SuperMimic: Scaling Humanoid Whole-Body Control with One Differentiable World Model

Qiayuan Liao Yiyang Shao* Junfeng Long* Kevin Zakka
Roberto Horowitz Pieter Abbeel† Koushil Sreenath†

UC Berkeley
https://supermimic.github.io

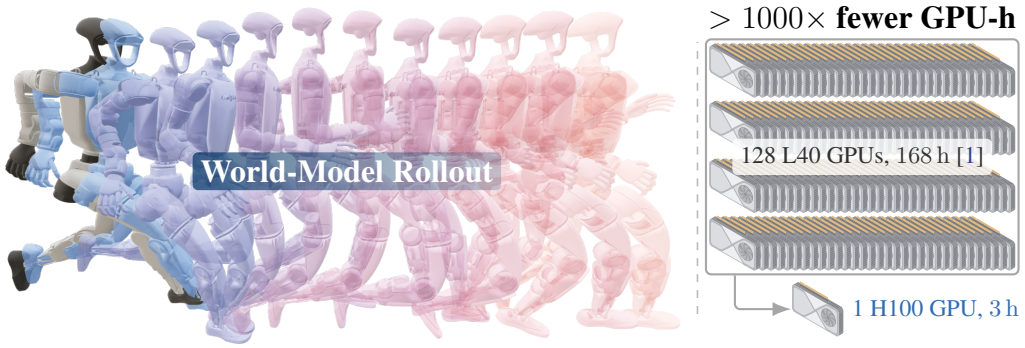

Figure 1: **A learned differentiable world model trains a humanoid whole-body controller and reduces GPU-hours by more than $1000\times$ versus cluster-scale model-free reinforcement learning (§4.1).** Left: the policy improves through imagined rollouts of the world model. Right: 128 L40 GPUs train for $168\,\mathrm{h}$ in SONIC [1], against one H100 for $3\,\mathrm{h}$ in this work.

Abstract: Tracking large human-motion datasets has become the dominant route to general, expressive humanoid whole-body control, but scaling across many motions is expensive. Current pipelines use model-free reinforcement learning, either merging per-motion specialists through lossy distillation, or brute-forcing a single policy with cluster-scale compute. SuperMimic amortizes learning across all motions with one differentiable world model of their shared dynamics, fit from logged transitions of a standard rigid-body simulator. The world model and the tracking policy train together in one loop, and backpropagation-through-time over short imagined rollouts gives the policy analytic gradients. Fast imagined rollouts and analytic gradients make policy learning efficient. In 3 hours on a single GPU, SuperMimic surpasses the return of a state-of-the-art motion tracker baseline trained for 7 days on 128 GPUs. It reduces GPU-hours by more than $1000\times$ and fails on only 25 of $129{,}514$ benchmark motions, versus 179 failures for the baseline. After 9.8 hours on two GPUs, SuperMimic achieves zero failures. From 1 to 264 hours of motion, the world model's prediction error stays stable, and GPU-hours to a target return grow slowly. Our policy transfers to a physical Unitree G1 across diverse and acrobatic motions. SuperMimic removes compute as the bottleneck to scaling humanoid whole-body control, making it practical to measure motion-tracking scaling laws.

Keywords: Humanoid Robot, Whole-Body Control, Model-Based Reinforcement Learning

1 Introduction

Humanoid robots hold the promise of operating in environments designed for people, but doing so requires whole-body controllers that span diverse human motion, from locomotion and dance to crouching and balance recovery. Physics-based motion imitation addresses this by training RL

*Equal contribution. †Equal advising.

10th Conference on Robot Learning (CoRL 2026), Austin TX, USA.

policies to track reference motion clips [2] drawn from large motion-capture (mocap) datasets [1]. Existing systems scale this idea either by distilling many clip-specific experts into one controller [3] or by training a generalist policy over a large motion library [1]. In both cases, a more diverse motion library produces a more capable controller. But both also become computationally expensive as motion coverage grows: the state-of-the-art tracker SONIC [1], for example, trains for $\sim 168\,\mathrm{h}$ on 128 GPUs, totaling roughly $21{,}000$ GPU-hours. Although reference motions differ, they share the same humanoid hybrid rigid-body dynamics. Model-free RL fails to explicitly model this shared structure, instead repeatedly estimating action consequences from noisy simulator rollouts. Scaling to more motions therefore amounts to scaling up compute.

SuperMimic addresses this inefficiency with a single learned, differentiable world model fit from logged simulator transitions. Backpropagation-through-time (BPTT) through short imagined rollouts provides analytic policy gradients through the learned model, allowing one dynamics model to amortize learning across the entire mocap library. Because these gradients come from the learned model rather than from differentiating the simulator, SuperMimic remains compatible with rigid-body simulators, including those with non-smooth contacts and non-differentiable implementations.

On one GPU, SuperMimic surpasses the normalized return of SONIC's released checkpoint within $3\,\mathrm{h}$ and reduces GPU-hours by more than $1000\times$ (§4.1). On a $129{,}514$-motion benchmark, SuperMimic fails on 25 motions after $3\,\mathrm{h}$ on one GPU and on none after $9.8\,\mathrm{h}$ on two GPUs, against 179 for SONIC's released checkpoint (Appendix F). Across a $264\times$ increase in dataset length, the world model's prediction error stays stable and SuperMimic's compute to a fixed return grows slowly. The learned policy transfers to a physical Unitree G1. In a preliminary study of downstream interfaces, an RL-trained velocity encoder outperforms a distillation baseline.

Contributions.

- **Model-based RL (MBRL) for deployable humanoid control**. We demonstrate that BPTT through a learned, differentiable world model can train whole-body tracking policies in a setting long dominated by model-free proximal policy optimization (PPO) [4], and that the resulting controller transfers to a physical Unitree G1.
- **Compute is no longer the bottleneck to scaling.** SuperMimic surpasses the return of SONIC, a cluster-scale PPO pipeline, on a single GPU and reduces GPU-hours by more than $1000\times$ (§4.1). Its training cost grows slowly as the motion library expands, making scaling humanoid whole-body control practical without huge GPU infrastructure.
- **An analysis of the ingredients behind the gains.** We identify three world-model properties behind the compute reduction: prediction error that stays stable over a $264\times$ increase in dataset length, rollouts more than $10\times$ faster than the simulator, and analytic gradients, with which the policy reaches the best return of PPO on the world model with about $64\times$ fewer imagined steps.
- **RL-trained downstream control beyond distillation.** As a preliminary study, the same learned world model trains a downstream goal encoder on the interface's own task reward. On velocity commands, the policy tracks more accurately with this encoder than with a distillation baseline.

2 Related Work

Large-scale physics-based motion tracking. Physics-based motion imitation learns controllers that reproduce reference motions under contact-rich rigid-body dynamics. DeepMimic [2] demonstrated expressive character skills through model-free reinforcement learning, and PHC [11] extended this paradigm to large motion sets with robust recovery. SONIC [1] further demonstrated that scaling motion data, policy capacity, and training compute produces generalist controllers that transfer to humanoid hardware. These results establish motion tracking as a practical route to broad whole-body behavior coverage. However, training these large-scale motion trackers requires extensive simulator rollouts and substantial computation. SuperMimic addresses this bottleneck by learning a differentiable dynamics model shared across all motions, so policy improvement can exploit analytic gradients rather than model-free rollout estimates.

Method	Mimic	Latent	Downstream	Task signal	Real robot
NPMP [5]	SVG(0)	BC	SVG(0)	RL	✗
ControlVAE [6]	MBRL		MBRL	RL	✗
PhysicsVAE [7]	PPO	BC[1]	PPO	RL	✗
PULSE [8]	PPO	DAgger	PPO	RL	✗
MaskedMimic [9]	A2C	DAgger		Distill	✗
BeyondMimic [3]	PPO	DAgger	CGD	Distill	✓
SONIC [1]	PPO			Distill[2]	✓
LATENT [10]	PPO	DAgger	PPO	RL	✓
SuperMimic	MBRL			Distill / RL	✓

Table 1: **Comparison of various methods and their training strategies across different stages.** The Mimic, Latent, and Downstream columns indicate the training strategy used at each stage. Task signal is what trains the downstream interface: its own task reward (RL) or a reference-tracking signal (Distill). Merged cells denote jointly trained stages, while adjacent unmerged cells indicate sequential training, with each stage trained after freezing the previous one.
SVG(0): stochastic value gradients; BC: offline behavior cloning on fixed expert rollouts, as opposed to DAgger's student rollouts; CGD: classifier guidance diffusion; A2C: advantage actor–critic; DAgger: dataset aggregation. [1]Behavior cloning with a one-step next-state consistency term through a frozen, learned dynamics model. [2]Goal encoders train jointly with the tracker on the motion-tracking reward and latent-alignment losses; no task-specific reward.

