CROSSBFM: DISTILLING A SHARED LATENT BEHAVIOR SPACE ACROSS HUMANOID EMBODIMENTS

Anonymous authors
Paper under double-blind review

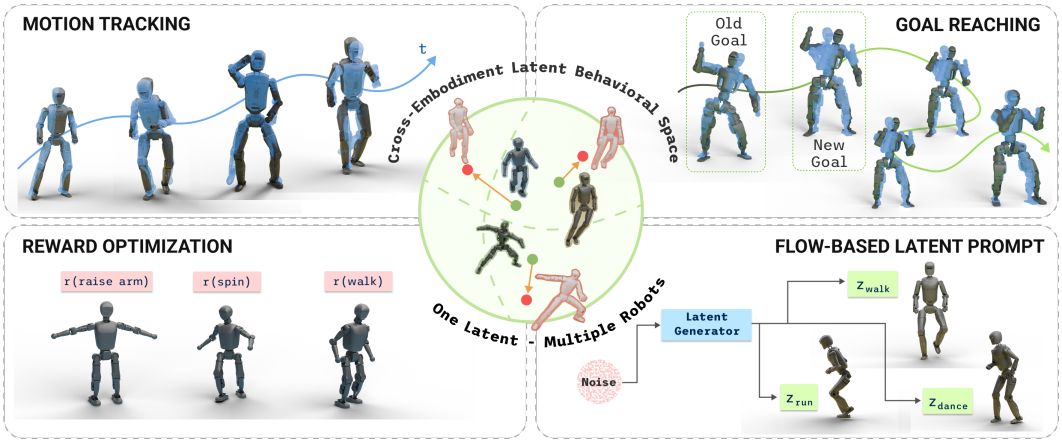


Figure 1: CROSSBFM leverages a unified encoder to distill a frozen source BFM latent onto new robots. The distilled latent can process various prompts, including the three original modes – motion tracking, goal reaching, and reward optimization – and also a new mode for flow-based generated latents.

ABSTRACT

Behavior Foundation Models (BFMs) give humanoids a promptable policy over a latent behavior space, enabling one single vector to represent a motion to imitate, a pose to reach, or a reward to maximize. Forward-Backward representations successfully produce such spaces, but at the cost of hundreds of GPU-hours for a single robot. Moreover, when the training process is repeated for a second robot, it produces a second space unrelated to the first, resulting in embodiment-specific latents that do not unify or transfer. We address these problems with CROSSBFM, treating the latent space as the transferable asset for various embodiments. As retargeting provides frame-level cross-embodiment correspondence, we propose a unified encoder architecture with no robot-specific parameters for distilling the behavior space to address all training embodiments simultaneously in less than a GPU-hour. Following this encoder, latent-conditioned trackers turn the distilled latent into whole-body control in a conventional PPO training manner in just 10 more GPU-hours. On three distilled humanoids, all three prompting modes transfer: motion tracking with latent-conditioned policy losing only 0.025 rad to its joint-conditioned counterpart, smooth goal reaching between poses with no falls, and reward optimization for all 41 reward prompts. Our experiments further reveal that 1) regressing the encoder on a quarter of the motion corpus costs only 5% of tracking performance and 2) training the encoder on a subset of robots and evaluating on an unseen one recovers up to 89% of the tracking performance of seen robots, demonstrating cross-embodiment generalization to morphologically similar robots. We also verify the pipeline on real robots across all three prompting modes and with flow-based generated latents. Project website: `https://crossbfm.github.io/`

1 INTRODUCTION

Humanoid whole-body control via Reinforcement Learning (RL) policies has witnessed remarkable progress with robots performing a wide range of motions, ranging from dancing or parkour to full

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

loco-manipulation in challenging settings (Yang et al., 2026; Liao et al., 2026; Wu et al., 2026; Luo et al., 2026). Commanding these policies typically requires a sequence of actions in joint space that were retargeted specifically for the robot to mimic. Besides joint-conditioned control, whole-body control, through Forward–Backward (FB) representations (Touati & Ollivier, 2021; Touati et al., 2023; Tirinzoni et al., 2025), established a compelling command interface in the latent space for humanoid robots via motion latent features. In this paradigm, a policy $\pi(\cdot \mid z)$, conditioned on a latent vector z, that is drawn from a trained behavioral space $\mathcal{Z} \subseteq \mathbb{R}^d$, can be prompted at test time to track a motion, reach a goal pose, or maximize a commanded reward, all without any retraining. BFM-Zero (Li et al., 2025) brought this idea from simulated characters (Tirinzoni et al., 2025) onto real robots, followed by frameworks such as UFO (RoboParty Lab Team, 2026), which further improve the training pipeline. Nevertheless, one training run still costs more than 100 GPU-hours for any robot.

While the pretrained BFM can handle various prompt types, it is always trained for one specific robot. For a new robot, the training has to run from scratch, which in turn produces a new latent space that does not related to one from a previous robot. Although both latent spaces are geometrically similar, the coordinates at a given point correspond to different behaviors. Since every downstream task depends on this latent behavior space, the resulting cross-embodiment transfer becomes as problematic as the doubled training cost. Previous cross-embodiment learning works introduce alternative approaches to expand BFM without retraining. Some (Bohlinger et al., 2024; Patel & Song, 2025; Ai et al., 2025) propose conditioning one policy on an embodiment description and training with multiple embodiments simultaneously, which generalizes to unseen embodiments. While these methods demonstrated cross-embodiment transfer, they are typically constrained to a single task (e.g., velocity-tracking locomotion or in-hand rotation) with their shared representation encoding the morphology rather than task-conditioned behavior. These frameworks lack representations encoding "the reward I want maximized" or "the pose I want reached" compared to BFM, which directly encodes the action value for each latent coordinate of the learned space (*value-functional*).

Our approach. We propose CROSSBFM (Figure 2) to tackle both the repeated training and latent space mismatch problems. First, we *freeze* a BFM pretrained for a specific robot, viewing its FB latent as a fixed *behavioral coordinate system*, and distill that coordinate system onto new embodiments. By leveraging retargeted motions for different robots along the same timeline, we can directly establish cross-embodiment correspondences for all the training robots, which reduces learning the backward map for the target robot to supervised regression. After this correspondence is established, we introduce a unified encoder to map all robot configurations into a shared latent space in only one training run. Our encoder is a fixed-width view of the robot proprioception with 33 joints and 8 key-body indices, into which any robot's degrees of freedom are placed. Indices that do not exist for a specific robot are masked out, such that the input dimensionality is identical across embodiments and the architecture becomes universal for a wide range of humanoids. Finally, we train *latent-conditioned trackers* that receive latents from the shared space, enabling whole-body control across various embodiments. While the source model spent on the order of 10^2 GPU-hours of online unsupervised RL — with a replay buffer, a discriminator, and domain randomization — to obtain a backward map for one robot, our full pipeline requires just around 10 hours on an RTX4090.

Contributions.

- **A unified encoder for BFM distillation.** We introduce a robot-independent encoder architecture allowing for multi-robot distillation in one training at a comparable accuracy of separate per-robot encoders, while producing more consistent latents across robots and generalizing to unseen embodiments of a similar morphology. (Section 4.2).
- **Retargeting as a correspondence oracle.** We show how latent transfer across humanoid embodiments is equivalent to supervised regression from a frozen source BFM latent, thanks to frame-level correspondence across retargeted datasets. This removes the simulator, RL training, and adversarial objective from the target robot side (Section 4.3).
- **Latent-condition whole-body control.** We evaluate our method on three distinct humanoids with trackers conditioned on the distilled latent $[z \mid \text{proprio}]$, achieving comparable results with their joint-conditioned counterparts while covering all three prompting modes of BFM-Zero. We extend the latent space by introducing a flow-based latent generator that takes in behavior modes and generates the corresponding latents. We also verify our pipeline on real robots, demonstrating its transfer to real hardware (Section 5).

2 Related Work

Behavior foundation models and unsupervised RL for humanoid control. Most humanoid whole-body controllers are trained as motion *trackers* with an on-policy RL algorithm such as PPO (Schulman et al., 2017) optimizing an explicit imitation reward for a retargeted reference trajectory (Luo et al., 2023; Cheng et al., 2024; He et al., 2024; Ze et al., 2025). This family of policies requires a per-frame reference in joint space, which is challenging to obtain for complex tasks without a high-level planner. Character animation instead learns a *latent* skill space from unlabeled motion and conditions a policy on the trained latent space (Peng et al., 2022; Tessler et al., 2023), gaining reusability at the price of a latent whose semantics are only implicitly defined. Unsupervised RL (Gregor et al., 2016; Eysenbach et al., 2019; Pathak et al., 2017) and in particular the Forward–Backward family (Touati & Ollivier, 2021; Touati et al., 2023) give the latent an explicit meaning as a task descriptor. In this setting, a reward, a goal, or a demonstration map to a vector in the latent space that the downstream policy conditions on. Following this line of work, FB-CPR (Tirinzoni et al., 2025) made this practical for high-dimensional humanoids by regularizing the unsupervised policy toward an unlabeled motion dataset with a latent-conditional discriminator, while BFM-Zero (Li et al., 2025) extends this further to a real robot through domain randomization and safety-oriented reward shaping. UFO (RoboParty Lab Team, 2026) rebuilt the infrastructure for speed and generality, cutting FB pre-training by five times to over 100 hours on a consumer GPU and showing that other unsupervised objectives, e.g., temporal-distance representations (Bae et al., 2024), can enhance the latent consistency. All of these works produce a behavior space *per robot*, i.e., unrelated to other robots. CrossBFM directly wires the pretrained space of one robot to other embodiments without rerunning the costly FB training.

