PRISM: Polynomial Representations for Interaction-Structured Motor Control

Seung Hyun Lee **Stella X. Yu**
University of Michigan

Abstract: Robot policies map observations to actions, typically using multilayer perceptrons (MLPs). Yet robot dynamics and control objectives involve multiplicative interactions among observable physical variables. MLPs can approximate these interactions but must learn them implicitly, at a representational cost. We introduce PRISM (Polynomial Representations for Interaction-Structured Motor Control), a physics-motivated yet data-driven representation that learns multiplicative interactions by recursively adding scaled bilinear features, without enumerating all polynomial terms. Replacing only the MLP's first affine layer, PRISM is plug-and-play for reinforcement and imitation learning, requiring no changes to training objectives, sensing, or controllers. Across humanoid locomotion and contact-rich manipulation, PRISM improves survival from 51.0% to 92.5%, outperforming even larger MLPs. More strikingly, its performance advantage persists under out-of-distribution changes in friction, robot mass, and payload, without retraining. PRISM also exhibits sensorless compliance-like behavior, increases failure-recovery success from 40.9% to 52.3% without explicit admittance control, improves real-robot manipulation, and learns features that better predict physical quantities. These results make PRISM a compelling alternative to standard MLPs for robot policy learning. The project page is available at https://lsh3163.github.io/prism/.

Keywords: Polynomial interaction, Motor control, Sensorless compliance

1 Introduction

Every robot policy answers the same question: Given what the robot can observe, what action should it take next? In learning-based robot control, multilayer perceptrons (MLPs) are widely used to map observations to actions. MLPs are general-purpose function approximators that can learn mappings between diverse inputs and outputs without assuming how they are related. This flexibility suits visual classification, where low-level light intensities are mapped to high-level semantic categories.

Robot control, however, involves a different input-output relationship. In visual classification, an output label describes the input image but does not change it. A robot policy, by contrast, acts on the same physical body it observes: Its action outputs change the robot's state, which in turn changes its subsequent proprioceptive inputs. This input-output coupling is governed by the body's dynamics, even though these dynamics are not explicitly provided to the policy. For articulated robots, Newtonian dynamics take the form [1]:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = Q, \tag{1}$$

where q describes the robot configuration, including joint positions, $\dot{q}$ the corresponding velocities, M the mass matrix, $C(q,\dot{q})\dot{q}$ the Coriolis and centrifugal effects, $g(q)$ the generalized gravitational forces and torques, and Q the applied generalized forces. Crucially, these dynamics couple joint motions through products of physical variables.

Multiplicative interactions extend beyond robot dynamics to control objectives. Energy-efficient motion and regulated contact involve quantities such as joint mechanical power and contact work.

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

These relationships share a common algebraic structure – products of physical variables:

$$[C(q,\dot{q})\dot{q}]_i = \sum_{j,k} c_{ijk}(q)\dot{q}_j\dot{q}_k, \qquad P_{\text{joint}} = \sum_i \tau_i \dot{q}_i, \qquad W_{\text{contact}} = \int_{t_0}^{t_1} f_c^\top v_c \, dt. \tag{2}$$

Although some variables such as joint torques and contact forces may not be directly observed, their effects can be reflected in proprioceptive and action histories. Learning effective actions can thus benefit from explicitly representing multiplicative interactions among available observations.

Standard MLPs, however, do not represent such interactions directly. They compose affine layers with nonlinear activations such as ReLU [2] or ELU [3]. They can approximate multiplicative interactions [4, 5], but must learn them implicitly. Finite ReLU networks are piecewise affine and cannot represent quadratic products exactly; greater accuracy requires more network complexity [6]. Smoother activations such as ELU avoid this restriction, but still learn these interactions implicitly, consuming capacity in the shallow MLPs common in robot control.

We propose **PRISM**, **P**olynomial **R**epresentations for **I**nteraction-**S**tructured **M**otor Control, to explicitly represent and learn multiplicative interactions among policy observations (Fig. 1). **1) We learn multiplicative interactions rather than prescribe them.** Rather than specifying robot dynamics or handcrafting physical quantities, PRISM learns useful products among task commands, proprioceptive state histories, and past actions, providing a physics-motivated inductive bias without imposing a dynamics model. **2) We represent these interactions compactly by recursively adding bilinear terms.** Rather than enumerating polynomial terms, PRISM constructs degree-K features from K affine branches through scaled element-wise products. Replacing only the first affine layer of an MLP, PRISM is plug-and-play for reinforcement and imitation learning, requiring no changes to training objectives, controllers, or sensing. Zeroing all scales reduces PRISM to a standard MLP.