Learned differentiable dynamics for motion control. SuperTrack [12] optimizes motion-control policies through a learned differentiable surrogate of the simulator. ControlVAE [6] jointly learns a dynamics model, latent skill representation, and generative controller, while MoConVQ [13] extends learned-model training to discrete motion representations over tens of hours of data. SuperDyno [14] scales differentiable neural dynamics to tens of thousands of motion clips. These methods establish the effectiveness of learned dynamics and analytic policy gradients for scalable simulated-character control. SuperMimic extends this direction to single-GPU training of a large-scale humanoid tracker and to its sim-to-real deployment: its world model conditions on simulator context, while the deployed actor uses noisy proprioceptive observations and a goal input (§3.2). We evaluate how model accuracy, rollout throughput, and differentiable policy updates contribute to training efficiency, and how prediction error behaves as motion coverage grows (§4.1–4.2), and demonstrate physical deployment of the resulting controller (§4.4).

Model-based policy optimization and robot deployment. Dreamer [15] and DreamerV3 [16] learn latent dynamics for imagination-based actor–critic learning, while TD-MPC [17] combines short-horizon planning with terminal-value prediction. DayDreamer [18] demonstrates world-model learning directly on physical robots. SHAC [19] uses differentiable simulation for short-horizon actor–critic updates, motivating SuperMimic's critic-bootstrapped objective. Analytic-gradient robot control can also combine a simplified differentiable surrogate with a high-fidelity forward simulator, as demonstrated for quadruped locomotion [20]. SuperMimic learns its surrogate from simulator transitions and backpropagates through short imagined rollouts, retaining compatibility with standard rigid-body simulators.

RWM [21] and RWM-U [22] demonstrate that policies trained in learned neural simulators can transfer to physical robots. Both optimize policies using PPO on imagined rollouts; RWM-U additionally uses uncertainty-penalized offline optimization. SuperMimic instead optimizes the policy by differentiating the return through its learned dynamics. RWM and RWM-U pretrain the world model on trajectories from other policies. In contrast, SuperMimic fits its world model during training on data its own policy collects.

Reusable motion representations and downstream control. Reusable controllers connect motion tracking to downstream goal interfaces (Table 1). NPMP [5] and PhysicsVAE [7] learn latent motor representations from expert behavior; PULSE [8] and LATENT [10] use learned motion representations for downstream reinforcement learning. MaskedMimic [9] supports partial-body and multimodal goals through motion inpainting, while BeyondMimic [3] uses guided diffusion for goal

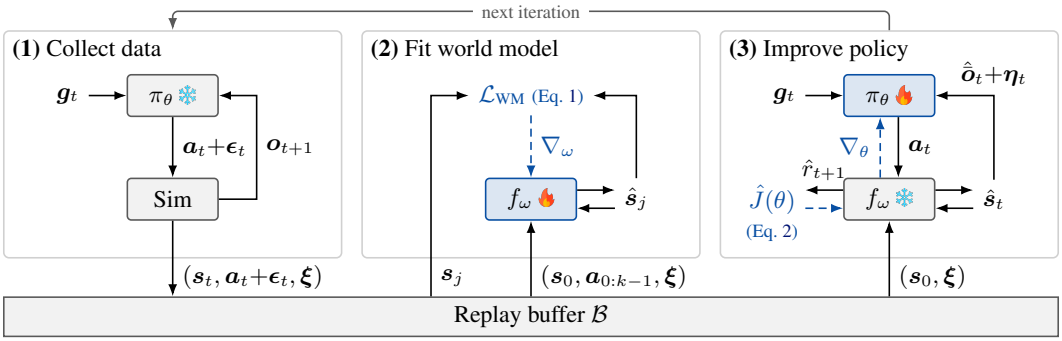

Figure 2: **SuperMimic training loop.** (1) The current policy π_θ collects transitions in simulation into a fixed-size replay buffer $\mathcal{B}$, where new transitions replace the oldest. (2) The differentiable world model is fit to k-step windows sampled from $\mathcal{B}$ (§3.2). (3) The policy is improved by backpropagation-through-time through the frozen world model over H imagined steps; a critic V_ψ, trained in the same phase, bootstraps the return beyond H (§3.3). Dashed arrows are gradients; ❄ = frozen, 🔥 = trained. Symbols: goal $\boldsymbol{g}_t$ (e.g., reference motion), action $\boldsymbol{a}_t$, action noise $\boldsymbol{\epsilon}_t$, observation $\boldsymbol{o}_t$, state $\boldsymbol{s}$ and predicted state $\hat{\boldsymbol{s}}$, and sim context $\boldsymbol{\xi}$ (domain-randomization parameters). $\mathcal{L}_{\text{WM}}$ is the world-model loss, $\hat{\boldsymbol{o}}_t$ the predicted noise-free observation, $\boldsymbol{\eta}_t$ the sensor noise, $\hat{r}_{t+1}$ the predicted reward, and $\hat{J}(\theta)$ the policy objective.

following. SONIC [1] jointly trains goal encoders with its tracker using tracking rewards and latent-alignment losses. ControlVAE [6] optimizes downstream task policies through its learned model while freezing the skill controller. SuperMimic jointly trains goal-specific encoders and a shared decoder through the same dynamics model (Appendix B). Goal interfaces can optimize either a reference-tracking reward or their own task reward. Our preliminary study trains a velocity interface concurrently with the motion-tracking interface, once on each objective (§4.5).

3 Method

3.1 SuperMimic training loop

SuperMimic trains one goal-conditioned whole-body controller across a multi-skill motion library and supports multiple goal interfaces. It captures the shared humanoid dynamics in a differentiable world model and improves the policy through short imagined rollouts. Training alternates between three phases in one loop (Figure 2). First, the current policy collects simulator transitions into a replay buffer. Second, we fit the differentiable world model to short replay windows. Third, the policy is improved by backpropagating an imagined return through the frozen world model, with each rollout seeded from a replay-buffer state in the Dyna style [23, 24]. The buffer, model, and policy are updated throughout training, so dynamics learning and control improve together rather than in separate stages (Algorithm A.1, Appendix D). Transitions from earlier policies remain valid for fitting the world model, because the dynamics do not depend on the policy that collected them.

We denote the goal-conditioned partially observable Markov decision process (POMDP) by $\mathcal{M}$ and use compact notation with state $\boldsymbol{s}_t$, goal $\boldsymbol{g}_t$, and action $\boldsymbol{a}_t$. State $\boldsymbol{s}_t$ gives the proprioceptive observation $\boldsymbol{o}_t$ and the termination d_t, which is 1 if $\boldsymbol{s}_t$ is terminal. Action $\boldsymbol{a}_t$ in state $\boldsymbol{s}_t$ yields the reward r_{t+1} and the next state $\boldsymbol{s}_{t+1}$. The policy is deterministic, $\boldsymbol{a}_t = \pi_\theta(\boldsymbol{o}_t, \boldsymbol{g}_t)$, and $\boldsymbol{a}_t$ parameterizes a normalized joint setpoint; for motion tracking, $\boldsymbol{g}_t$ is read from a reference motion. During data collection, the simulator executes the noisy action $\boldsymbol{a}_t + \boldsymbol{\epsilon}_t$ with action noise $\boldsymbol{\epsilon}_t \sim \rho_\epsilon$, and the replay buffer stores this executed noisy action. The noise widens the range of actions on which the world model is fit; it serves only the model fit and is optional. Returns use a soft survival weight, $c_0 = 1$ and $c_{t+1} = c_t(1 - d_{t+1})$, giving $J(\theta) = \mathbb{E}[\sum_{t\geq 0} \gamma^t c_t r_{t+1}]$ with discount $\gamma \in [0, 1)$. The simulator logs hard terminations, while the learned model predicts a termination probability $\hat{d}_t \in [0, 1]$, which keeps the survival weight differentiable in imagined rollouts. Throughout, a hat marks the world-model counterpart of a simulator quantity. We also factor the observation as

$\boldsymbol{o}_t = \bar{\boldsymbol{o}}_t + \boldsymbol{\eta}_t$, with $\bar{\boldsymbol{o}}_t$ the noise-free observation and known observation noise $\boldsymbol{\eta}_t \sim \rho_\eta$ sampled outside the world model. Appendix A gives the full spaces and reward.

The default interface is full-reference motion tracking. Other goal interfaces, such as velocity, share the same learned dynamics and train jointly with motion tracking (Appendix B).