Cross-embodiment learning. Training a single policy across many robots requires architectures and training paradigms that can either condition on or abstract over embodiment differences. Previous works propose Graph Neural Networks (GNNs) to directly use the kinematic structure of robots as part of the network (Wang et al., 2018; Huang et al., 2020), and more recent works use attention-based architectures that leverage body parts as tokens (Gupta et al., 2022; Sferrazza et al., 2025; Patel & Song, 2025; Ai et al., 2025; Bohlinger & Peters, 2026) or infer the embodiment from long interaction histories (Liu et al., 2025; Li et al., 2026). These approaches achieve generalization to unseen embodiments but focus on a single task, with the shared component being the embodiment encoder rather than the behavior space. A second family directly establishes correspondence without policy conditioning, aligning state spaces of two policies with optimal transport (Fickinger et al., 2022) or graph matching (Le et al., 2025). This approach is often more expensive, while CrossBFM direcly leverages retargeted datasets as cross-embodiment correspondences. A third and more recent family learns unified cross-embodiment latent spaces (Yan & Lee, 2026; Kim et al., 2026; Chen et al., 2026; Zhi et al., 2026). These are closest to our work by intuition, but their latents are learned from scratch and are descriptive rather than *value-functional*, thus, they do not come with a closed form that directly turns a reward into a latent. CrossBFM differs on both aspects. Our latent space is not learned but *inherited* from a frozen FB model, and it therefore retains the FB prompting semantics on the new embodiments it is distilled onto.

Retargeting and distillation. Motion retargeting maps a human mocap trajectory onto a robot's kinematics conditioned on embodiment-specific characteristics such as joint limits or key body constraints. Modern retargeting pipelines are accurate and fast enough to be run both offline over large dataset as well as responsive enough for real-time teleoperation (Araújo et al., 2025; Ze et al., 2025; Yang et al., 2026). In this work, we leverage retargeted data as ground-truth correspondence for cross-embodiment transfer. We formulate our training objective to be a form of knowledge distillation (Hinton et al., 2015) with the teacher (source BFM) and the students (new robots) observing *embodiment-specific* retargeted motions aligned to each other by the shared timeline.

3 Preliminaries

3.1 Unsupervised RL and successor measures

We consider a reward-free discounted Markov decision process $\mathcal{M} = (S, A, P, \mu, \gamma)$, with state space S, action space A, transition kernel $P(\mathrm{d}s' \mid s, a)$ taking over all possible states s' over an infinitesimal region around it ds', initial-state distribution μ and discount $\gamma \in (0, 1)$. Because no reward is given at training time, the object an unsupervised RL agent can learn is the *dynamics* of its own policies. For a policy π, the *successor measure* (Dayan, 1993; Blier et al., 2021) records where

that policy goes after infinite steps t,

$$M^{\pi}(X \mid s,a) \;:=\; \sum_{t\geq 0} \gamma^t \,\Pr\big(s_{t+1}\in X \,\big|\, s,a,\pi\big), \qquad X\subseteq S. \tag{1}$$

This representation is particularly useful as it factorizes the value function into successor probability multiplied by the reward gained at that state. Specifically, for *any* reward $r : S \to \mathbb{R}$,

$$Q^{\pi}_{r}(s,a) \;=\; \int_{S} M^{\pi}(\mathrm{d}s' \mid s,a)\, r(s'). \tag{2}$$

Equation 2 separates "how the policy moves" from "how much reward the policy gains", which can be used to evaluate any reward at test time without any reward-driven finetuning.

3.2 Forward–Backward representations

FB representations (Touati & Ollivier, 2021; Touati et al., 2023) realize the unsupervised RL objective by taking a finite-rank approximation of the successor measure. Given a state distribution ρ, one learns a *forward* map $F : S \times A \times \mathcal{Z} \to \mathbb{R}^d$ and a *backward* map $B : S \to \mathbb{R}^d$, along with a latent-conditioned policy π_z, such that

$$M^{\pi_z}(\mathrm{d}s' \mid s,a) \;\simeq\; F(s,a,z)^{\top} B(s')\,\rho(\mathrm{d}s'), \qquad \pi_z(s) \;=\; \arg\max_{a} F(s,a,z)^{\top} z, \tag{3}$$

where $\mathcal{Z} \subseteq \mathbb{R}^d$ is conventionally the sphere of radius $\sqrt{d}$, and F, B are trained to minimize the temporal-difference residual of the measure-valued Bellman equation (Touati & Ollivier, 2021; Tirinzoni et al., 2025). Substituting Equation 3 into Equation 2 gives the closed-form of the latent inference for any reward r,

$$Q^{\pi_{z_r}}_{r}(s,a) \;=\; F(s,a,z_r)^{\top} z_r, \qquad z_r \;=\; \mathbb{E}_{s\sim\rho}\big[\, B(s)\, r(s)\,\big]. \tag{4}$$

Consequently, the backward map B converts a reward function into the latent whose policy maximizes it in *closed form*.

3.3 The three prompting modes

With F, B and π_z trained by the objectives in Tirinzoni et al. (2025), a BFM can answer three kinds of prompts at test time without any retraining or planning:

- **Motion tracking**: given a reference motion $\tau = (s_1, \ldots, s_n)$, the latent at time t is a look-ahead embedding of the reference given by $z_t = \mathrm{proj}_{\mathcal{Z}}\big(\sum_{t'=t}^{t+H} B(s_{t'})\big)$.
- **Goal reaching**: given a target state s_g, then $z_g = \mathrm{proj}_{\mathcal{Z}}\big(B(s_g)\big)$.
- **Reward optimization**: given samples $\{(s_i, r_i)\}_{i=1}^{M}$ with $s_i \sim \rho$, use the empirical form of Equation 4, $z_{\mathrm{rew}} = \mathrm{proj}_{\mathcal{Z}}\big(\sum_i \omega_i\, r_i\, B(s_i)\big)$.

The backward map B enables the smooth conversion from desired behaviors to corresponding latents, which then drive the policy. This is the structural reason why our method focuses on the backward map for direct latent distillation rather than both the forward and backward maps. If we can, for a new embodiment, produce latents in the frozen source's coordinate system, we directly inherit all three prompting modes at once via the closed-form computation of B.

4 CrossBFM

4.1 Setup and notation

We freeze a source BFM trained on the Unitree G1 (Li et al., 2025) and focus only on its backward map B_S. We denote $d = 256$ and $\mathcal{Z}$ is the sphere of radius $\sqrt{d}$, where $\mathrm{proj}_{\mathcal{Z}}$ is the radial projection onto it. For every frame t of a reference motion, we compute the source latent

$$z^{\star}_{t} \;=\; \mathrm{proj}_{\mathcal{Z}}\big(B_S(o^{G1}_{t})\big), \tag{5}$$

which is the ground-truth latent label retained from the source model. Each clip is retargeted independently onto every robot X on the *same* timeline with the same frequency, so the target frame o^{X}_{t} and the source frame o^{G1}_{t} correspond to the same motion of the same behavior. We then attach $z^{\star}_{t}$ to their corresponding retargeted data as ground-truth latent. The correspondence that cross-domain imitation normally has to learn is thereby handled by the retargeting algorithm. Consequently, the target-side learning problem becomes the regression $E_X : o^{X}_{1:N} \mapsto \hat{z}_{1:N} \in \mathcal{Z}$ that maps a *new* robot's proprioception onto a latent that a *source* robot produces.

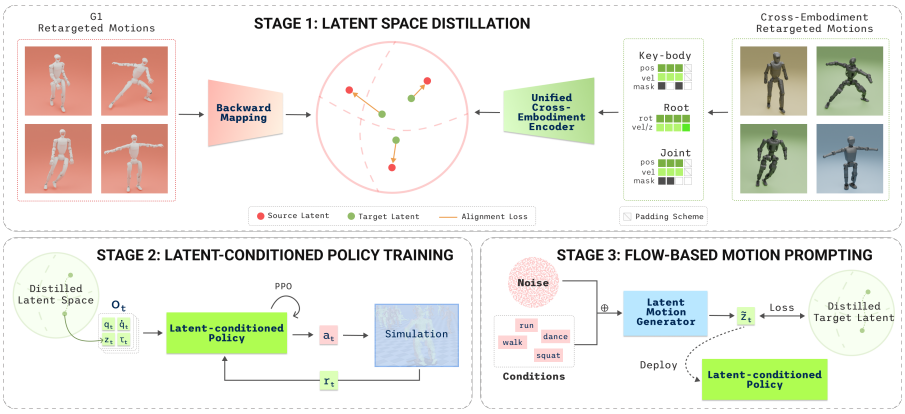


Figure 2: CROSSBFM overview. *Top:* a unified encoder mapping all new robots' proprioception onto the frozen source latent using frame-level correspondences from retargeted data. *Bottom:* a z-conditioned policy and a flow-based latent generator on top of the distilled latent space.

4.2 THE UNIFIED ENCODER ARCHITECTURE

Different humanoids may vary in the number of actuated joints, the joint order, and their body composition. A per-robot encoder absorbs these differences explicitly by having a different input configuration, but then requires multiple networks to be trained without a shared representation. This design also hinders generalization to new robots with different morphologies. We instead define a fixed-width, robot-independent input, covering joints, key bodies, and root configuration, then map each robot's retargeted data into this unified input for the encoder.

Our encoder focuses on two main morphological components. First, *8 key-body indices*, comprising $\{\text{left}, \text{right}\} \times \{\text{knee}, \text{foot}, \text{elbow}, \text{arm tip}\}$. Here, the root is deliberately excluded from the key-body set, since key-body positions are expressed relative to it thus the root row would be simply zero. Second, *33 canonical joint indices* include 12 leg, 3 waist (yaw / roll / pitch), 16 arm and 2 head joints. This vocabulary intentionally contains indices that *no* robot in our experiments has (such as waist roll and pitch or head joints), such that the encoder can be universally applied for a wide range of robots. Missing indices are *masked* with a binary flag and zero-padded, such that the network can tell "this joint is at zero" from "this joint does not exist." We additionally include a *robot pelvis proprioception* that carries the root's global height, orientation and linear velocity, which serves as the global anchor for other key bodies. More details are provided in Appendix A.2.