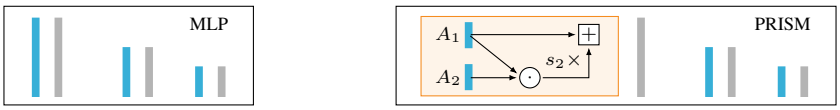


Figure 1: **PRISM adds learnable polynomial interactions to MLPs by replacing only the first affine layer. Left**: A three-layer MLP with affine and ELU layers. **Right**: The orange-shaded block in PRISM combines two affine branches as $A_1 + s_2 \times (A_1 \odot A_2)$. Additional branches can recursively extend the representation to higher orders; all other layers remain unchanged.

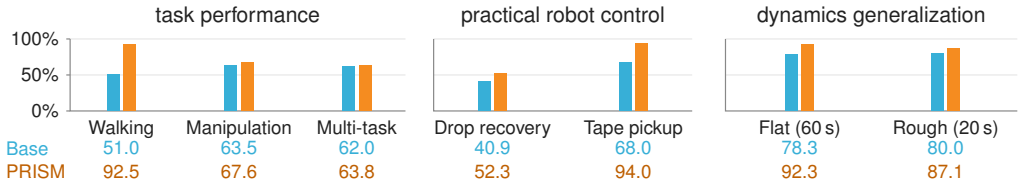


Figure 2: **PRISM improves task performance and practical robot control, while generalizing to out-of-distribution physical conditions.** Cyan and orange denote baseline and PRISM policies. *Left:* Humanoid-Gym [7] survival and manipulation success on ManiSkill3 [8] and LIBERO [9]. *Middle:* StackCube [8] failure recovery and real-robot SO-101 tape pickup. *Right:* Mean survival rates across six out-of-range dynamics settings on each terrain: flat (60-second trials) and rough (20-second trials), varying friction, mass, and payload. PRISM outperforms baselines across settings.

Experiments on humanoid locomotion and contact-rich manipulation reveal several benefits (Fig. 2). **1)** PRISM substantially outperforms standard MLPs, raising survival in Humanoid-Gym [7] from 51.0% to 92.5%, gains not matched by larger MLPs. **2)** It exhibits sensorless compliance-like behavior, rapidly relaxing contact force without explicit admittance control. It also increases failure-recovery success from 40.9% to 52.3%. On a physical SO-101, tape-pickup success rises from 68% to 94%. **3)** More strikingly, its advantage persists under out-of-distribution shifts in friction, mass, and payload, without retraining. **4)** Linear probes show that its features better predict mechanics-inspired quantities, including joint power, slip, contact impulse, and a contact-work proxy. Together, these findings make PRISM a compelling alternative to standard MLPs in robot policy learning.

2 Related Work

Polynomial and multiplicative neural networks. Multiplicative interactions expand the representable function classes of neural networks and provide a useful inductive bias for coupling multiple information streams [10]. Polynomial neural networks further use factorized parameterizations to represent higher-order dependencies without explicitly enumerating all monomials [11, 12, 13]. Polynomial expansions have also been studied as a tool for analyzing the behavior of trained neural networks [14]. Related work in robot learning incorporates physical priors through auxiliary estimators, constrained objectives, or physics-informed loss functions [15, 16, 17]. These approaches explicitly estimate selected physical quantities or enforce task-specific consistency constraints during training. In contrast, PRISM does not impose a physical model or require auxiliary physical supervision. Instead, it introduces a factorized low-degree interaction structure into the proprioceptive representation itself. Thus PRISM learns physical interactions from data through a structural inductive bias, while remaining sensor-minimal at deployment.

Robotic control with proprioceptive data. Practical deployment often imposes strict sensing constraints, making proprioception central to robust robot control [18, 19, 20, 21]. Legged-robot policies commonly use privileged training and history-based adaptation modules to infer the effects of changing terrain, payload, and dynamics [22]. Recent humanoid systems additionally emphasize integrated perceptual-motor representations for complex terrain and fall recovery [23, 24]. In contact-rich manipulation, other approaches introduce additional interaction modalities, including force, tactile, audio, or contact-related observations [25, 26, 27, 28, 29], or employ specialized compliance controllers [30, 31, 32]. PRISM is complementary to these approaches. Rather than reconstructing unobserved physical parameters, it exposes interactions among signals already available to the policy. This allows the policy to represent the observable consequences of changing contact and dynamics directly within its proprioceptive conditioning pathway.