3.2 Learning a differentiable world model

The simulator's articulated multi-rigid-body dynamics are known, but the simulator need not provide gradients, so SuperMimic learns a differentiable surrogate to obtain policy gradients: it supplies the predictions and Jacobians for the BPTT update in §3.3. The resulting analytic policy gradient is the gradient of a surrogate for the expected return in simulation. Since the simulator's randomness has known sources, the world model stays deterministic: it conditions on the per-episode sim context $\boldsymbol{\xi}$ (friction, calibration offsets, and torso center-of-mass) and predicts the noise-free observation $\bar{\boldsymbol{o}}_t$.

The world model contains a learned dynamics model and reward, observation, and termination heads; each head is either learned or computed analytically,

$$\hat{\boldsymbol{s}}_{t+1} = f_\omega(\hat{\boldsymbol{s}}_t, \boldsymbol{a}_t, \boldsymbol{\xi}), \qquad \hat{r}_{t+1} = r_\omega(\hat{\boldsymbol{s}}_t, \boldsymbol{a}_t, \boldsymbol{g}_t), \qquad \hat{\bar{\boldsymbol{o}}}_t = g_\omega(\hat{\boldsymbol{s}}_t, \boldsymbol{\xi}), \qquad \hat{d}_t = d_\omega(\hat{\boldsymbol{s}}_t, \boldsymbol{g}_t).$$

We fit the network weights ω on k-step autoregressive replay windows (k=8), seeded at the logged state $\hat{\boldsymbol{s}}_0 = \boldsymbol{s}_0$ and rolled forward on logged actions, by minimizing

$$\mathcal{L}_{\text{WM}}(\omega) = \mathbb{E}_{\mathcal{B}} \sum_{j=1}^{k} \Big[\|\hat{\boldsymbol{s}}_j - \boldsymbol{s}_j\|^2 + (\hat{r}_j - r_j)^2 + \|\hat{\bar{\boldsymbol{o}}}_j - \bar{\boldsymbol{o}}_j\|^2 + \text{BCE}(\hat{d}_j, d_j) \Big], \tag{1}$$

where an analytic head has no loss term. Here r_j is the simulator reward, $d_j \in \{0, 1\}$ is the simulator termination label, and BCE is the binary cross-entropy.

3.3 Learning the policy through the world model

During each policy update, we hold the world-model weights ω fixed and optimize π_θ and V_ψ on imagined rollouts of H=32 steps. The critic V_ψ estimates the return beyond step H, so the model must be accurate only within these H steps. This imagination horizon exceeds the k-step replay windows on which we fit the model (§3.2). In practice, the learned model remains stable over indefinitely long rollouts, because the k-step autoregressive fit trains it to correct its own prediction errors.

Each imagined rollout starts from a replay-buffer state $\boldsymbol{s}_0 \sim \rho_0$ with its recorded sim context $\boldsymbol{\xi}$. At step t, the model emits $\hat{\bar{\boldsymbol{o}}}_t$, we re-add sensor noise, and the policy acts: $\boldsymbol{a}_t = \pi_\theta(\hat{\bar{\boldsymbol{o}}}_t + \boldsymbol{\eta}_t, \boldsymbol{g}_t)$, $\boldsymbol{\eta}_t \sim \rho_\eta$. The policy objective over imagined rollouts is

$$\hat{J}(\theta) = \mathbb{E}_{\boldsymbol{s}_0 \sim \rho_0,\, \pi_\theta} \Big[\sum_{t=0}^{H-1} \gamma^t \hat{c}_t\, \hat{r}_{t+1} \;+\; \gamma^H \hat{c}_H\, V_\psi(\hat{\boldsymbol{s}}_H, \boldsymbol{g}_H, \boldsymbol{\xi}) \Big]. \tag{2}$$

For motion tracking, the reward r_{t+1} is the BeyondMimic motion-tracking reward [3] without its self-collision penalty (Appendix A).

The world model is differentiable, so BPTT computes a first-order gradient $\nabla_\theta \hat{J}$ through the H imagined steps via the world model's Jacobians with respect to $\hat{\boldsymbol{s}}_t$ and $\boldsymbol{a}_t$. In contrast, PPO treats the dynamics as a black box: it weights the score $\nabla_\theta \log \pi_\theta$ of a stochastic policy by the advantages of sampled actions, a likelihood-ratio estimate. BPTT instead differentiates the return with respect to each action, so over short horizons its gradient has much lower variance.

The critic V_ψ is fit to generalized advantage estimation returns [25] computed backward in time over the imagined transitions only,

$$\mathcal{L}_V(\psi) = \mathbb{E} \sum_{t=0}^{H-1} \big(V_t - \hat{R}_t\big)^2, \qquad \hat{R}_t = \hat{r}_{t+1} + \gamma(1 - \hat{d}_{t+1}) \big[(1-\lambda) V_{t+1} + \lambda \hat{R}_{t+1}\big], \tag{3}$$

with $V_t = V_\psi(\hat{\boldsymbol{s}}_t, \boldsymbol{g}_t, \boldsymbol{\xi})$. The recursion starts from $\hat{R}_H = V_H$, the critic's estimate of the return beyond the horizon. The targets $\hat{R}_t$ are computed once per rollout and remain fixed while ψ is fit.

The policy and critic updates together form a short-horizon actor–critic in the spirit of SHAC [19], which backpropagates through a differentiable simulator. SuperMimic replaces SHAC's hard episode termination with the survival weight $\hat{c}_t$, and its critic has no exponential-moving-average target network. Prior motion-tracking controllers trained with analytic gradients often omit a critic and optimize only the H-step reward sum [12, 6]. Without a critic, training can reach a similar nominal tracking return, but this unbootstrapped objective is more myopic. The critic matters after perturbations, where recovery or balance rewards can lie beyond the imagination window (§4.3).

4 Experimental Results

We show that SuperMimic scales favorably relative to PPO pipelines, identify the world-model properties behind this scaling, demonstrate transfer on a real Unitree G1, and report a preliminary study of downstream goal interfaces. The simulator is MuJoCo Warp, a GPU implementation of MuJoCo [26]. All our reported costs are measured on NVIDIA H100 (80 GB SXM5) GPUs of Lambda on-demand instances. We ran several random seeds and found training very stable, which we attribute to the dense reward that motion tracking gives at every step. The figures therefore do not report seed variation. Normalized return, return for short, is the undiscounted sum of the per-step reward over the full episode, divided by the largest value this sum can take. A policy that terminates early therefore scores low.

4.1 SuperMimic reaches SONIC-level tracking with far fewer GPU-hours

We compare against SONIC, a published PPO pipeline trained on a 128-GPU cluster. Both methods build on the BeyondMimic motion-tracking formulation [3], making SONIC the closest available published comparison. Within 3 h on one GPU, SuperMimic surpasses the normalized return of the released SONIC checkpoint (Figure 3(a)). On a shared benchmark of 129,514 motions, it also fails on fewer motions than SONIC: 25 after 3 h on one GPU and 0 after 9.8 h on two GPUs, against 179 for SONIC's released checkpoint (Appendix F). We state the 3 h conservatively: from then on, every checkpoint is ahead of SONIC by at least half of the gap between SONIC and SuperMimic's best result, in both return and failed motions. SuperMimic thus uses more than $50\times$ less wall-clock time and more than $1000\times$ fewer GPU-hours than SONIC.

Compute accounting. The ratios compare SONIC's reported 128 NVIDIA L40 GPUs for 7 days (about 21,000 GPU-hours) with 3 h on one NVIDIA H100, within which SuperMimic surpasses the return of SONIC's released checkpoint. They are about $56\times$ ($> 50\times$) in wall-clock time and about $7200\times$ ($> 1000\times$) in GPU-hours. The 3 h cover the full training loop; checkpoint evaluations run after training and are not counted. SuperMimic trains on 264.56 h of motion from BONES-SEED [27], the public subset of SONIC's data, about $2.3\times$ less than SONIC's 611 h. The GPU models also differ: in peak throughput, an H100 delivers $0.74\times$ the fp32 and $5.5\times$ the tensor-core performance of an L40; how much of that peak either run used is unknown. Nonetheless, $2.3\times$ in data and at most $5.5\times$ in peak GPU throughput cannot account for a $1000\times$ gap in GPU-hours.