4.3 STAGE 1: BACKWARD MAP REGRESSION

The unified encoder takes as input a short window of length T of the proprioception in a reference motion of one robot and returns a latent for every frame in it. First, we use a linear layer to map each frame to the encoder embedding width H, followed by a positional encoding to embed the frame order. We add pre-LayerNorm transformer blocks (Vaswani et al., 2017; Xiong et al., 2020) to allow frames to attend to one another, which improves the temporal consistency and prevents mode collapse. Finally, we use a linear layer to turn each frame into the shape of the frozen BFM latent, rescaled to the sphere of radius $\sqrt{d}$ (more in Appendix A.3). The latent projection is given by:

$$\hat{z}_t \;=\; \mathrm{proj}_{\mathcal{Z}}\big(E_X(o^X_{t-T+1:t})\big), \qquad \mathrm{proj}_{\mathcal{Z}}(v) \;=\; \sqrt{d}\, v / \max(\|v\|, \varepsilon). \tag{6}$$

We introduce *causal* attention to our encoder that applies an upper-triangular attention mask so that frame t attends only to frames $\leq t$. While the bidirectional version sees roughly one second of the *future reference* and remains acceptable for offline motion processing, it is unsuitable for real-world control. The objective is a frame-wise cosine regression onto the frozen source latent:

$$\mathcal{L}_{\text{align}} \;=\; \frac{1}{N} \sum_t \Big(1 - \cos\big(\hat{z}_t, z^\star_t\big)\Big). \tag{7}$$

The cosine distance serves as a proximity measure for the cross-embodiment representation on the latent space. With this simple architecture and objective, the encoder learns to map the target robot's proprioception to the source latent space in just under an hour on a single consumer GPU, against the $\sim 10^2$ GPU-hours that are needed to train B_S. Our encoder has no robot-specific weights and

each robot configuration enters only through the fixed index maps in Section 4.2. Appendix C.2 compares our unified encoder architecture against three separate per-robot encoders.

4.4 Stage 2: latent-conditioned policies

With the distilled latent space, we then train a tracking policy π_X for each robot with PPO (Schulman et al., 2017). These trackers are latent-conditioned, meaning that they only take as input the behavior latent and proprioception. The proprioception includes base angular velocity, IMU roll and pitch, joint positions and velocities, and the previous action. In this setting, there is no explicit reference trajectory in the actor's observation apart from the latent behavior. We keep the critic asymmetric with the privileged reference observations available in simulation. The privileged critic keeps the value estimation accurate without leaking unavailable information into the policy. Because the latent z is shared across embodiments, we can leverage a generative model to approximate this space to command *every* robot. In this work, we use a single rectified-flow model (Liu et al., 2023; Lipman et al., 2023) trained over latent chunks, so that a behavior can be given by a wider range of prompts (text, for example), rather than restricted to robot-specific joint-space references. More about the latent generator in Appendix D.

4.5 Deployment: prompting without a target-side FB model

Two of the three prompting modes of Section 3.3 immediately transfer after the encoder training. Tracking reference is $\hat{z}_t = E_X(o^X_{t-T+1:t})$ on a retargeted reference while goal reaching's is E_X on a predefined window ending at the goal pose. However, reward optimization requires post-processing on the target side, as Equation 4 requires two components from the frozen source BFM. While the backward map B_S can be directly substituted with the trained encoder E_X, the distribution ρ of visited states collected during RL training, over which the expectation is taken, is unavailable on the target robot, which never ran unsupervised RL. To address this, we approximate the distribution ρ for the distilled robots using their retargeted datasets. This ensures the reward-weighted projection selects states feasible for the target robot, rather than relying on the source distribution, which may include poses unreachable on the target side. For the reward function, to address the variation in joint configurations, we keep only the key-body kinematics (root height and velocity, uprightness, angular velocity, arm-tip and knee heights) as reward features and discard terms on the joint configuration. Specifically, we first leverage the trained encoder to convert proprioception into latents and compute the reward features for each pose in the cached dataset over selected features. We then compute a reward-weighted average over all cached latents as the final reference latent for the policy to track:

$$\underbrace{Z_i = E_X\big(o^X_{i-T+1:i}\big)}_{\text{encode once}}, \qquad \underbrace{r_i = r\big(\phi(o^X_i); \theta^{(X)}\big)}_{\text{score cached features}}, \qquad \underbrace{z_{\text{rew}} = \text{proj}_{\mathcal{Z}}\Big(\textstyle\sum_i \omega_i\, r_i\, Z_i\Big)}_{\text{project}}, \tag{8}$$

with $\omega = \text{softmax}(\tau\, r_{1:M})$, M cached frames of the retargeted dataset, and a temperature τ. Another problem remains, which is that the source-robot reward often refers to absolute configurations, such as a specific hand height or reach distance, that a differently proportioned robot cannot attain. We propose leveraging each robot's keybody distribution over the retargeted dataset so that keybody positions for different robots fall in similar pose percentiles within the shared dataset. For example, the "raise arm" task of a medium robot remarked by $1.0\,\text{m}$ hand height would lie in the p percentile of its feature distribution. With the p percentile as an anchor for the other robots' feature distribution, it may convert to $0.820\,\text{m}$ on a smaller robot and $1.157\,\text{m}$ on a larger robot. Given Φ_S, Φ_r being the CDFs of key-body features for source and target robots, respectively, the transport is therefore specified by:

$$\theta^{(r)} \;=\; \Phi_r^{-1}\big(\Phi_S(\theta)\big). \tag{9}$$

With this alignment, we can analytically compute the reward formulation for all new robots simultaneously without having to hand-craft thresholds for each robot. Details in Appendix B.3.

5 Experiments

Setup. We distill a frozen BFM pretrained on the Unitree G1 robot (Li et al., 2025) onto three target robots (Inhouse M3, Booster T1, Fourier N1) that differ in their number of joints, links, and topology. Following the pretrained BFM, we use the LAFAN dataset (Harvey et al., 2020) for our experiments. More about the task settings is in Appendix B. We ask 3 questions:

Table 1: Latent-conditioned vs. joint-conditioned tracker by joint mean absolute error.

Robot	DoF / bodies	Tracking MAE (rad) ↓			Goal MAE (rad) ↓	
		Joint (TWIST2)	Latent (ours)	Δ_{cond}	Latent (ours)	Δ_{goal}
M3	27/30	0.1777 ± 0.0033	0.2024 ± 0.0067	$+0.0247$	0.2345 ± 0.0011	$+0.0321$
T1	23/24	0.1844 ± 0.0043	0.1901 ± 0.0032	$+0.0057$	0.1967 ± 0.0012	$+0.0066$
N1	23/29	0.1362 ± 0.0032	0.1391 ± 0.0038	$+0.0029$	0.1512 ± 0.0023	$+0.0121$

- How well do the three original prompting modes transfer to distilled robots in the training set, both in simulation and in the real world?
- Can the unified encoder generalize to unseen motion within the trained latent space, both functionally for motion tracking and geometrically for representation reconstruction? Since the source space is well-structured, how much data do we need to distill this space?
- Can the encoder generalize to unseen robots?

5.1 Transferring three original prompting modes

Tracking and Reaching. We compare latent-conditioned policies against joint-conditioned counterparts trained by TWIST2 (Ze et al., 2025) under the same training settings (Table 1). The joint-conditioned policies are naturally the upper bound for the tracking task as at every control step, they have access to exactly which joint angles to track. In contrast, the latent-conditioned policies are only given a single latent corresponding to the desired behavior and must infer the pose from it.

Nevertheless, across all three robots, the gap (latent $-$ joint) stays within 0.025 rad and the latent-conditioned policies successfully follow the reference motions. Replacing a dense per-frame reference with one latent vector therefore marginally hurts the performance while allowing for smooth transition between poses, the property that the joint-conditioned policies do not have. In fact, for the joint-conditioned policies, commanding a discontinuous jump in joint targets produces large torques and early termination (RoboParty Lab Team, 2026). For this reason, goal reaching is only reported for the latent-conditioned policies, where interpolating in $\mathcal{Z}$ results in smooth transitions at a cost of at most 0.032 rad compared to the tracking performance. The latent-conditioned policies also express prompted behaviors generated by the flow-based latent generator within the distilled space, including walking, running, and dancing (see our website). Together, this suggests that a sufficiently fine-grained latent space with more complex primitive behaviors could let an operator command a humanoid through *behavioral intent* rather than joint targets, which is a substantially more natural interface for complex long-horizon loco-manipulation.

Reward optimization. Table 2 compares the latent produced by using the state buffer collected during RL training with the true backward map B_S of the source model versus the latent produced by the cached retargeted dataset with our percentile reward threshold using the feature transformation. Computing z_{rew} on the source robot with the true backward map B_S is $3.5\times$ and $3.0\times$ worse on the M3 and N1 compared to ours. The reason is that the reward-weighted projection selects *which states are worth visiting*, which depends on which states the executing robot can actually reach. Thus, a latent, which is optimal for the G1's reachable set, is not optimal for other robots. Together with tracking and goal reaching, this covers all three BFM prompting modes (Touati et al., 2023; Tirinzoni et al., 2025) on new robots, demonstrating *functional* property for a shared behavior space.

Table 2: Reward optimization prompt results by normalized returns. *Source-side* computes z_{rew} on the G1 with the true backward map B_S; *target-side (ours)* computes it via Equation 8.

Robot	Source	Target (ours)	Gain
M3	0.145	**0.508**	$3.50\times$
T1	0.228	**0.261**	$1.14\times$
N1	0.177	**0.523**	$2.95\times$
Mean	0.183	**0.431**	$2.35\times$

5.2 Latent inference on unseen motions

We evaluate how well the unified encoder infers latent behaviors across the training and validation motions, both functionally for the tracking task and geometrically for latent alignment.

Table 3: Closed-loop key-body error `mpjpe_local`↓ ($\times 10^{-2}$ m) for the tracking task using the oracle latent $z^\star$ versus the encoder's inference, split by training/validation set.

