within about 150 updates (Figure 3), later dips to about 0.7 near update 600 and settles at 0.75–0.8. Together with the other fixes in this section, the sideways drift disappeared. The same reasoning explains r_{vel} : with a summed error over 19 joint velocities, $\alpha_{\text{vel}} = 0.1$ kept the term near zero. We kept summed errors, as in our implementation, and tuned α instead of switching to per-joint means.

The general point is that an $\exp(-\alpha e)$ term is only informative when αe is of order one for the errors the policy actually makes. A term that is near 1 from update 0 is as broken as a term that is near 0.

4.4 A penalty that became a bonus

A double-negative typo, `--0.30`, turned the airborne penalty into a reward. The policy learned to hop, with both feet off the ground 17% of the time. Because the tracking terms were still partly satisfied, the total reward stayed positive and kept rising. The failure was only visible in the video and in the logged airborne rate. In the final version, the airborne rate is logged as a diagnostic only and is not part of the reward.

4.5 Reset-time inconsistencies

Two bugs were at reset. First, reference velocities are computed by finite differences, and the first two frames of the resampled clip are duplicates, so starting at frame 0 gave a wrong velocity and a heading kick. We never start before frame 10. Second, RSI added noise to the joint positions but kept the reference joint velocities. The resulting state was inconsistent, the PD controller spiked on the first step and the robot fell immediately. This persisted for more than 2,700 updates and looked exactly like a policy that “needs more training”. Setting joint velocities to zero at reset fixed it.

4.6 A floating reference

The retargeted reference is not exactly on the floor. We placed the H1 model in the reference pose at every 50th frame of the clip (from frame 10 onward), ran forward kinematics, and took the lowest point of either foot as the height of the foot capsule’s centre minus its radius (0.014 m). Over the sampled frames, the lowest foot point is on average approximately 3.7 cm above the floor (minimum 0.8 cm, maximum 6.1 cm). This measure is exact only when the capsule is level; when the foot is pitched, at heel strike or toe-off, one end of the capsule sits lower than its centre, so the true gap on those frames is somewhat smaller.

A floating reference cannot be matched exactly by a robot that stands on the floor. It also affects the reset: since RSI places the robot at the reference root position, the robot starts slightly above the floor and drops a few centimetres at the start of each episode. PHC deals with the same issue by adjusting the reference root height so that the feet touch the ground at the start of each episode [8]. We do not apply such a correction; our end-effector and CoM terms are pelvis-relative and so do not depend on the absolute height.

4.7 Training-loop bugs

Fallen robots were kept in simulation until the end of each 24-step rollout instead of being reset immediately. This inflated the fall rate and filled the batch with uninformative transitions. An earlier auto-reset reset the episode counters but not `qpos/qvel`, so robots restarted from their fallen pose, and the reward diverged without any error. Separately, the action log-standard-deviation was stored as a Python float rather than a learnable parameter, so exploration noise never changed, and the push randomisation never fired, because the push interval (750 steps) was longer than the episode (500 steps). Once fixed, all 4,096 environments received the same push, because the random key depended only on the step count.

5 Diagnostic Protocol

None of the bugs above was found by reading the reward curve. They were found by comparing the robot to the reference, joint by joint. We used two tools:

- **Reference “ghost” viewer.** The policy rollout is drawn on top of the reference motion, with per-joint distance overlays and per-joint error tables.
- **Headless drift diagnostic.** A deterministic evaluation from several start frames that reports: excess yaw per support phase; per-joint error split into stance and swing, with left and right mirrored for comparison; stride-level correlation between joint error and turning; foot placement at touchdown compared to the reference; pivoting of the planted foot; and action and torque saturation.