Figure 4(a) gives a controlled comparison on LAFAN1 [28]. The baseline, PPO-Sim, is PPO trained in the simulator. PPO-Sim shares SuperMimic's motions, observations, reward, termination, domain randomization, actor and critic network sizes, and evaluation protocol. It uses BeyondMimic's PPO hyperparameters [3] and 8192 parallel environments per GPU, twice SuperMimic's. PPO-Sim estimates its policy gradient from sampled simulator rollouts; SuperMimic backpropagates its imagined return through world-model rollouts. SuperMimic reaches a return of 0.82 in evaluation within 1.2 h on one GPU. Within the 18 h budget, PPO-Sim does not reach it. Its final return is 0.60 on one GPU, 0.70 on four, and 0.72 on eight, so going from four to eight GPUs barely helps.

Figure 3(b, c) shows that, for SuperMimic, GPU-hours to the target return grow slowly as dataset length increases, while on the same sweep the world model's prediction error stays stable; §4.2 traces both trends to one world model amortizing the dynamics shared across motions.

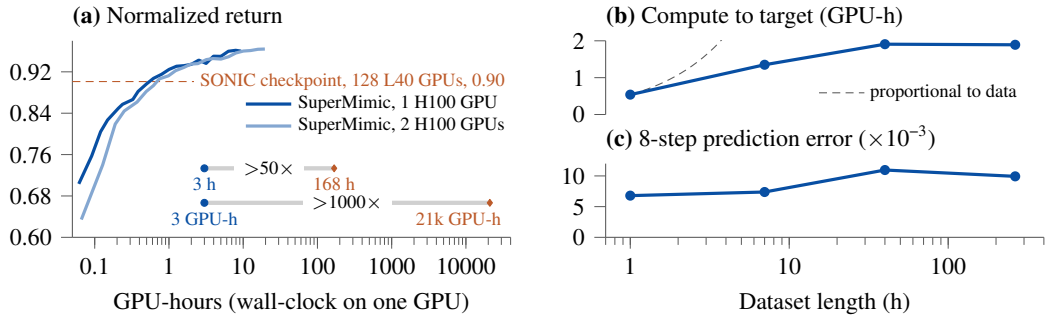

Figure 3: **SuperMimic surpasses SONIC's tracking return with far fewer GPU-hours.** (a) SuperMimic on one and two H100 GPUs surpasses the return of the released SONIC checkpoint (dashed), all three scored on the same benchmark of 129,514 motions (264 h). Two budget rows compare SuperMimic (dot) with SONIC (diamond): SuperMimic reduces wall-clock time by more than 50× and GPU-hours by more than 1000×. (b, c) On a shared dataset-length sweep, GPU-hours to target return (b) grow slowly while the world model's 8-step prediction error (c) stays within 0.007–0.011 as the dataset grows from 1 to 264 h, consistent with one world model amortizing shared body and contact dynamics across motions. The prediction error counts the state only, is normalized by a predefined nominal error, and is measured online on policy rollouts.

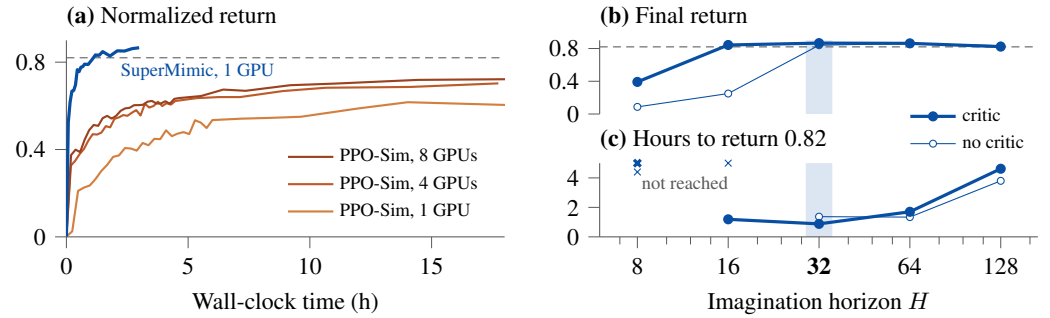

Figure 4: **In controlled comparisons, SuperMimic reaches a higher return sooner than PPO-Sim, and the critic enables shorter imagination horizons.** All runs use LAFAN1 (40 clips, 2.45 h). (a) SuperMimic backpropagates its imagined return through its world model on one GPU; PPO-Sim runs PPO in the simulator on 1, 4, or 8 GPUs (§4.1). Each point on a curve is a saved checkpoint scored on all 40 clips. (b, c) Final training return (b) and wall-clock hours until the training return reaches 0.82 (c) for each imagination horizon H, with and without the critic. The dashed line in (a) and (b) marks this target return of 0.82. Training return is the return of training episodes. Their initial states follow return-based reset sampling (Appendix D.1) and shift during training, so training return differs from the checkpoint scores in (a). In (c), a cross marks a horizon at which return 0.82 is not reached: both variants at $H{=}8$, no critic at $H{=}16$. The shaded band marks the default $H{=}32$.

4.2 Properties of the learned world model

These results rest on three properties of the learned world model f_ω: its accuracy holds as the motion library grows, and its rollouts are fast and differentiable.

One model amortizes the motion library. In Figure 3(c), the world model's 8-step prediction error stays between 0.007 and 0.011 as the dataset grows from 1 to 264 h. Every motion is executed by the same humanoid, so added clips supply more evidence about the shared body and contact dynamics rather than demanding new capacity. This is consistent with the slow growth of the compute needed to reach the target return (§4.1).

Imagination accelerates rollouts. During policy optimization, rollout speed sets the update cost. Figure 5(a) shows that the world model runs more than 10× faster than MuJoCo Warp at the batch sizes used in practice, and up to 62× at 64k parallel environments.

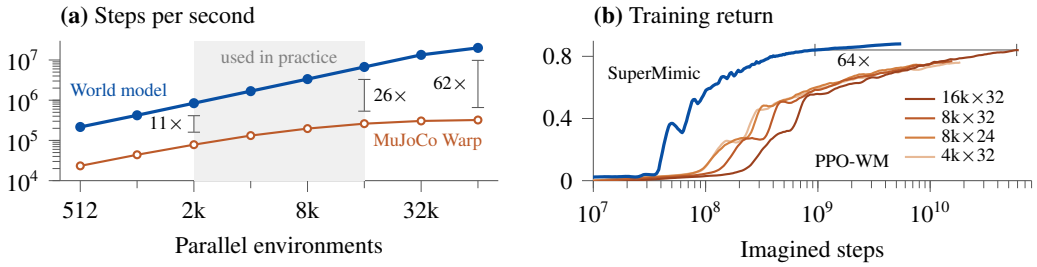

Figure 5: **Imagined rollouts are fast and differentiable.** (a) On an H100, over the batch sizes used in practice (shaded, 2k–16k parallel environments), the world model runs more than $10\times$ faster than MuJoCo Warp; the gap widens to $62\times$ at 64k, where MuJoCo Warp has flattened. One step is one $50\,\mathrm{Hz}$ control step, which MuJoCo Warp computes as four $5\,\mathrm{ms}$ substeps; the world model runs at the TF32 precision used in SuperMimic's training. (b) On walk/run/sprint motions from LAFAN1 [28], SuperMimic, trained with its actor–critic update, reaches the best training return of PPO trained on world-model rollouts (PPO-WM) with about $64\times$ fewer imagined steps. PPO-WM runs at four batch sizes (parallel rollouts $\times$ steps per rollout); 16k$\times$32 is the best. Each curve is a centered moving average over ± 0.05 decade of imagined steps.

First-order gradients need fewer imagined steps. As a mechanism ablation, Figure 5(b) isolates the benefit of first-order policy updates: SuperMimic reaches the best PPO-WM return with about $64\times$ fewer imagined steps, so the gain is not just rollout speed.

4.3 A critic enables shorter horizons and push recovery

Figure 4(b, c) sweeps H with and without the critic. Without the critic, training fails at $H \leq 16$. With it, $H{=}16$ matches the critic-free $H{=}32$, and $H{=}32$ reaches a return of 0.82 fastest. Raising H from 64 to 128 nearly triples the time to that return and lowers the final return; $H{=}128$ is the longest horizon that fits on an H100. At matched return, the critic still changes behavior. Under 0.5–$4\,\mathrm{m/s}$ velocity pushes while holding a static T-pose in simulation, the critic policy settles within 0.5–$2.4\,\mathrm{s}$. The critic-free policy never settles: it keeps stumbling even at $0.5\,\mathrm{m/s}$ and falls at $4\,\mathrm{m/s}$.

4.4 Real-robot transfer

One general motion tracker deploys on a real Unitree G1 without per-motion tuning (Figure 6). The four motions cover locomotion (a skipping jump), contact-rich whole-body motion with the torso and limbs on the floor (a side roll), and acrobatics (a 90° flip and an in-place 360° flip).