Encoder split	M3		T1		N1	
	oracle	infer.	oracle	infer.	oracle	infer.
Training (30)	8.14	8.02	5.70	5.43	5.58	5.57
Validation (10)	6.40	7.07	5.26	5.60	5.46	5.60

Functionality. Table 3 demonstrates the performance of latent trackers conditioned on oracle and inferred latents in. On the encoder's

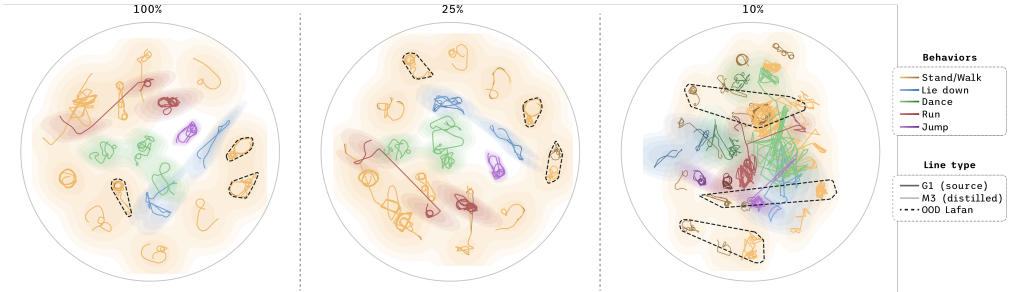


Figure 3: t-SNE projection of source (G1, dark) and distilled (M3, bright) latent trajectories, colored by behavior type. Dashed outlines represent validation clips (*unseen* clips of similar behaviors).

Table 4: Generalization to unseen robots. *2-R E.* represents the encoder trained on two robots and evaluated on the third, with fixed pretrained trackers. *3-R E.* is the unified encoders trained on all 3 robots.

Unseen robot	Validation cosine ↑		Closed-loop joint MAE (rad) ↓			Signal gap closed ↑
	2-R E.	3-R E.	2-R E.	3-R E.	random z	
M3	0.6251 ± 0.0196	0.8658	0.1829 ± 0.0054	0.1645	0.3371	$\mathbf{89.3\% \pm 3.5}$
N1	0.5861 ± 0.0100	0.8640	0.2237 ± 0.0048	0.1370	0.3320	$\mathbf{55.5\% \pm 3.2}$
T1	0.2533 ± 0.0168	0.8300	0.2886 ± 0.0138	0.1679	0.3063	$\mathbf{12.8\% \pm 10.3}$

training clips, the predicted latent is marginally better than the oracle latent $z^\star$. On validation motions, the latent costs 1.4–6.7 mm of key-body error while maintaining good balance for the whole trajectories. Since all eight behaviors of the LAFAN dataset appear in the training set, we conclude that the encoder is able to generalize to new *clips* of similar behaviors.

Geometrical alignment. Figure 3a projects source and distilled latent trajectories into 2D, showing how distilled trajectories *overlay* the source trajectories *almost identically*. This means that our encoder enforces latents to live in the source's coordinate system while forming clear behavioral clusters. The validation clips (dashed) also fall clearly *inside* the cluster of their own behavior type rather than beside the manifold or in the gaps between the clusters. For example, an unseen walking clip is not mapped somewhere new but rather into the region of $\mathcal{Z}$, that the walking cluster already occupies. As a result, the tracker trained on that region can adapt to validation latents well (Table 3).

5.3 Data scaling: a quarter of the corpus suffices

We shrink the distillation dataset up to 95% by randomly selecting $K = \sum_1^{40} K_i$ non-overlapping segments of 1.5–3 s for all 40 motions. The total length of these K segments adds up to $T_K \in (5, 75)$ percent of the LAFAN dataset. We choose continuous segments rather than isolated frames, such that local temporal structure is preserved and all the 40 behaviors remain available. These segments are then used to train unified encoders with the shrunk dataset. After training, we feed the latent inferred by these encoders to a tracker trained on 100% of the data to see how well the shrunk encoders can reconstruct *behaviors*. Figure 4 shows how the joint MAE only rises from 0.2132, at 100% of the corpus to 0.2241, at 25%, demonstrating a 5% degradation for four times less data. The tracking performance only starts to break down at 10% (0.2706) against a random-latent control at 0.4411. The geometry degrades gradually, similar to the tracking performance. From Figure 3b, at 25%, the distilled trajectories still overlay the source with the behavior classes clearly separated, even for the validation clips, unlike the tangled structure at 10% data. Hence, a quarter of the corpus not only retains comparable tracking performance but also reconstructs the *geometry* of the source's behavior space.

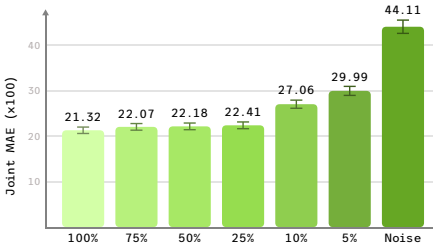


Figure 4: Data scaling for the M3 robot, evaluated by closed-loop joint MAE as we shrink the distillation corpus from 100% to 5% to only noise.

5.4 Generalize to unseen robots

Table 4 reports the generalizability of the unified encoder to unseen robots by training it on two robots and evaluating on the third one. Apart from cosine similarity and tracking performance, we also report signal gap closed $=$ (random $-$ pred)/(random $-$ oracle), representing how meaningful the latent signal is to the tracker, with the upper bound being the seen robot and the lower bound

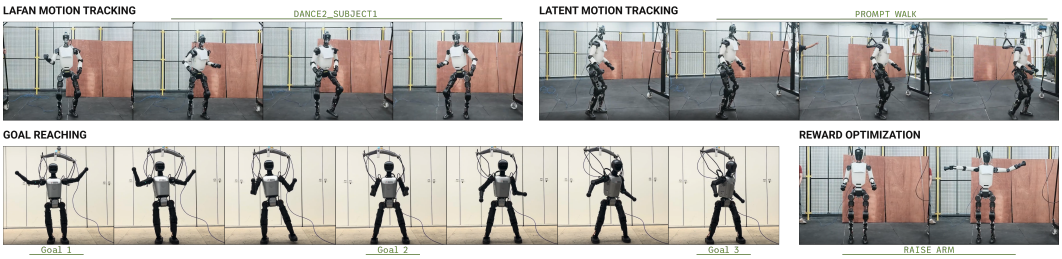


Figure 5: Real-world rollouts of T1 and M3 across all 4 prompt modes. Videos in our website.

being pure noise. The cross-embodiment transfer works best for morphologically similar robots. Specifically, an encoder that never saw M3 recovers 89.3% of the tracking performance of the encoder trained on all 3 robots, while for unseen T1, it recovers only 12.8%.

Cosine distance does not fully capture policy performance. With the cosine distance alone, the latents of unseen robots lie far away form the ground-truth, illustrated by 0.625 cosine similarity for the best case with M3. However, this encoder still generates latents that recover 89.3% of the performance of trackers trained on the oracles' latents. While cosine similarity directly answers how close $\hat{z}$ is to $z^\star$, it does not ask whether the policy reproduces the correct behavior. Our latent-conditioned policy is robust to some latent mismatch, provided that the inferred latents fall into the behavior cluster that the policy can effectively utilize, even when the cosine distance appears large.

Cosine distance relates to morphological proximity. In our experiments, M3 and N1 are a close pair both in terms of joint configuration and body shape, while T1 is smaller and significantly different in its morphology. From Table 4, when the unseen robot still has a close relative in training, cosine similarity lands at 0.59–0.63. In contrast, when the unseen robot is the T1 with no close relative, the latent similarity drops to 0.25. This result suggests that *given a significantly large set of embodiments where every new robot has a close relative, generalization to new robots then becomes interpolation rather than extrapolation*. Our finding aligns well with the conclusion from (Bohlinger & Peters, 2025), in which locomotion policies trained with thousands of randomized embodiments demonstrated zero-shot transfer to unseen robots.

5.5 Real-world deployment

We verify the performance of our latent-conditioned policy with its joint-conditioned counterpart in the real world on the M3 and T1 robots. Table 5 shows the quantitative results on the M3 with all 3 prompt modes, with the joint-conditioned policy staying slightly ahead of our latent-conditioned policy, similar to simulation. For the goal-reaching and reward optimization tasks, the latent-conditioned policy also performs comparably to the simulation results (Table 1), with some small degradation due to the sim-to-real gap.

Table 5: Sim2real results on 3 prompt modes: motion tracking (T), goal reaching (G) and reward optimization (R). Metrics: T and G are joint MAE in radians; R is normalized return.

Mode	Prompt	Real	Sim
T (joint ref.)	5 clips	0.1985	0.1777
T (latent, ours)	5 clips	0.2206	0.2024
G	9 poses	0.2675	0.2345
R	3 tasks	0.562	0.508

6 Conclusion

CrossBFM transfers a behavior space across humanoid embodiments by treating retargeted data of different robots using the same dataset as cross-embodiment correspondence and distilling the latents of a frozen source BFM through supervised learning. We propose a unified encoder with no per-robot parameters that converts proprioception into latent behavior in one training run under a GPU-hour. Our unified encoder also enables generalization to unseen motions of similar behaviors and to unseen robots of similar morphology. The tracking policies conditioned on distilled latents achieve tracking performances comparable to their joint-conditioned counterpart while allowing for smooth goal reaching by latent interpolation, and zero-shot reward prompting across 41 tasks. We also demonstrate that, with our pipeline, using only one-quarter of the dataset is enough to effectively distill the frozen latent space onto a new robot, retaining both the geometrical alignment and functionality. Future work may focus on the scaling law across two main axes, including richer training data for a more fine-grained behavior space and a larger robot training set for better generalization to unseen robots. Additionally, with our shared latent space, prompting with text instructions to address long-horizon loco-manipulation tasks is also a promising direction.