From this experience we suggest a short checklist before tuning any reward:

1. Check that every reference pose is reachable by the action space and the joint limits, joint by joint.
2. Check the coordinate frame of every quantity that enters the reward or observation, for both the robot and the reference.
3. For each $\exp(-\alpha e)$ term, evaluate αe on typical errors; it should be of order one. A term near 0 or near 1 from update 0 gives no signal.
4. Check that the reset state is physically consistent (positions and velocities) and that the first frames of the clip have valid velocities.
5. Evaluate the trained policy in exactly the same simulation model as training.
6. Log every reward term separately, plus saturation and contact statistics, and look at the robot, not only the curve.

6 Results

6.1 Training and long-horizon evaluation

Figures 2 and 3 show a training run of 1,605 PPO updates (about 4.9 hours on one RTX 4070 Ti). The mean reward rises quickly to a peak of about 1.3 near update 350, then settles at about 1.0–1.07 for the rest of training. The fall rate drops to almost zero by about update 500 and is exactly zero at the end of training. The termination rate settles at about 0.3% (0.0028 at the last update), most of it from divergence terminations (about 0.2%). The switch to Phase 1 at update 1,000 is visible as a small jump in the termination and divergence rates and a drop in r_{vel} , as expected from the tighter limits and sharper α values. The airborne rate falls to about 1% by update 600 but rises again to about 5% after update 1,300; we have not yet investigated this. In a viewer check at update 1,000 of an earlier run ($\alpha_{\text{com}} = 500$), the robot walked full 10s episodes at about 1.5 m/s, keeping the pelvis within 3–6 cm of the reference and drifting less than 8 cm sideways; versions before the fixes drifted more than 2 m. In a headless evaluation of the update-600 checkpoint of the same run, the policy walks the *entire* 14.7-minute reference clip without a single fall, with a maximum sideways offset of 24 cm from the reference path, even though it was only trained on 10s episodes.

This uses only the plain DeepMimic tracking terms plus the regularisers and termination penalties above. Removing an earlier velocity-command term, a path term and a symmetry term from the reward did not stop the robot from walking straight. Several other attempts did not help and were reverted: world-frame root-position tracking made the robot slide instead of walk, a heading reward caused oscillation, and straight-line and root-position rewards made the robot drag its feet. Heading drift was instead solved by the heading termination, the heading and path-offset observations, and an r_{com} term that actually gives a signal.

6.2 Ablation: termination penalties

We trained four variants that differ only in which termination penalties are active, using one seed each and evaluating from start frame 10 (Table 6). These four runs were made *before* the rollout fix described in Section 4.7: fallen robots stayed in the batch until the end of the 24-step rollout, so the fall penalty fired on every remaining step of that rollout rather than once. The effect of the penalties may therefore be smaller with the corrected loop. With that caveat, the penalties mainly speed up early learning and prevent a bad early habit; the fall penalty does most of the work. They are not needed for the final path tracking: by update 900–1000, all runs track the path similarly, but the run without penalties keeps a right shoulder-roll