4.5 Downstream tasks: RL encoder vs. distillation baseline

SuperMimic trains downstream goal encoders jointly with the motion-tracking interface, sharing one decoder and world model (Appendix B). Our study of downstream interfaces is preliminary. We train one velocity encoder on a velocity-tracking reward through the world model. We compare it with a distillation baseline: a velocity encoder trained on the motion-tracking reward, with commands read from the reference motions. Figure 7 evaluates both on 672 step responses spanning the velocity space (Appendix G). The RL encoder has the lower steady-state error on 650 of them, and the distillation encoder on 15. Only the distillation encoder falls, on seven fast turns with almost no motion-library data.

5 Conclusion

SuperMimic removes compute as the bottleneck to scaling humanoid whole-body control. A single learned, differentiable world model captures the dynamics every motion shares, and backpropagation-through-time through it amortizes a whole dataset into one policy. The policy trains on a single GPU, surpasses the return of a policy trained on a 128-GPU cluster, and transfers to a real Unitree G1. Its training cost grows slowly over a controlled dataset-length sweep. The

Figure 6: **One general motion tracker, trained through the learned world model, transfers to a real Unitree G1.** (a) Skipping jump, (b) side roll, (c) 90° flip, and (d) in-place 360° flip.

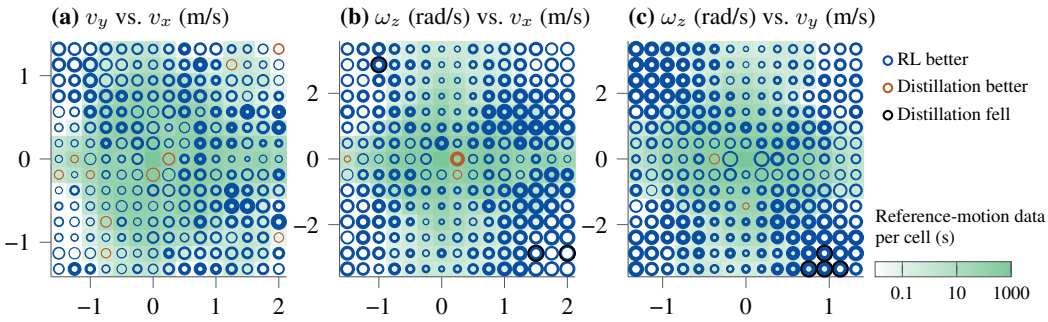

Figure 7: **The policy with the RL-trained velocity encoder tracks step commands more accurately than the distillation baseline.** Each panel is a plane of the command space with the third component at zero, and each ring is one command. The outer disc's area is proportional to the larger of the two policies' relative steady-state errors in the step response to that command, and the hole's area to the smaller one; the color marks the better policy. Details in Appendix G.

same model can train a downstream velocity interface directly on the interface's own task reward. Measuring the scaling laws of motion tracking is now affordable, at one GPU per run.

6 Limitations and Future Work

- **Multi-interface training is understudied.** We train the velocity encoder concurrently with the motion tracker, but the velocity-tracking reward updates only the velocity encoder and not the shared decoder (Appendix B). Concurrent training is therefore the same as training the velocity encoder after the motion tracker, and might not be necessary. Whether concurrent training helps once the velocity-tracking reward also updates the decoder is untested.
- **Simulator-only dynamics.** The world model learns only from simulator transitions. Real-robot data could refine it, and its differentiability could support system identification.
- **No terrain or object interaction.** SuperMimic trains only on flat ground. Extending it to terrain and object interaction is ongoing work.

Acknowledgments

We thank Bike Zhang, Sangli Teng, Nima Rahmanian, and Songbo Hu for helpful discussions and help with the experiments. This work was supported in part by Komatsu, the BAIR Center for Humanoid Intelligence, Apple, and the RAI Institute. We thank Lambda for sponsoring compute for this project. Motion Data by Bones Studio (https://bones.studio/). Pieter Abbeel and Koushil Sreenath hold concurrent appointments as Professors at UC Berkeley and as Amazon Scholars. This paper describes work performed at UC Berkeley and is not associated with Amazon.

References

[1] Z. Luo, Y. Yuan, T. Wang, C. Li, F. Castañeda, S. Chen, Z.-A. Cao, J. Li, D. Minor, Q. Ben, J. Park, D. Sami, Z. Wang, X. Da, R. Ding, C. Hogg, L. Song, E. Lim, E. Jeong, T. He, H. Xue, W. Xiao, S. Yuen, J. Kautz, Y. Chang, U. Iqbal, L. Fan, and Y. Zhu. SONIC: Supersizing motion tracking for natural humanoid whole-body control. *Science Robotics*, 11(117):eaed4592, 2026. doi:10.1126/scirobotics.aed4592.

[2] 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):143:1–143:14, 2018. doi:10.1145/3197517.3201311.

[3] 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. doi:10.1126/scirobotics.adx8924.

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

[5] J. Merel, L. Hasenclever, A. Galashov, A. Ahuja, V. Pham, G. Wayne, Y. W. Teh, and N. Heess. Neural probabilistic motor primitives for humanoid control. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJl6TjRcY7.

[6] H. Yao, Z. Song, B. Chen, and L. Liu. ControlVAE: Model-based learning of generative controllers for physics-based characters. *ACM Transactions on Graphics (TOG)*, 41(6):1–16, 2022. doi:10.1145/3550454.3555434.

[7] J. Won, D. Gopinath, and J. Hodgins. Physics-based character controllers using conditional VAEs. *ACM Transactions on Graphics (TOG)*, 41(4):1–12, 2022. doi:10.1145/3528223.3530067.

[8] Z. Luo, J. Cao, J. Merel, A. Winkler, J. Huang, K. Kitani, and W. Xu. Universal humanoid motion representations for physics-based control. In *International Conference on Learning Representations*, volume 2024, pages 56766–56782, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/fb19caf91f4966e426026685028cb73a-Abstract-Conference.html.

[9] 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)*, 43(6):1–21, 2024. doi:10.1145/3687951.

[10] Z. Zhang, H. Lu, Y. Lian, Z. Chen, Y. Liu, C. Lin, H. Xue, Z. Zeng, Z. Qi, S. Zheng, Q. Luan, J. Wang, J. Xing, H. Wang, and L. Yi. Learning athletic humanoid tennis skills from imperfect human motion data. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026. URL https://arxiv.org/abs/2603.12686.

[11] Z. Luo, J. Cao, A. Winkler, K. Kitani, and W. Xu. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10895–10904, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Luo_Perpetual_Humanoid_Control_for_Real-time_Simulated_Avatars_ICCV_2023_paper.html.

[12] L. Fussell, K. Bergamin, and D. Holden. SuperTrack: Motion tracking for physically simulated characters using supervised learning. *ACM Transactions on Graphics (TOG)*, 40(6):1–13, 2021. doi:10.1145/3478513.3480527.

[13] H. Yao, Z. Song, Y. Zhou, T. Ao, B. Chen, and L. Liu. MoConVQ: Unified physics-based motion control via scalable discrete representations. *ACM Transactions on Graphics (TOG)*, 43(4):1–21, 2024. doi:10.1145/3658137.

[14] Y. Lei, Z. Luo, T. He, J. Cao, G. Shi, and K. Kitani. Scalable humanoid whole-body control via differentiable neural network dynamics. In *ICLR 2025 Workshop on World Models: Understanding, Modelling and Scaling*, 2025. URL https://openreview.net/pdf/b3031990328dd52267a571b265c15328bf6d6120.pdf.

[15] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

[16] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025. doi:10.1038/s41586-025-08744-2.

[17] N. Hansen, X. Wang, and H. Su. Temporal difference learning for model predictive control. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8387–8406. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

[18] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel. DayDreamer: World models for physical robot learning. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 2226–2240. PMLR, 2023. URL https://proceedings.mlr.press/v205/wu23c.html.

[19] J. Xu, V. Makoviychuk, Y. Narang, F. Ramos, W. Matusik, A. Garg, and M. Macklin. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ZSKRQMvttc.

[20] Y. Song, S. Kim, and D. Scaramuzza. Learning quadruped locomotion using differentiable simulation. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 258–271. PMLR, 2025. URL https://proceedings.mlr.press/v270/song25a.html.

[21] C. Li, A. Krause, and M. Hutter. Robotic world model: A neural network simulator for robust policy optimization in robotics. *arXiv preprint arXiv:2501.10100*, 2025. URL https://arxiv.org/abs/2501.10100. Presented at the NeurIPS 2025 Workshop on Embodied World Models for Decision Making.