AI USE STATEMENT

In this work, we used generative AI tools for none of the tasks for which disclosure is required. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat data sets, or interpret results, and assistance with translation and support for qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to edit a research paper to improve readability. We have reviewed all AI-assisted work. Every proposed rewording was accepted, rejected, or modified by an author; no proposal was permitted to introduce or alter a claim, a number, or a citation; and all results reported in the paper were produced by the authors' own code and checked against the experiment logs. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The unified encoder architecture is specified index by index in Table 7 together with the joint-link mapping in Appendix A.3. We also provide the encoder training protocol (Tables 8–9), policy architecture, PPO hyperparameters, reward weights and terminations (Appendix A.5), domain randomization (Appendix A.6), metrics and their resolution (Appendix B.1), the reward-optimization task suite (Appendix B.2) and cross-embodiment transport region (Appendix B.3). Code will be released upon acceptance.

ETHICS STATEMENT

This work uses publicly available motion-capture data (LAFAN (Harvey et al., 2020)) retargeted onto humanoid robots in simulation. No human subjects, personal data, or physical deployments around people are involved.

REFERENCES

Bo Ai, Liu Dai, Nico Bohlinger, Dichen Li, Tongzhou Mu, Zhanxin Wu, K. Fay, Henrik I. Christensen, Jan Peters, and Hao Su. Towards embodiment scaling laws in robot locomotion. In *Conference on Robot Learning (CoRL)*, 2025.

João Pedro Araújo, Yanjie Ze, Pei Xu, Jiajun Wu, and C. Karen Liu. Retargeting matters: General motion retargeting for humanoid motion tracking. *arXiv preprint arXiv:2510.02252*, 2025.

Junik Bae, Kwanyoung Park, and Youngwoon Lee. TLDR: Unsupervised goal-conditioned rl via temporal distance-aware representations. In *Conference on Robot Learning (CoRL)*, 2024.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*, 2021.

Nico Bohlinger and Jan Peters. Multi-embodiment locomotion at scale with extreme embodiment randomization. *arXiv preprint arXiv:2509.02815*, 2025.

Nico Bohlinger and Jan Peters. Shape your body: Value gradients for multi-embodiment robot design. In *Conference on Robot Learning (CoRL)*, 2026.

Nico Bohlinger, Grzegorz Czechmanowski, Maciej Piotr Krupka, Piotr Kicki, Krzysztof Walas, Jan Peters, and Davide Tateo. One policy to run them all: an end-to-end learning approach to multi-embodiment locomotion. In *Conference on Robot Learning (CoRL)*, 2024.

Boyu Chen, Yi Chen, Lu Qiu, Jerry Bai, Yuying Ge, and Yixiao Ge. UniT: Toward a unified physical language for human-to-humanoid policy learning and world modeling. *arXiv preprint arXiv:2604.19734*, 2026.

Xuxin Cheng, Yandong Ji, Junming Chen, Ruihan Yang, Ge Yang, and Xiaolong Wang. Expressive whole-body control for humanoid robots. In *Robotics: Science and Systems (RSS)*, 2024.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations (ICLR)*, 2019.

Arnaud Fickinger, Samuel Cohen, Stuart Russell, and Brandon Amos. Cross-domain imitation learning via optimal transport. In *International Conference on Learning Representations (ICLR)*, 2022.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. MetaMorph: Learning universal controllers with transformers. In *International Conference on Learning Representations (ICLR)*, 2022.

Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. Robust motion in-betweening. *ACM Transactions on Graphics (SIGGRAPH)*, 39(4), 2020.

Tairan He, Zhengyi Luo, Xialin He, Wenli Xiao, Chong Zhang, Weinan Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. OmniH2O: Universal and dexterous human-to-humanoid whole-body teleoperation and learning. In *Conference on Robot Learning (CoRL)*, 2024.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS Deep Learning Workshop*, 2015.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pp. 4455–4464. PMLR, 2020.

Kihyun Kim, Chaeyun Kim, Jongho Shin, Taeyoun Kwon, Junghyun Kim, Mijin Koo, and Haon Park. PHASOR: Phase-anchored universal action representations for humanoid embodiments. *arXiv preprint arXiv:2606.01851*, 2026.

An T. Le, Kay Pompetzki, Jan Peters, and Armin Biess. Kinematics correspondence as inexact graph matching. In *German Robotics Conference (GRC)*, 2025.

Dichen Li, Bo Ai, Nico Bohlinger, Jan Peters, Hao Su, and Henrik I Christensen. Rapid embodiment adaptation for quadrupedal locomotion. *arXiv preprint arXiv:2608.01506*, 2026.

Yitang Li, Zhengyi Luo, Tonghe Zhang, Cunxi Dai, Anssi Kanervisto, Andrea Tirinzoni, Haoyang Weng, Kris Kitani, Mateusz Guzek, Ahmed Touati, Alessandro Lazaric, Matteo Pirotta, and Guanya Shi. BFM-Zero: A promptable behavioral foundation model for humanoid control using unsupervised reinforcement learning. *arXiv preprint arXiv:2511.04131*, 2025.

Qiayuan Liao, Takara E Truong, Xiaoyu Huang, Yuman Gao, Guy Tevet, Koushil Sreenath, and C Karen Liu. Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion. *Science Robotics*, 11(117):eadx8924, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Min Liu, Deepak Pathak, and Ananye Agarwal. Locoformer: Generalist locomotion via long-context adaptation. In *Conference on Robot Learning*, pp. 532–546. PMLR, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Zhengyi Luo, Jinkun Cao, Kris Kitani, Weipeng Xu, et al. Perpetual humanoid control for real-time simulated avatars. In *International Conference on Computer Vision (ICCV)*, 2023.

Zhengyi Luo, Ye Yuan, Tingwu Wang, Chenran Li, Fernando Castañeda, Sirui Chen, Zi-Ang Cao, Jiefeng Li, David Minor, Qingwei Ben, et al. Sonic: Supersizing motion tracking for natural humanoid whole-body control. *Science Robotics*, 11(117):eaed4592, 2026.

Austin Patel and Shuran Song. GET-Zero: Graph embodiment transformer for zero-shot embodiment generalization. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, 2017.

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. ASE: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics (SIGGRAPH)*, 41(4), 2022.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

RoboParty Lab Team. UFO: An open-source unsupervised reinforcement learning framework for humanoid control. `https://github.com/Roboparty/UFO`, 2026.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Carmelo Sferrazza, Dun-Ming Huang, Fangchen Liu, Jongmin Lee, and Pieter Abbeel. Body transformer: Leveraging robot embodiment for policy learning. In *Conference on Robot Learning*, pp. 3407–3424. PMLR, 2025.

Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. CALM: Conditional adversarial latent models for directable virtual characters. *ACM SIGGRAPH Conference Proceedings*, 2023.

Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric, and Matteo Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models. In *International Conference on Learning Representations (ICLR)*, 2025.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *International Conference on Learning Representations (ICLR)*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*, 2018.

Zhen Wu, Xiaoyu Huang, Lujie Yang, Yuanhang Zhang, Xi Chen, Pieter Abbeel, Rocky Duan, Angjoo Kanazawa, Carmelo Sferrazza, Guanya Shi, et al. Perceptive humanoid parkour: Chaining dynamic human skills via motion matching. *arXiv preprint arXiv:2602.15827*, 2026.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning (ICML)*, 2020.

Yashuai Yan and Dongheui Lee. Learning a unified latent space for cross-embodiment robot control. *arXiv preprint arXiv:2601.15419*, 2026.

Lujie Yang, Xiaoyu Huang, Zhen Wu, Angjoo Kanazawa, Pieter Abbeel, Carmelo Sferrazza, C Karen Liu, Rocky Duan, and Guanya Shi. Omniretarget: Interaction-preserving data generation for humanoid whole-body loco-manipulation and scene interaction. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Yanjie Ze, Siheng Zhao, Weizhuo Wang, Angjoo Kanazawa, Rocky Duan, Pieter Abbeel, Guanya Shi, Jiajun Wu, and C. Karen Liu. Twist2: Scalable, portable, and holistic humanoid data collection system. *arXiv preprint arXiv:2511.02832*, 2025.

Heng Zhi, Wentao Tan, Lei Zhu, Fengling Li, Jingjing Li, Guoli Yang, and Heng Tao Shen. MOTIF: Learning action motifs for few-shot cross-embodiment transfer. *arXiv preprint arXiv:2602.13764*, 2026.

A Implementation and Training Details

A.1 Robot platforms

Table 6 describes the four embodiments in our experiment. The source is a frozen BFM on the Unitree G1, from which we retain only the backward map B_S. The three targets differ from G1, and from each other, in DoF count, body count, joint names, and joint order.

Table 6: The four humanoid configurations.

Robot	Role	DoF	Bodies	Index filled	Morphology
Unitree G1	source	29	32	—	reference topology; 3-DoF waist, 3-DoF wrists
M3	target	27	30	27/33	nearest to the source; waist yaw only
Booster T1	target	23	24	21/33	wrist-less, 4-DoF arms, 2-DoF head
Fourier N1	target	23	29	23/33	one wrist DoF per arm; waist yaw only

A.2 The encoder input

Table 7 provides the full encoder input, as described in Section 4.2. In this setting, every robot presents the same 163 dimensions regardless of how many degrees of freedom it has, and the presence masks tell the encoder whether the current entry is present or missing.

Table 7: The unified cross-embodiment encoder canonical input.

Range	Contents	Dims
[0:24)	key-body positions, root-relative, / SCALE	8×3
[24:48)	key-body velocities (finite difference $\times$ fps)	8×3
[48:56)	key-body presence mask	8
[56:57)	root height / SCALE	1
[57:61)	root rotation quaternion	4
[61:64)	root linear velocity / SCALE	3
[64:97)	joint positions q, canonical slot order	33
[97:130)	joint velocities $\dot{q}$	33
[130:163)	joint presence mask	33
Total		**163**

A.3 Encoder architecture

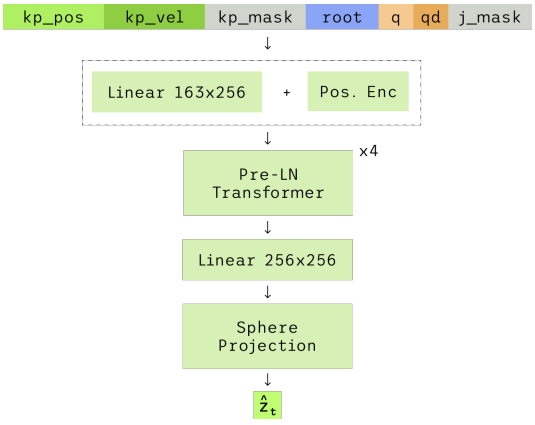


Figure 6: Unified encoder architecture.