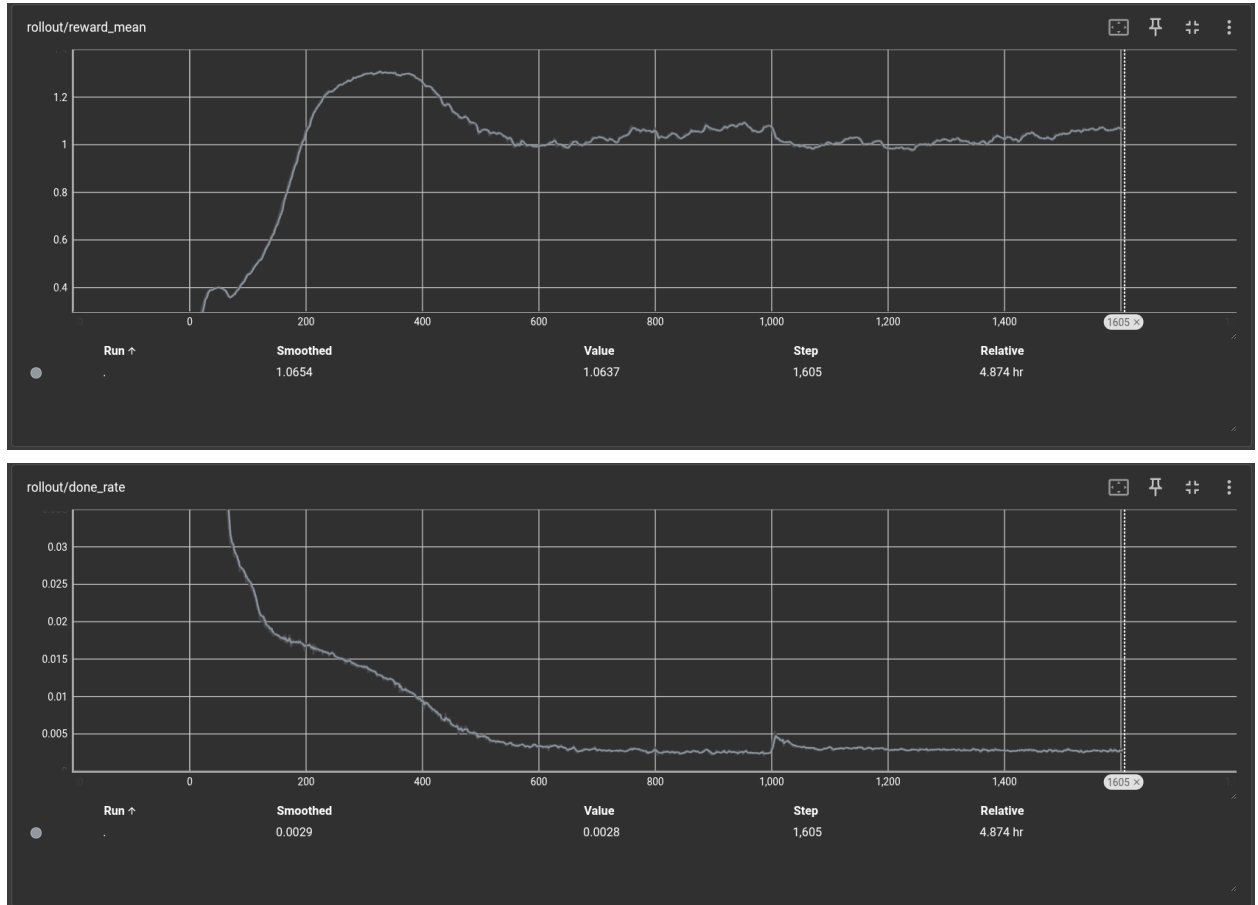


Figure 2: Mean reward per step (top) and termination rate (bottom) over 1,605 PPO updates. The curriculum switches to Phase 1 at update 1,000.