[22] C. Li, A. Krause, and M. Hutter. Uncertainty-aware robotic world model makes offline model-based reinforcement learning work on real robots. *arXiv preprint arXiv:2504.16680*, 2025. URL https://arxiv.org/abs/2504.16680. Presented at the 2nd Workshop on World Models at ICLR 2026.

[23] R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991. doi:10.1145/122344.122377.

[24] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html.

[25] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1506.02438.

[26] 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*, pages 5026–5033, 2012.

[27] Bones Studio. BONES-SEED: Skeletal everyday embodiment dataset, 2026. URL https://huggingface.co/datasets/bones-studio/seed.

[28] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal. Robust motion in-betweening. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 39(4):60:1–60:12, 2020. doi: 10.1145/3386569.3392480.

A Instantiation: observation, action, reward, and sim context

This appendix instantiates the shared spaces of the POMDP of §3.1. The motion-tracking interface follows the BeyondMimic formulation [3]; the changes we make are noted inline.

Observation. The proprioceptive observation $\boldsymbol{o}_t$ comprises the joint positions relative to the default pose $\boldsymbol{q} - \boldsymbol{q}^0$, the joint velocities $\dot{\boldsymbol{q}}$, the base angular velocity from the inertial measurement unit (IMU), the projected gravity in the base frame, and the previous action $\boldsymbol{a}_{t-1}$ (Table A.1); the policy receives the last 10 observations.

Observation Term	Dim.	Noise $\boldsymbol{\eta}_t$	Unit
Joint positions $\boldsymbol{q} - \boldsymbol{q}^0$	29	$\mathcal{U}[-0.01, 0.01]$	rad
Joint velocities $\dot{\boldsymbol{q}}$	29	$\mathcal{U}[-0.5, 0.5]$	rad/s
Base angular velocity (IMU)	3	$\mathcal{U}[-0.2, 0.2]$	rad/s
Projected gravity	3	$\mathcal{U}[-0.05, 0.05]$	–
Previous action $\boldsymbol{a}_{t-1}$	29	–	–

Table A.1: **Proprioceptive observation $\boldsymbol{o}_t$.** The noise column gives the per-dimension sensor noise $\boldsymbol{\eta}_t \sim \rho_\eta$.

Goal. For the motion-tracking interface, the goal $\boldsymbol{g}_t$ is read from the reference motion. Compared with BeyondMimic [3], we condition the encoder on a $K = 10$ future reference sequence rather than a single-frame target. Other goal interfaces replace $\boldsymbol{g}_t$ with their corresponding command representation, such as target velocity commands (Appendix B).

Action. The action $\boldsymbol{a}_t$ maps to a joint setpoint $\boldsymbol{q}^{\text{sp}} = \boldsymbol{\alpha}\, \boldsymbol{a}_t + \boldsymbol{q}^0$ about the default pose $\boldsymbol{q}^0$. Following BeyondMimic, the low PD gains make this setpoint a proxy for joint torque rather than a desired position.

Reward. We use BeyondMimic's motion-tracking reward (Table A.2), removing its self-collision penalty. That penalty guards against reward hacking by a stochastic policy. When the reference brings two bodies close without contact, pressing them together suppresses the action noise and raises the velocity-tracking reward. Our policy is deterministic in imagined rollouts (§3.3), so the exploit has no payoff and the term is unnecessary. The joint-limit penalty is lowered from -10 to -2.

Termination. An episode terminates when the root body deviates from the reference beyond a height or tilt threshold ($|\boldsymbol{e}_{p,z}| > 0.25\,\text{m}$ or $e_{\text{tilt}} > 0.8$). The tilt error e_{tilt} is the absolute difference between the z-components of the gravity direction expressed in the reference and robot root (pelvis) frames. For an upright reference, e_{tilt} reaches 0.8 at about 78° of tilt. Unlike BeyondMimic, we do not terminate on end-effector deviation.

Sim context and domain randomization. The per-episode sim context $\boldsymbol{\xi}$ collects the domain-randomization parameters fed to the world model (§3.2); Table A.3 lists them. We omit BeyondMimic's root-velocity perturbations: such pushes inject state discontinuities that the deterministic world model cannot capture, and we find randomization over the physical parameters alone suffices for transfer to hardware.

B Joint training across goal interfaces

In §3, the goal $\boldsymbol{g}_t$ is read from a full reference motion. Downstream goal interfaces specify the motion only in part or in another form. The same dynamics model can train several goal interfaces because the body physics is shared. We instantiate one POMDP $\mathcal{M}_i$ per goal interface (for example, a full-reference motion target, a velocity command, end-effector poses, or an SMPL body-pose target), each with its own goal space $\mathcal{G}_i$, reward r_i, and termination d_i, and train each with its own

Reward Term	Equation	Weight
Task (tracking)		
Body Position	$\exp\Big(-\big(\frac{1}{\lvert\mathcal{T}\rvert}\sum_{b\in\mathcal{T}}\lVert\mathbf{p}_b^{\text{des}}-\mathbf{p}_b\rVert^2\big)/0.3^2\Big)$	1.0
Body Orientation	$\exp\Big(-\big(\frac{1}{\lvert\mathcal{T}\rvert}\sum_{b\in\mathcal{T}}\lVert\log(R_b^{\text{des}}R_b^\top)\rVert^2\big)/0.4^2\Big)$	1.0
Body Linear velocity	$\exp\Big(-\big(\frac{1}{\lvert\mathcal{T}\rvert}\sum_{b\in\mathcal{T}}\lVert\mathbf{v}_b^{\text{des}}-\mathbf{v}_b\rVert^2\big)/1.0^2\Big)$	1.0
Body Angular velocity	$\exp\Big(-\big(\frac{1}{\lvert\mathcal{T}\rvert}\sum_{b\in\mathcal{T}}\lVert\boldsymbol{\omega}_b^{\text{des}}-\boldsymbol{\omega}_b\rVert^2\big)/3.14^2\Big)$	1.0
Root Position	$\exp\Big(-\lVert\mathbf{p}_{\text{root}}^{\text{des}}-\mathbf{p}_{\text{root}}\rVert^2/0.3^2\Big)$	0.5
Root Orientation	$\exp\Big(-\lVert\log(R_{\text{root}}^{\text{des}}R_{\text{root}}^\top)\rVert^2/0.4^2\Big)$	0.5
Regularization		
Action smoothness	$\lVert\mathbf{a}_t-\mathbf{a}_{t-1}\rVert^2$	-0.1
Joint limit	$\sum_j\left[\max(0,\,q_j-q_j^{\max})+\max(0,\,q_j^{\min}-q_j)\right]$	-2

Table A.2: **Motion-tracking interface reward, following BeyondMimic [3] with the self-collision penalty removed and the joint-limit weight lowered from -10 to -2 (§3.3).** $\mathcal{T}$ is the set of tracked bodies.

Domain Randomization	Sampling Distribution
Friction coefficients	$\mu\sim\mathcal{U}[0.3,\,1.2]$
Joint calibration offsets [rad]	$\Delta q_j^0\sim\mathcal{U}[-0.01,\,0.01]$
Torso's center-of-mass offset [m]	$\Delta x\sim\mathcal{U}[-0.025,0.025]$, $\Delta y\sim\mathcal{U}[-0.05,0.05]$, $\Delta z\sim\mathcal{U}[-0.05,0.05]$

Table A.3: **Domain-randomization parameters that make up the sim context $\boldsymbol{\xi}$, following BeyondMimic [3] with the root-velocity perturbations removed.** ($\mathcal{U}[\cdot]$: uniform distribution.)

critic V_{ψ_i}. A single policy network uses per-interface goal encoders $\text{enc}_{\theta,i}$ and a shared decoder dec_θ,

$$\boldsymbol{z}=\text{enc}_{\theta,i}(\boldsymbol{g})\in\mathcal{Z},\qquad \boldsymbol{a}=\text{dec}_\theta(\boldsymbol{z},\boldsymbol{o}),\qquad \boldsymbol{g}\in\mathcal{G}_i, \tag{A.1}$$

and training optimizes the lifted objective $\max_\theta\mathbb{E}_{i\sim p(\mathcal{I})}[\hat{J}_i(\theta)]$, where $p(\mathcal{I})$ is a distribution over the set $\mathcal{I}$ of goal interfaces. Each interface i thus has its own policy $\boldsymbol{a}=\pi_i(\boldsymbol{o},\boldsymbol{g}):=\text{dec}_\theta(\text{enc}_{\theta,i}(\boldsymbol{g}),\boldsymbol{o})$; the velocity encoder also receives a feature of the observation history (Table A.4). Each $\hat{J}_i$ is the objective of Eq. 2 with π_i in place of π_θ, the reward and termination of $\mathcal{M}_i$, and the critic V_{ψ_i}. Every interface trains through the same dynamics model f_ω, which is not retrained per interface.