3 Polynomial Representations for Interaction-Structured Motor Control

PRISM introduces a factorized polynomial representation into the observed-state pathway of a robot policy. It augments a direct affine path with products of learned affine factors, making multiplicative interactions available to the downstream policy. Fig. 3 illustrates the representation and its integration into reinforcement-learning actors and imitation-learning conditioning pathways. We first describe the policy inputs, then present the polynomial construction and its use in policy learning.

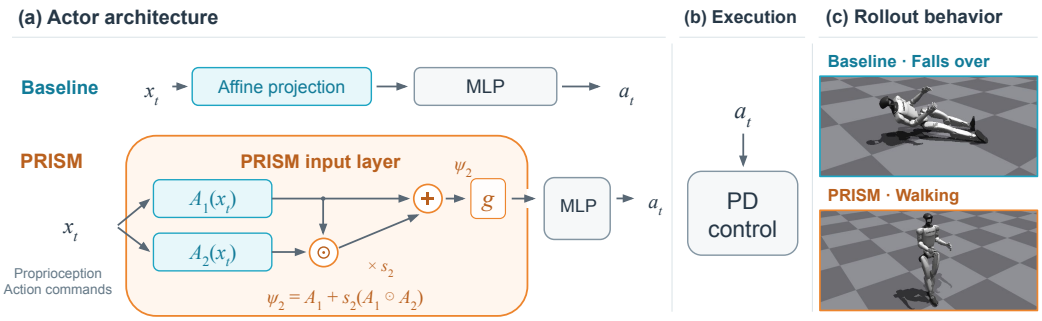


Figure 3: **PRISM replaces the initial affine projection with a factorized polynomial representation while retaining the downstream architecture.** (a) Actor architecture. The baseline uses an affine projection followed by an MLP. PRISM instead combines two learned affine factors, $A_1(x_t) = W_1 x_t + b_1$ and $A_2(x_t) = W_2 x_t + b_2$, to construct $\psi_2(x_t) = A_1(x_t) + s_2(A_1(x_t) \odot A_2(x_t))$. The projection g maps the resulting features to the downstream MLP, whose architecture remains unchanged. (b) Execution. Both policies output actions through the same low-level PD controller. (c) Rollout behavior. Rollouts show the baseline falling (top) and PRISM walking (bottom).

Policy inputs. Let $\pi_\theta(a_t \mid o_t)$ denote a policy that predicts action a_t from input o_t. We partition the input as $o_t = (x_t, c_t)$, where $x_t \in \mathbb{R}^d$ contains the state, command, and history inputs processed by PRISM, and c_t contains the remaining policy inputs.

- **Locomotion.** x_t contains the actor's proprioceptive observations together with commands and observation history. Proprioceptive signals describe the robot's configuration and motion, such as joint positions and velocities, base angular velocity, and projected gravity. Commands and previous actions, when included in the actor input, participate in the learned interactions alongside the observed state. The observation channels and temporal sampling follow the original policy.
- **Manipulation.** x_t contains the proprioceptive input or history, while c_t contains RGB observations and, when available, language instructions. PRISM processes the proprioceptive pathway; visual and language inputs retain their existing encoders and conditioning pathways. The resulting polynomial features are combined with these modalities for action prediction.

Factorized polynomial representation. For polynomial degree K, we introduce K affine feature layers, each with weight matrix W_k and bias b_k:

$$A_k(x_t) = W_k x_t + b_k, \qquad k = 1, \ldots, K. \tag{3}$$

PRISM constructs polynomial features by recursively adding scaled multiplicative interactions:

$$\text{Initialization:} \qquad \psi_1(x_t) = A_1(x_t), \tag{4}$$

$$\text{For } k = 2: \qquad \psi_2(x_t) = A_1(x_t) + s_2 \left(A_1(x_t) \odot A_2(x_t)\right), \tag{5}$$

$$\text{For } k \geq 2: \qquad \psi_k(x_t) = \psi_{k-1}(x_t) + s_k \left(\psi_{k-1}(x_t) \odot A_k(x_t)\right). \tag{6}$$

Here, $\odot$ denotes element-wise multiplication, and s_k scales each interaction term. By default, these scales are learned jointly with the policy.