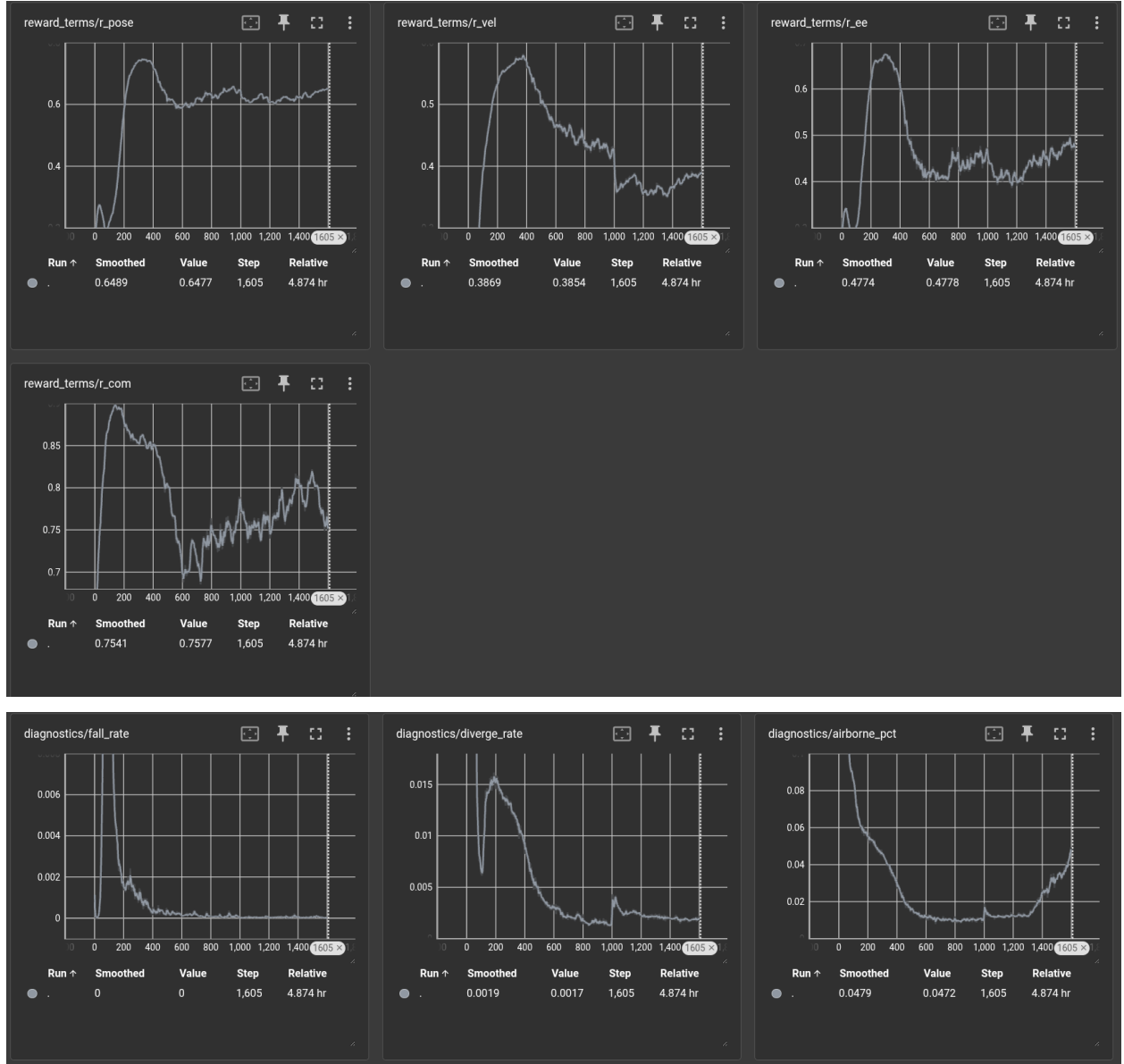


Figure 3: Top: the four tracking terms r_{pose} , r_{vel} , r_{ee} and r_{com} . Bottom: fall rate, divergence rate and airborne rate (both feet off the ground; logged only, not rewarded).

bias of $+8-9^\circ$ (vs. about -1° with penalties), a habit learned early when the right arm was stuck against the torso.

Table 6: Termination-penalty ablation (one seed each, start frame 10).