The encoder is a sequence-to-sequence transformer that maps a window of canonical proprioceptive frames to one latent per frame. We provide its architecture in Table 8. Most of that configuration is conventional with two crucial design choices. The first is *pre-LayerNorm*, which turned out to be

necessary to prevent mode collapse. When we used post-LayerNorm instead, the model collapsed to one constant latent, and the cosine objective cannot recover because every prediction then points in the same direction. The second is the *output projection onto the sphere*, which mirrors the source model's own `project_z` and therefore prevents the encoder from emitting a latent that falls out of the manifold of the frozen tracker. We make this projection part of the network rather than a post-processing step, so that gradients flow through it during training.

Three more conventions make the encoder universal, which are *fixed* data conventions rather than learned parameters, so none of them adds per-robot weights to the model.

1. **Name maps.** The key-body indices follow the same correspondence that the trackers were trained against. The joint index order follows the joint ordering that the retargeting scripts use.
2. **Sign canonicalization.** T1 measures waist yaw with the opposite sign convention to G1, M3 and N1. We therefore provide an additional index for this joint rotation convention.
3. **Robot scale normalization.** We divide all lengths by SCALE, which is the median root height of that robot over its own corpus (0.7956 m for M3, 0.6743 m for N1 and 0.6552 m for T1). This normalization is what lets a 0.65 m robot and a 0.80 m robot present geometrically comparable inputs to the same encoder.

With these conventions, adding new robots only requires a name map and a scale constant, which require only a configuration file rather than additional training or architecture-wise modification.

Table 8: Encoder configuration.

Component	Value
Input	
Input width	163 (Table 7)
Window length T	64 frames $= 2.13$ s at 30 fps
Input projection	$\mathrm{Linear}(163 \to H)$
Positional encoding	sinusoidal, max length 1024, non-persistent buffer
Trunk	
Layers	$4\times$ pre-LayerNorm transformer encoder layers
Attention heads	8
Feed-forward width	$4H$
Activation	GELU
Dropout	0
Attention mask	bidirectional (default); upper-triangular in the causal variant
Head	
Output head	$\mathrm{LayerNorm}(H) \to \mathrm{Linear}(H \to 256)$
Output projection	$z \leftarrow \sqrt{256}\, v / \max(\|v\|, \varepsilon), \varepsilon = 10^{-6}$
Output rate	one latent per frame (not per window)
Width and parameter count	
Unified encoder	$H = 256$, 3.27M parameters total (one checkpoint, all robots)
Per-robot encoders	$H = 512$, 12.8M parameters each, 38.5M total
Per-robot conditioning	none; robot identity enters only through the fixed slot maps

A.4 ENCODER TRAINING

Table 9 lists the training protocol used for encoder ablation experiments in this paper. For data sampling, `MultiRobotWindowDataset` draws a single window index and then returns *the same window* for all R robots, and it supervises all R copies coupled with a shared ground-truth $z^\star$. Every optimizer step therefore presents R morphological views of one motion against a shared label.

A.5 TRACKER ARCHITECTURE AND TRAINING

Each embodiment receives its own tracking policy trained with PPO (Schulman et al., 2017) in mjlab. The two conditioning policies (latent-conditioned and joint-conditioned) of Table 1 run in the *same* environment with two different configurations. First, we swap the motion command for a variant that additionally carries the frozen G1 latent. Second, we remove the five reference-derived terms (`motion_root_vel_xy_b`, `motion_root_z`, `motion_root_roll_pitch`,

Table 9: Encoder training protocol across experiments.

Parameter	Value
Objective	$\mathcal{L} = \frac{1}{N}\sum_t \big(1 - \cos(\hat{z}_t, z_t^\star)\big)$
Optimizer	AdamW
Learning rate	2×10^{-4}
Weight decay	10^{-4}
Gradient clipping	global norm 1.0
Epochs	400
Batch size (unified)	22 windows $\times$ 3 robots $= 66$
Batch size (per-robot)	64 windows
Window length	64 frames
Training data	LAFAN, 30 clips, 195,746 frames per robot
Validation data	10 clips (`walk:3,*:1` quota)
Evaluation	validation cosine every 10 epochs
Wall-clock	under one hour on a single consumer GPU

`motion_root_yaw_ang_vel_b`, `motion_joint_pos`) from the actor and replace them with z. Everything else is shared between the two variants: the rewards, the terminations, the randomization, the PPO budget, and even the motion `.pkl` files, which carry z as one extra field alongside the standard tracking reference. We leave the critic with privileged future reference untouched in both variants, which exists only in simulation and is discarded at deployment.

Table 10: Tracker configuration and PPO hyperparameters.

Parameter	Value
Environment	
Simulator	mjlab (MuJoCo), flat terrain
Physics rate	200 Hz (timestep 0.005 s)
Control rate	50 Hz (decimation 4)
Episode length	10 s $=$ 500 control steps
Parallel environments	4096
Action space	joint position targets for all actuated DoF, per-robot action scale, applied as an offset from the default pose
Phase sampling	uniform over the clip
Observations	
Actor, current	$[z^{256} \mid \text{proprio}]$: base angular velocity ($\times 0.25$), IMU roll/pitch, q, $\dot{q}$ ($\times 0.05$, ankle DoF masked to 0), previous action
Actor, history	proprioception only, 11 steps, encoded to a 64-dim latent
Joint-conditioned variant	replaces z^{256} with the reference terms listed above
Critic	privileged future reference sequence + current proprioception + privileged extras (base linear velocity, root pose, key-body positions, foot contact, and the sampled randomization parameters)
Networks	
Actor / critic trunk	MLP $(512, 256, 128)$, ELU, running observation normalization
Action distribution	Gaussian, scalar state-independent std, $\sigma_0 = 1.0$
History encoder	64-dim latent; motion/command encoder 64-dim
PPO	
Steps per environment per update	24
Iterations	30,000 ($\approx 2.9 \times 10^9$ environment steps)
Learning rate	10^{-3}, adaptive schedule with target KL $= 0.01$
Discount γ / GAE λ	0.99 / 0.95
Clip range	0.2, with clipped value loss
Value / entropy coefficient	1.0 / 0.005
Learning epochs / minibatches	5 / 4
Max gradient norm	1.0
Seed	0 for the A/B; 3 seeds for the numbers in Table 1
Wall-clock	under ten hours per embodiment on one RTX 4090

Table 11: Tracker reward terms.

Tracking		*Regularization*	
Term	Weight	Term	Weight
joint position	2.0	alive	$+0.5$
joint velocity	0.2	feet air time	1.0 (5.0)
key-body position	2.0	action rate	-0.03 (-0.1)
key-body position (global)	2.0	joint acceleration	-2.5×10^{-8} (-5×10^{-8})
root height	1.0	ankle joint acc.	-2×10^{-7} (-1×10^{-7})
root rotation	1.0	joint limit	-10.0
root linear velocity	1.0	DoF position limits	-5.0
root angular velocity	1.0	self collision	-10.0
		DoF torque limits	-1.0
		feet stumble	-1.25
		feet slip	-0.1
		body angular vel. xy	-0.01
		joint velocity	-1×10^{-4}
		ankle joint velocity	-2×10^{-4}
		feet contact force	-5×10^{-4}

A.6 Domain randomization and observation noise

Both conditioning variants train under identical randomization, listed in Table 12. Domain randomization is disabled during the evaluation rollouts.

Table 12: Domain randomization.

Event	Mode	Operation	Range
Pelvis added mass	startup	add	$[-3.0, 3.0]$ kg
Pelvis COM offset	startup	add	x: $[-0.025, 0.025]$, y, z: $[-0.05, 0.05]$ m
Motor strength (k_p, k_d)	startup	scale	$[0.8, 1.2]$
Foot friction	startup	set	$[0.3, 1.2]$
Joint encoder bias	startup	add	$[-0.01, 0.01]$ rad
Actuator delay	startup	—	0–4 physics steps, per-environment phase
Push (base velocity)	interval every 1–3 s	set	$v_{x,y}$: ±0.5, v_z : ±0.2 m/s; $\omega_{r,p}$: ±0.52, ω_y : ±0.78 rad/s
Additive uniform observation noise (actor only)			
Base angular velocity			±0.1
IMU roll / pitch			±0.1
Joint position q			±0.01
Joint velocity $\dot{q}$			±0.1

B Tasks and Metrics

B.1 Metrics

Policy-free metrics. *Alignment* is the validation cosine between the encoder's prediction and the frozen source latent, $\cos(\hat{z}_t, z_t^\star)$, averaged over the validation frames. *Agreement* is the cosine $\cos(\hat{z}_{r,t}, \hat{z}_{r',t})$ between two robots' predictions on the same validation frame.

Closed-loop metrics with policy. For motion tracking and goal reaching task, we report the following metrics. *Joint MAE* is the mean absolute deviation, in radians, between the executed joint positions and the retargeted reference, averaged over the episode and over the joints. *Key-body error* `mpjpe_local` is the mean Euclidean distance, in meters, between the executed and the reference key-body positions, both expressed in the root frame.

B.2 Reward optimization task suite

We evaluate reward prompting on the 41 task rewards designed for G1 robot (Li et al., 2025) (Table 13). Each reward is a function of the state, normalized to $[0, 1]$. For every task, we infer a single z_{rew} from the new robot's cached buffer (Equation 8) and then hold that latent constant for 300

control steps, which is one full episode. We score only the settled second half of the episode after the robot reached a stable pose.

Table 13: The 41 source task rewards by group.