A downstream interface is trained by distillation or by RL, depending on its reward. Distillation uses the motion-tracking reward (§3.3), with the command read from the reference motion. The reward gradient then trains that interface to reproduce the reference motion from this command. RL optimizes the interface's own task reward, such as velocity tracking, directly through the world model. On a downstream interface's rollouts, the return gradient passes through the shared decoder to that interface's encoder, with a stop-gradient on the decoder's weights; the decoder is still trained by the motion-tracking objective. The RL-trained velocity encoder of §4.5 also has a latent-alignment loss toward the motion encoder's latent on rollouts with reference commands (Appendix G).

C Network architectures

Table A.4 lists the world-model dynamics network of §3.2 and the policy and critic networks of Appendix B.

Network	Input	Output	Architecture
Dynamics f_ω	$\hat{\boldsymbol{s}}_t,\ \boldsymbol{a}_t,\ \boldsymbol{\xi}$	$\hat{\boldsymbol{s}}_{t+1}$	MLP (2048, 2048, 2048)
Motion encoder $\mathrm{enc}_{\theta,\mathrm{motion}}$	$K{=}10$ future reference frames	$\boldsymbol{z}$	MLP (2048, 1024, 512, 256)
Velocity encoder $\mathrm{enc}_{\theta,\mathrm{vel}}$ (Appendix G)	normalized command (v_x, v_y, ω_z), feature of the last 10 frames of $\boldsymbol{o}$	$\boldsymbol{z}$	MLP (256, 256)
Shared decoder dec_θ	$\boldsymbol{z}$, last 10 frames of $\boldsymbol{o}$	$\boldsymbol{a}_t$	MLP (2048, 1024, 512, 256)
Critic V_{ψ_i}	$\hat{\boldsymbol{s}}_t,\ \boldsymbol{g}_t,\ \boldsymbol{\xi}$	scalar	MLP (256, 128); one critic per interface

Table A.4: **World-model, policy, and critic networks.** MLP: multilayer perceptron, listed by hidden widths. The latent $\boldsymbol{z}$ is 32-dimensional and bounded by $\tanh$.

Algorithm A.1 SuperMimic training. Both ω and θ update every iteration; ω is held fixed (stop-gradient) only inside the policy's backward pass. For a downstream interface, $\hat{J}_i$ updates only its encoder $\mathrm{enc}_{\theta,i}$.

Require: world model f_ω; policy π_θ shared across interfaces $\{\mathcal{M}_i\}_{i\in\mathcal{I}}$; replay buffer $\mathcal{B}$
for each iteration **do**
  ▷ **Collect.** Roll out π_θ with action noise in simulation; push transitions to $\mathcal{B}$, replacing the oldest
    Draw reset start points by return-based sampling ▷ Appendix D.1
  ▷ **Fit world model.** Sample a logged k-step window from $\mathcal{B}$, roll f_ω out autoregressively on the logged actions, and step $\omega \leftarrow \omega - \alpha_\omega \nabla_\omega \mathcal{L}_{\mathrm{WM}}$ ▷ Eq. 1
  ▷ **Improve policy.** Sample an interface $i \sim p(\mathcal{I})$ and a start state from $\mathcal{B}$
    Imagine H steps on-policy through the fixed f_ω: $\hat{\boldsymbol{s}}_{t+1} = f_\omega\big(\hat{\boldsymbol{s}}_t,\ \pi_i(\hat{\tilde{\boldsymbol{o}}}_t + \boldsymbol{\eta}_t, \boldsymbol{g}_t),\ \boldsymbol{\xi}\big)$
    Step $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \hat{J}_i$ ▷ Eq. 2; analytic gradients through f_ω; critic V_{ψ_i}
    Fit the critic V_{ψ_i} to the GAE targets of the imagined rollout ▷ Eq. 3
end for

D Training loop and hyperparameters

D.1 Algorithm

Algorithm A.1 summarizes the SuperMimic training loop of §3.1; its three steps correspond one-to-one to the panels of Figure 2.

Return-based reset sampling. Simulator episodes reset to points of the motion library, favoring those where the policy currently earns low return. The motion timeline is divided into $1\,\mathrm{s}$ bins, and each bin b keeps a running estimate G_b of the discounted return from it, updated on the policy's own rollouts. Resets mix three distributions: 50% uniform over bins, 40% from $\mathrm{softmax}_b(-25\,G_b/G_{\max})$, where $G_{\max}$ is the largest attainable discounted return, and 10% in proportion to each bin's past softmax weight, so a hard motion keeps some practice after it is learned.

D.2 Hyperparameters

Table A.5 lists the hyperparameters. Where they differ, values are given for the one-GPU and the two-GPU run of Table A.8.

E Motion data filtering

We train on the publicly available G1-retargeted BONES-SEED motion set [27]. The raw release contains 142,220 clips, but many interact with terrain or objects the robot cannot use (stairs, boxes, ladders, chairs) or carry retargeting artifacts that make them infeasible for a free-standing humanoid on flat ground. We remove these in two passes and keep 129,514 clips (Table A.7).

Name filter. Each clip is named by its action. We drop any clip whose name contains a term denoting contact with terrain or an object the robot cannot use—e.g. `stair`, `ladder`, `box`, `chair`,

Hyperparameter	Symbol	Value
World model		
Window length	k	8
Optimizer		Adam
Learning rate	α_ω	1 GPU: 2×10^{-4}; 2 GPUs: $\sqrt{2} \times 2 \times 10^{-4}$
Batch size		10 batches of 2048 windows per update
Gradient clipping		global norm 1.0
Imagination and policy		
Imagination horizon	H	32
Optimizer		Adam, $\beta = (0.9,\ 0.95)$
Learning rate	α_θ	same as α_ω
Batch size		4096 imagined rollouts per update
Gradient clipping		global norm 1.0, then element-wise 0.1
Discount	γ	0.99
Sensor noise	$\boldsymbol{\eta}_t \sim \rho_\eta$	Table A.1
Critic		
GAE parameter	λ	0.95
Optimizer		Adam
Learning rate		same as α_ω
Gradient clipping		global norm 1.0
Critic epochs per update		5, each over 4 shuffled minibatches
Data collection		
Simulator environments per GPU		4096
Action noise	$\boldsymbol{\epsilon}_t \sim \rho_\epsilon$	$\mathcal{N}(0,\ 0.1^2)$ per action dimension
Replay-buffer size		1000 steps per environment
Schedule		
Update-to-data ratio		0.4
Policy updates per world-model update		1
Learning-rate schedule		cosine decay to $0.03\times$ over 250,000 iterations, then constant
Training length		320,000 iterations
Multi-GPU		data parallel; gradients averaged across GPUs

Table A.5: **SuperMimic hyperparameters.**

Reward Term	Equation	Weight
Task (velocity tracking)		
Linear velocity	$\exp\Big(-\lVert\mathbf{v}_{xy} - \mathbf{v}_{xy}^{\mathrm{cmd}}\rVert^2 / 0.5^2\Big)$	2.0
Yaw rate	$\exp\Big(-(\omega_z - \omega_z^{\mathrm{cmd}})^2 / 0.5\Big)$	2.0
Regularization		
Action smoothness	$\lVert\mathbf{a}_t - \mathbf{a}_{t-1}\rVert^2$	-0.3
Joint limit	$\sum_j \big[\max(0,\ q_j - q_j^{\max}) + \max(0,\ q_j^{\min} - q_j)\big]$	-2
Torso sideways lean	$\sin^2 \phi_{\mathrm{torso}}$	-4
Torso roll and pitch rate	$\omega_{\mathrm{roll}}^2 + \omega_{\mathrm{pitch}}^2$	-0.05
Termination (fall)		
$h_{\mathrm{root}} < 0.35\,\mathrm{m}$ or $1 + u_z > 0.7$		

Table A.6: **Velocity-tracking reward and termination used to train the RL encoder.** $\mathbf{v}_{xy}$ is the planar root velocity in the heading frame, ω_z the world yaw rate, and $(\mathbf{v}_{xy}^{\mathrm{cmd}}, \omega_z^{\mathrm{cmd}})$ the command. ϕ_{torso} is the sideways lean of the torso z-axis (forward lean is not penalized) and $\omega_{\mathrm{roll}}, \omega_{\mathrm{pitch}}$ the torso roll and pitch rates, all in the heading frame. h_{root} is the root height and u_z the z-component of the unit gravity direction in the root frame, so $1 + u_z$ is 0 upright and 2 upside down, and the tilt condition triggers at about 73°. Every term, here and in Table A.2, is multiplied by the control period $\Delta t = 0.02\,\mathrm{s}$.