Run	Update $\sim 500-600$	Update $\sim 900-1000$
Fall + divergence	Walks cleanly	Pelvis within $\sim 0.05-0.15$ m, y within ± 0.1 m
Fall only	Walks cleanly, arms fine	–
Divergence only	Falls behind the reference, drifts left, falls	Path tracking catches up; movement wobbly
None	Falls behind the reference and falls ($\sim$ step 939); right arm stuck against torso	Path tracking similar; right shoulder-roll bias $+8-9^\circ$ vs. $\sim -1^\circ$

7 Limitations

This is a single case study, and its evidence is limited. The ablation uses one seed per variant, so the differences in Table 6 should be read as indications, not established effects; single-run results are common in this area (DeepMimic also reports most of its results from one run [1]), but they remain a limitation. The ablation was also run before the rollout fix (Section 4.7) and may overstate the effect of the penalties; it should be repeated with the corrected loop. We do not correct the floating reference (Section 4.6). We use a single walking clip, a single robot and simulation only; nothing here has been tested on hardware. We do not compare against LocoMuJoCo’s H1 baseline [2], which by our own observation walks more smoothly and in more directions; the purpose of this paper is the failure analysis, not the final policy. The elbow links are only an approximation of DeepMimic’s hand end-effectors. Finally, a residual asymmetry remains: the left foot lands about 8 cm narrower than in the reference in every run, and we have not yet found its cause.

8 Conclusion

Building a motion-imitation pipeline from scratch exposed many failures that the reward curve did not show. The most important ones looked like reward-design problems (drift, shuffling feet, still arms, hopping) but came from the action space, coordinate frames, reward scales, reset logic and the training loop. Once those were fixed, the plain DeepMimic reward was enough for the Unitree H1 to walk a 14.7-minute reference clip without falling. We hope the catalogue and checklist save time for others building similar systems, and we suggest that per-joint comparison against the reference should come before reward tuning, not after.

Acknowledgements

The diagnostic scripts described in Section 5 and the drafting of this manuscript were done with the help of an AI assistant (Claude, Anthropic). The pipeline, experiments, results and conclusions are the author’s own. The author has checked all content and takes full responsibility for it.

Code and Data

Code: <https://github.com/Bhargavoza1/h1-imitation-learning>. Reference motion: LocoMuJoCo datasets [2]. Robot model: MuJoCo Menagerie [15].

References

- [1] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 37(4), 2018. doi:[10.1145/3197517.3201311](https://doi.org/10.1145/3197517.3201311).

- [2] F. Al-Hafez, G. Zhao, J. Peters, and D. Tateo. LocoMuJoCo: A comprehensive imitation learning benchmark for locomotion. *arXiv preprint arXiv:2311.02496*, 2023.
- [3] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.
- [4] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. <http://github.com/jax-ml/jax>.
- [5] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence*, 2018.
- [6] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry. Implementation matters in deep policy gradients: A case study on PPO and TRPO. In *International Conference on Learning Representations (ICLR)*, 2020.
- [7] S. Huang, R. F. J. Dossa, A. Raffin, A. Kanervisto, and W. Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022.
- [8] Z. Luo, J. Cao, A. Winkler, K. Kitani, and W. Xu. Perpetual humanoid control for real-time simulated avatars. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- [9] Q. Liao, T. E. Truong, X. Huang, Y. Gao, G. Tevet, K. Sreenath, and C. K. Liu. BeyondMimic: From motion tracking to versatile humanoid control via guided diffusion. *Science Robotics*, 11(117):eadx8924, 2026. arXiv:2508.08241. doi:10.1126/scirobotics.adx8924.
- [10] X. B. Peng. MimicKit: A reinforcement learning framework for motion imitation and control. *arXiv preprint arXiv:2510.13794*, 2025.
- [11] F. Amadio and E. Mingo Hoffman. What matters in humanoid general motion tracking? An empirical study. *arXiv preprint arXiv:2607.19903*, 2026.
- [12] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2022.
- [13] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023.
- [14] R. Bouchoucha, A. Haj Yahmed, D. Patil, J. Rajendran, A. Nikanjam, S. Chandar, and F. Khomh. Toward debugging deep reinforcement learning programs with RLEplorer. In *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2024. arXiv:2410.04322. doi:10.1109/ICSME58944.2024.00019.
- [15] K. Zakka, Y. Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022. http://github.com/google-deepmind/mujoco_menagerie.
- [16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [17] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.