Category	Count	Tasks
Standing	2	`move-ego-0-0`, `move-ego-low0.5-0-0`
Locomotion	8	`move-ego-0-0.7`, `move-ego-90-0.7`, `move-ego--90-0.7`, `move-ego-low0.6-0-0.7`, `move-ego-0-0.3`, `move-ego-90-0.3`, `move-ego-180-0.3`, `move-ego--90-0.3`
Rotation	2	`rotate-z-5-0.5`, `rotate-z--5-0.5`
Arm raise	4	`raisearms-{l,m}-{l,m}`
Locomotion + arms	16	`move-arms-`θ`-`v`-{l,m}-{l,m}` for $(\theta, v) \in \{(0, 0.7), (90, 0.7), (180, 0.4), (-90, 0.7)\}$
Rotation + arms	6	`spin-arms-`$\pm$`5-{l-l, l-m, m-l}`
Ground poses	3	`crouch-0`, `crouch-0.25`, `sitonground`
Total	**41**	

B.3 Marginal transport

A reward defined on G1 often refers to an absolute configuration that a differently proportioned target (such as the shorter one T1) cannot reach. When that happens, there is no pose in the cached buffer for the new robot that could return a positive reward. Among these rewards, the quantity affected by this is *arm-tip height*, which the 26 arm-related tasks' thresholds roughly divide into a "low" band and a "medium" band. If those thresholds are directly transferred to other robots, the G1's medium band of ≥ 1.0 m asks a robot that is only 0.66 m tall for something it physically cannot do.

We therefore map each threshold through the two robots' empirical marginals. We compute those marginals over the 40 frame-aligned LAFAN retargeted motions, which give us $264{,}625$ frames per robot. We convert each G1 threshold θ into a quantile of the G1's own arm-tip height distribution, then use this quantile to trace back the corresponding threshold of the target's distribution, so that $\theta^{(r)} = \Phi_r^{-1}(\Phi_S(\theta))$. The G1's "low" band edges of 0.6 and 0.8 m sit at quantiles 0.079 and 0.320, and its "medium" lower edge of 1.0 m sits at quantile 0.938.This way, a robot with a smaller reachable range receives a proportionally tighter ramp instead of an absolute one it could never satisfy (Table 14).

Table 14: Arm-height bands after quantile mapping, given as (lower, upper, margin) in meters.

Robot	Low band	Medium band
M3	$(0.497,\ 0.824,\ 0.276)$	$(1.157,\ \infty,\ 0.138)$
N1	$(0.416,\ 0.627,\ 0.240)$	$(0.925,\ \infty,\ 0.120)$
T1	$(0.509,\ 0.675,\ 0.172)$	$(0.820,\ \infty,\ 0.086)$

C On the Encoder design choice

C.1 Causal encoding for behavior prediction

For bidirectional encoders, each frame's latent is passed with roughly one second of future frames. That is acceptable for offline evaluation, but it requires future observations unavailable during live teleoperation. We therefore compare bidirectional and causal attention and evaluate their effects on latent alignment and closed-loop tracking (Table 15). We compare the two attention masks at 100% and 25% of the distillation corpus, measured by *alignment* – the validation cosine against $z^\star$ – and by the closed-loop joint tracking error *joint MAE*.

In both metrics, the causal encoder achieves comparable and performant results compared to its bidirectional counterpart. On the full corpus, causal attention reduces validation cosine similarity by 0.0275 relative to bidirectional attention, while lowering closed-loop joint MAE from 0.2068 to 0.2043 rad, matching the reported MAE under oracle conditioning. With 25% of the corpus, the two encoders achieve nearly identical alignment, while the causal encoder yields lower joint MAE

Table 15: Causal versus bidirectional attention.

Corpus	Encoder	Alignment ↑	Joint MAE (rad) ↓	Δ vs. oracle
100%	bidirectional	**0.8695**	0.2068	+0.0025
	causal	0.8420	**0.2043**	0.0000
25%	bidirectional	0.6825	0.2129	+0.0086
	causal	**0.6829**	**0.2045**	+0.0002
—	oracle $z^\star$	—	0.2043	—

Table 16: Unified versus per-robot encoders.

		Per-robot	Unified	Δ
Alignment ↑	M3	0.9026 ± 0.0017	0.8899 ± 0.0040	-0.0127
	T1	0.8307 ± 0.0012	0.8265 ± 0.0058	-0.0042
	N1	0.8879 ± 0.0011	0.8698 ± 0.0043	-0.0181
Agreement ↑	all pairs	0.867–0.913	**0.938–0.965**	+0.07

(0.2045 versus 0.2129 rad). These results suggest that, under the evaluated training settings, causal attention maintains tracking performance despite a modest reduction in alignment at full data, while removing the need for future observations and enabling real-time reference streaming.

C.2 Unified encoder versus embodiment-specific encoders

Table 16 compares the unified encoder against three independently trained per-robot encoders. While alignment results (inferred latent cosine distance to ground-truth latent) are similar between per-robot encoders and unified encoder, the unified encoder generates latents with higher agreement among 3 new robots. This illustrates that the unified design significantly improves cross-embodiment latent consistency without sacrificing ground-truth alignment. This is because the per-robot encoders are trained on the same data and with the same architecture, but they are not constrained to agree with each other. Together with generalizability to unseen robots of similar morphology (demonstrated in Section 5.4), the unified encoder is a better choice for a shared behavioral coordinate system that enables commanding the whole platform with the same latent, which is the central goal of this work. We further investigate the effect of the unified encoder's trunk width in Table 17. Reducing the trunk width from $H{=}512$ to $H{=}256$ *improves* mean cosine by $+0.010$ ($0.8548 \rightarrow 0.8652$), while $H{=}128$ significantly degrades the performance of the encoder.

C.3 Sensitivity to retargeting error

Section 4.3 relies on the assumption that retargeting already supplies frame-level correspondence. Here we test sensitivity to violations of this assumption by perturbing the retargeted inputs and their temporal alignment. Corruption is applied to **M3 encoder data** while the ground-truth of G1 $z^\star$, the frozen trackers, the reference motions used to train trackers are unchanged. We conduct the experiment in two regimes: 1) *training time*: the unified encoder is retrained with 2 uncorrupted datasets and a corrupted one; 2) *inference time*: the trained encoder is fed corrupted input at inference.

Data corruption strategy. Every corruption is applied to the canonical proprioceptive frames that the encoder takes as input, rather than the ground-truth labels or anything the tracker sees in order to avoid tracker retraining. Two strategies perturb the *geometry* of the correspondence and two perturb its *timing*:

- **Geometry - Systematic bias ($\times N$).** We draw a fixed per-joint offset vector and apply it to every frame of the corpus. This is how a miscalibrated retargeter fails for a wrong joint-limit mapping or offset calibration. We set $\times 1$ to 0.027 rad rms, which is the size of the disagreement between two plausible tunings of GMR (Araújo et al., 2025) (the retargeting pipeline used in our paper).
- **Geometry - Non-systematic noise (σ).** A zero-mean Gaussian perturbation drawn independently for each frame and each joint. In practice, this is the more commonly seen mismatch between retargeters, which affects each frame differenly rather than staying as a consistent bias.

Table 17: Trunk width of the unified encoder, measured against $z^\star$.

H	M3	T1	N1	Mean
512	0.8788	0.8254	0.8603	0.8548
256 (ours)	**0.8928**	**0.8304**	**0.8722**	**0.8652**
128	0.6968	0.6722	0.6841	0.6844

- **Timing - Constant timing offset (δ frames).** Every window is paired with a label δ frames away (δ=1 equals to 33 ms at 30 fps). This error typically belongs to a mismatch in synchronization or motion up/downsampling.
- **Timing - Per-clip timing offset (k).** An offset drawn independently for each clip from $\mathcal{U}\{-k, \ldots, k\}$, which models a retargeter whose per-clip alignment is unreliable.

Table 18: **Deployment regime.** Pretrained encoder with corrupted inference input. *Phase*: per-clip offset that best match a clean clip; *cos@phase*: recovered cosine upon back-shifting.

Corruption	Align ↑	Phase	cos@phase ↑↑	MAE (rad) ↓	Gap closed ↑
clean	0.8916	0	0.8916	0.1655	99.4%
bias ×1	0.8897	0	0.8897	0.1626	101.1%
bias ×2	0.8833	0	0.8833	0.1611	101.9%
bias ×4	0.8580	0	0.8580	0.1647	99.8%
bias ×8	**0.7647**	0	0.7647	**0.1913**	84.4%
bias ×16	**0.5210**	0	0.5210	**0.2849**	30.2%
Gaussian σ=0.05	0.8684	0	0.8684	0.1678	98.0%
shift 2 fr	0.8623	−2	0.8915	0.1664	98.9%
shift 8 fr	0.6589	−8	0.8913	0.1890	85.8%
shift 32 fr	0.5416	−32	0.8917	0.2127	72.1%
per-clip k=4	0.8692	+1	0.8914	0.1721	95.5%

Table 19: **Training regime.** The unified encoder retrained with M3 corrupted input.

Corruption	Align ↑	Phase	cos@phase ↑	MAE (rad) ↓	Gap closed ↑
clean	0.8632	0	0.8644	0.1669	98.6%
bias ×2	0.8525	0	0.8537	0.1740	94.4%
bias ×4	0.8242	0	0.8254	0.1848	88.2%
Gaussian σ=0.05	0.8427	0	0.8440	0.1675	98.2%
shift 2 fr	0.8191	**+2**	0.8673	0.1755	93.6%
shift −2 fr	0.8033	**−2**	0.8599	0.1673	98.3%
shift 4 fr	0.7039	**+4**	0.8621	0.1853	87.9%
per-clip k=4	0.8191	0	0.8347	0.1734	94.8%

Timing: the original timeline can be recovered by a search. At every magnitude we tried, the phase scan recovers the injected offset exactly, and the cosine returns to 0.8913–0.8917 against a clean 0.8916. The latent is not degraded but rather becomes the correct latent for a neighboring frame, and even at 32 frames (1.07 s) retrieval is undisturbed at 0.998. Tracking performance recovers after applying the same back-shifting. With a phase search enabled, the corresponding back-shifting of 8, 16 and 32 frames gives phase-aligned MAE of 0.1646, 0.1641 and 0.1621 rad against a clean 0.1655. The residual +0.047 in Table 18 is therefore the cost of executing the right motion at the wrong time, which is a property of the prompt rather than of the encoder. Training on a shifted corpus has a similar behavior. Here the encoder *learns* the shifted correspondence rather than correcting it, so when it is fed clean input, it outputs a latent offset by exactly δ, which is the phase column of Table 19 that recovers +2, −2 and +4 in both directions. Moreover, the cosine at that offset returns to 0.8599–0.8675 against a clean 0.8644. For per-clip random offset, as no global pattern exists, the encoder performs badly even with back-shifting.