Stage	Clips
Raw BONES-SEED (G1 retarget)	142,220
After name filter	129,667
After feasibility filter	129,514

Table A.7: **Number of BONES-SEED clips before filtering and after each filter.**

`sit`, `climb`, `jump_off_50cm`, `look_over_fence`. This pass (adapted from the keyword list in SONIC's released data-processing code [1], plus a few terms it misses) removes 12,553 clips.

Feasibility filter. The name filter misses clips whose names do not reveal the problem (e.g. a routine that walks onto a box). We run each motion through MuJoCo forward kinematics and flag it from the resulting body-link and foot-geometry trajectories, writing $z_{\min}(t) = \min_b z_b(t)$ for the height of the lowest body link at time t with the floor at $z = 0$:

- **Elevated support.** $z_{\min}$ holds above $0.20\,\mathrm{m}$ for at least $0.6\,\mathrm{s}$ (standing or climbing on a box).
- **Ground penetration.** $z_{\min}$ falls below $-0.12\,\mathrm{m}$, or below $-0.08\,\mathrm{m}$ for over 80% of the clip (legs driven through the floor by a faulty retarget).
- **Floating feet.** For at least $0.5\,\mathrm{s}$ *both* feet stay more than $0.02\,\mathrm{m}$ above the floor—measured as MuJoCo's exact geom-to-plane distance (`mj_geomDistance`) from the foot collision geometry, not a link-origin proxy—while the pelvis is upright and at standing height and the body is nearly stationary. A free-standing humanoid cannot hold both feet off the floor while standing still, so this is infeasible on flat ground whatever the cause (a low platform, a retarget float, or a slow rise; a ballistic jump is excluded by the stationary gate). The clean ground stance sits at $-0.004 \pm 0.004\,\mathrm{m}$, so the $0.02\,\mathrm{m}$ gate is several σ clear of the floor.

This pass removes a further 153 clips. All thresholds are tuned for the G1 retarget at $50\,\mathrm{Hz}$.

F Evaluation

F.1 Benchmark metrics

We evaluate SONIC and SuperMimic on the same filtered BONES-SEED training-set benchmark, containing 129,514 evaluated motions. For SONIC we evaluate its released checkpoint and its v1.1 checkpoint [1]. Each motion plays once from its first frame, with the same simulation for every policy: foot friction 1.0, no domain randomization or observation noise, and deterministic physics. Each policy uses its own actuator gains and action scale, and receives its observations as its training environment builds them. A motion is marked as failed if the rollout triggers SuperMimic's training termination (Appendix A). SONIC's own success criterion fails a motion on root and end-effector height deviations above $0.25\,\mathrm{m}$, without an orientation term [1].

F.2 Failed motions

Table A.8 counts failed motions per motion group. SONIC fails whole groups (all 44 flips, all 22 side rolls, and all 8 motions of one dance routine) and 15 of the 24 broad jumps. Its v1.1 checkpoint fails 239 motions, among them the same flips and dance motions and 19 of the 22 side rolls. After $3\,\mathrm{h}$ on one GPU, SuperMimic's failures concentrate in a few small groups of hard motions: the first five groups of Table A.8, 195 motions, hold 14 of its 25 failures. In no group does it fail more motions than SONIC. After $9.8\,\mathrm{h}$ on two GPUs, SuperMimic fails on none. Figure A.1 shows the count over training. The benchmark motions are training motions for both methods, so the counts measure how fully each policy covers these training motions, not generalization to new motions.

Motion group (motions)	SONIC	SuperMimic	
		1 GPU $\times$ 3 h	**2 GPUs** $\times$ 9.8 h
side rolls (22)	**22**	2	0
flips: 360°, 180°, 90° (44)	**44**	1	0
fire-in-the-hole turns (20)	3	3	0
crossed legs (10)	9	0	0
lying, resting and standing up (99)	28	8	0
broad jumps (24)	15	0	0
dance routine (8)	**8**	0	0
all other motions (129,287)	50	11	0
total (129,514)	179	25	0

Table A.8: **Failed motions per motion group on the filtered BONES-SEED training-set benchmark.** SONIC: released checkpoint. Bold: the policy fails every motion in the group.

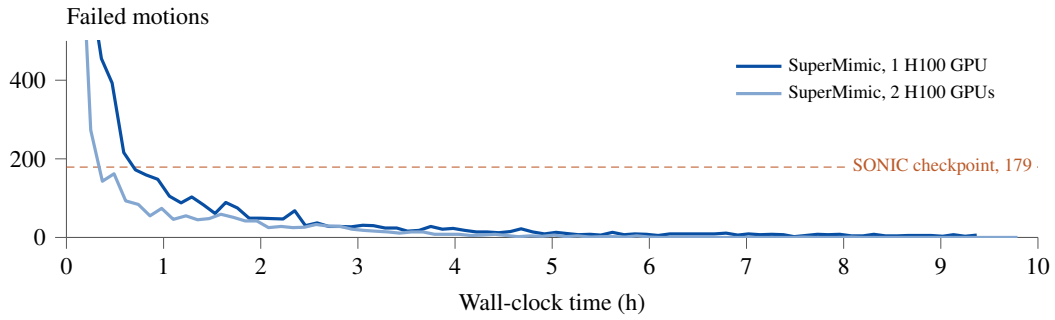

Figure A.1: **Failed motions during training on the filtered BONES-SEED training-set benchmark.** Checkpoints are scored every 4000 iterations, for the runs of Figure 3(a). SONIC: released checkpoint.

F.3 Experiment protocols

Figure 3. SuperMimic trains on the filtered BONES-SEED set (Appendix E). The dataset-length sweep in (b, c) uses subsets of 1, 7, 40, and 264 h.

G Velocity-command task

G.1 Training setup

The velocity encoder of §4.5 maps the command (v_x, v_y, ω_z) and a feature of the last 10 frames of $\boldsymbol{o}$ to the shared latent.

Each encoder is trained from scratch in its own run, jointly with the motion-tracking interface, for 100,000 iterations. Both runs use a walking and running library filtered from LAFAN1 and BONES-SEED. Half of the 4096 simulator environments and half of each imagined batch use the velocity interface, so $p(\mathcal{I})$ is uniform; unlike Algorithm A.1, which samples one interface per update, each imagined batch mixes both.

Commands. The command $\boldsymbol{g}_t = (v_x, v_y, \omega_z)$ stacks the target planar root velocity in the heading (yaw-only) frame and the target world-frame yaw rate. A reference command is the centered 25-frame mean of the reference root velocity in the reference's heading frame. For the RL encoder, each environment resamples its command source at every reset and every 250 control steps (5 s): with probability 0.5 it reads the command from the reference clip at every step, and otherwise it holds a command drawn uniformly from $v_x \in [-1.5, 2]\,\mathrm{m/s}$, $v_y \in [-1.32, 1.32]\,\mathrm{m/s}$ and $\omega_z \in [-3.35, 3.35]\,\mathrm{rad/s}$, zeroed (standing) with probability 0.1.

Encoder. On rollouts with reference commands, a mean-squared loss of weight 0.1 pulls the velocity latent toward the motion encoder's latent at the same step, with a stop-gradient on the motion latent.

Rollouts, reward, and termination. Each interface's imagined rollouts start from states logged by its own simulator environments. Velocity rollouts of the RL encoder use the velocity-tracking reward and a fall-only termination (Table A.6); those of the distillation baseline use the motion-tracking reward and termination of Appendix A.

G.2 Step test

Protocol. From standing, the policy receives a fixed command for $5\,\mathrm{s}$; each command is tested in four trials. The relative steady-state error is the norm of the mean velocity error over the last $2\,\mathrm{s}$, divided by the norm of the command, with the root velocity measured in the heading frame. The per-command value is the median over the trials that did not fall. The 672 step responses form a 15×15 grid on each of three planes, spanning the uniform command range above, with the standing command left out. In Figure 7, errors above 0.9 are drawn at 0.9, and a policy that fell in all four trials is drawn at the cap.

Results. The RL encoder has the lower error on 650 of the 672 step responses, and the distillation encoder on 15. Their median relative errors are 0.16 and 0.49. Distillation falls in every trial on seven commands, all combining translation with a turn of at least $2.8\,\mathrm{rad/s}$, where the motion library has almost no data. Each policy has one training seed.