Geometry: tolerant to non-systematic noise up to ×4. Non-systematic noise by Gaussian perturbation at σ=0.05 costs +0.002 rad during inference and +0.001 rad during training, which is tolerable for our tracking policy. For systematic bias, it is harmless up to roughly ×4 and destructive

Table 20: Data corruption behavior across the two regimes.

	Timing (δ, k)	Geometry (bias $\times N$, σ)
Alignment	collapses ($0.89 \to 0.54$)	degrades smoothly
Closed-loop tracking	unchanged once re-phased	flat to $\times 8$, breaks by $\times 16$
Recoverable	yes, by a one-dimensional search	not shown to be recoverable

by $\times 8$, demonstrated by the clear drop in both alignment (from 0.8580 to 0.7647) and MAE (from 0.1647 to 0.1913). We summarize our findings with data corruptions in Table 20.

C.4 Cross-embodiment generalizability experiment protocol

We compare nine encoders, which are three folds with three seeds. Our training procedure follows Appendix A.4, with modifications in the robot set and data loader batch size. As the sampler returns the same window for every robot, the effective batch is `batch_size` $\times R$. We therefore use 33 for the two-robot folds, which gives $33 \times 2 = 66$ windows per step and matches the 22×3 of the original encoder. The validation clip split is the quota split of Section 5.2, that is 10 clips spanning all eight behaviors, and it is the same for every robot. We measure the random-latent floor inside this harness as well, and it comes out at 0.3371 on M3 rather than the 0.4411 of Section 5.3 due to the difference in the rollout protocol (10 clips instead of all 40 clips). The oracle, predicted, and random cells of Table 4 all share one harness and one clip set and are compared amongst each other.

D Flow-based Motion Generator

Because the distilled space $\mathcal{Z}$ is shared by every embodiment (Section 4.3), a generator that samples trajectories *inside* $\mathcal{Z}$ prompts all robots at once, with no reference motion and no backward map at inference. We train one conditional rectified-flow model (Liu et al., 2023; Lipman et al., 2023) over chunks of $H = 64$ consecutive latents $z_{1:H} \in \mathcal{Z}^H$ drawn from the retargeted corpus. The velocity field v_θ is a pre-LN transformer (Vaswani et al., 2017) (6 layers, width 512, 8 heads, $\approx$ 16M parameters) whose per-frame tokens are $[z_t^{(i)} \,\|\, \phi(t)]$ — the noised latent concatenated with a sinusoidal embedding of the flow time — mixed by RoPE self-attention over the 64 frames and by cross-attention to the condition tokens described next.

Conditions. Three conditions enter through two paths. The *mode* prompt — one learned embedding per behavior plus a NULL row, which doubles as the unconditional token and as the fallback for a clip whose name does not parse (a frozen T5 encoder (Raffel et al., 2020) is a drop-in replacement when free-form captions are available) — and the *history* of the 16 preceding latents are both turned into tokens and concatenated into one cross-attention memory, so every frame can read either. The *motion* condition is instead frame-aligned: the retargeted features are embedded per frame and added to the corresponding frame token through a scalar gate initialized to zero, so the generator starts as a pure mode-and-history model and admits frame-aligned information only as it earns loss. Each condition is dropped independently with probability 0.1 during training, which supplies the unconditional branch for guidance and lets any subset of the three be used at sampling time.

Objective. For a data chunk z_1 and noise $\epsilon \sim \mathcal{N}(0, I)$ we take the straight interpolant $z_t = (1-t)\epsilon + t\, z_1$ with target velocity $v^\star = z_1 - \epsilon$, and minimize the masked conditional flow-matching loss

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{t,\epsilon,(z_1,c)} \left\| v_\theta(z_t, t, c) - v^\star \right\|_2^2, \tag{10}$$

averaged over valid frames only, with t drawn from a logit-normal schedule and c the (partially dropped) conditions. Sampling integrates $\mathrm{d}z/\mathrm{d}t = v_\theta$ from $t = 0$ to $t = 1$ with 20 denoising steps and classifier-free guidance (Ho & Salimans, 2022) at scale 2.5, re-applying the radial projection $\mathrm{proj}_{\mathcal{Z}}$ after every step so the trajectory stays on the sphere of radius $\sqrt{d}$ that the trackers were trained against. Long sequences are produced autoregressively: each chunk is conditioned on the last 16 generated frames. The output is a latent stream that the frozen latent-conditioned policies of Section 5.1 consume directly — no tracker, no kinematic decoder, and nothing robot-specific in between.

Data selection and window sampling. Targets are the latents already baked into the retargeted corpus by the Stage-1 encoder, so no additional labeling is required. Mode labels are parsed from

the clip name, and a clip that does not parse keeps the NULL mode rather than being dropped, so unlabeled data still trains the unconditional branch; clips shorter than the 64-frame horizon are discarded. Windows are drawn by sampling a clip uniformly and then a start index within it, which equalizes exposure across behaviors on a corpus whose clips differ widely in duration. Evaluation holds out *whole clips*, so the reported flow loss measures generalization to an unseen motion rather than to an unseen window of a seen motion.

E PIPELINE INFERENCE

We extensively evaluate our framework in both simulation and in the real world across all four prompt modes. Full videos in our website



Figure 7: Motion tracking and goal-reaching task for LAFAN motions.

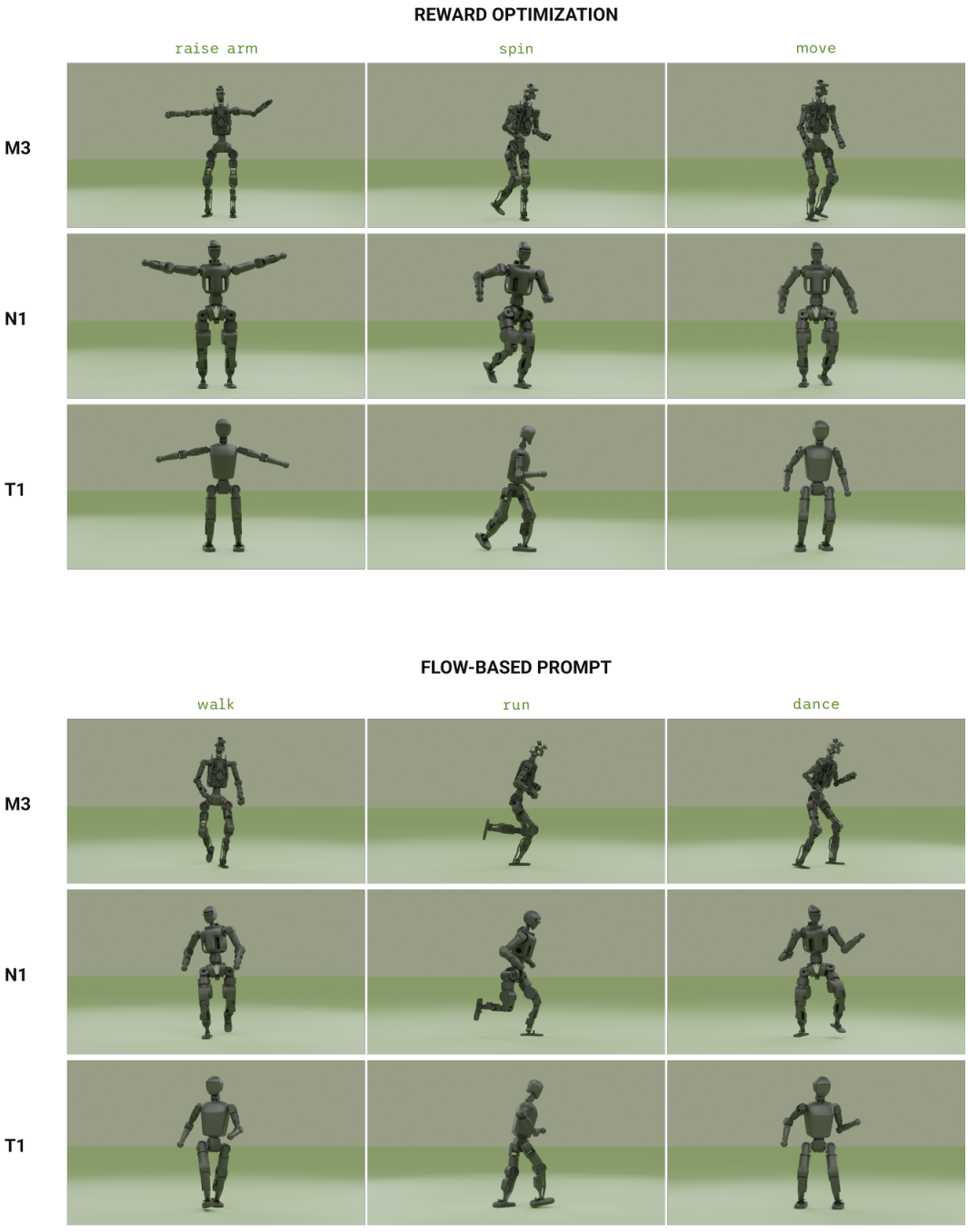


Figure 8: Reward optimization for 3 tasks in the 41-task suite and 3 prompt conditions.