- $k = 1$: ψ_1 is an affine feature, as in a standard MLP.
- $k = 2$: ψ_2 retains the affine path while adding products of two independently learned affine features, A_1 and A_2. Tying their weights and biases would restrict each product to the square of a single affine projection. Separate weights allow interactions between different learned input directions, capturing couplings across joints, observation channels, and history frames. These interactions are learned from the task objective rather than assigned predefined physical meanings.
- $k \geq 2$: In general, each step multiplies the preceding feature by a new affine feature and adds the result, extending the polynomial degree without enumerating the full monomial basis.

Setting scales $s_2, \ldots, s_K$ to zero gives $\psi_K = A_1$, reducing PRISM to a standard MLP. Thus, PRISM replaces only the first affine layer with ψ_K, leaving subsequent nonlinear operations g unchanged:

$$z_t = g(\psi_K(x_t)). \tag{7}$$

Multiplication precedes these nonlinear operations: ψ_K is polynomial in x_t, whereas z_t need not be. The policy combines z_t with existing features derived from c_t, when present.

PRISM for reinforcement learning. For reinforcement learning, PRISM replaces the state or history projection feeding the actor. The polynomial encoder constructs $\psi_K(x_t)$, and g maps it to the feature dimension expected by the remaining actor network. The action head retains its original parameterization, and predicted actions are executed through the existing low-level controller.

The polynomial encoder and actor are optimized jointly using the policy's native reinforcement-learning objective. The observation history, reward definition, and action interface are unchanged. When privileged observations are used for critic training, they remain restricted to value estimation and are not provided to the polynomial encoder or deployed actor.

PRISM for imitation learning. For imitation learning, PRISM replaces the proprioceptive conditioning projection while preserving its output dimension and downstream interface. Given the

proprioceptive input or history x_t, we compute $z_t^{\text{proprio}} = g(\psi_K(x_t))$. This representation is combined with the existing visual and task-conditioning features through the policy's original conditioning mechanism. The polynomial representation is optimized jointly with the policy's trainable components using its native imitation-learning objective. The visual and language pathways, action-generation process, and low-level controller are retained. Implementation-specific input configurations, projection networks, and training settings are provided in the Appendix.

4 Experimental Results

We ask four questions: **1)** Does PRISM improve locomotion and manipulation performance over standard and larger baselines? **2)** Does PRISM improve contact behavior and real-robot manipulation without additional sensing or controller changes? **3)** Does its advantage persist under out-of-distribution shifts in dynamics? **4)** What physical information can be extracted from its learned representations, and how do multiplicative features affect action prediction?

4.1 Task Performance and Capacity Controls

We evaluate whether PRISM improves policy performance across locomotion and manipulation backbones, and whether increasing baseline capacity reproduces these gains. Within each backbone comparison, methods share the observation and action interfaces, training data or environment, native learning objective, and low-level controller.

For locomotion, we evaluate velocity-command tracking on flat terrain in Humanoid-Gym [7] using MLP actors trained with PPO [33]. We report episode length, linear- and yaw-velocity tracking errors, and 24 s survival rate. We also evaluate BFM-Zero [34] on reference-motion tracking using Earth Mover's Distance (EMD). For manipulation, we evaluate task-specific ACT [35] on three tasks in ManiSkill3 [8], a physics-based manipulation benchmark, and multi-task SmolVLA [36] on LIBERO [9], a language-conditioned manipulation benchmark, reporting task success.

Performance across diverse policy backbones. In Humanoid-Gym and ManiSkill3 manipulation, PRISM improves the reported locomotion metrics and task success across all evaluated tasks, respectively (Tables 1 and 2). The benefit also extends to reference-motion tracking (Table 3) and multi-task SmolVLA (Table 4), with lower tracking EMD and higher mean LIBERO success.

Table 1: **PRISM improves humanoid locomotion beyond the tested increase in model capacity.** Performance comparison on Humanoid-Gym [7]. Results are mean ± standard deviation over five training seeds. The larger MLP matches the reported parameter count of PRISM. Survival rate denotes the percentage of episodes reaching 24 seconds. Bold indicates the best reported mean. At the same parameter count as MLP-L, PRISM increases survival from 52.25% to 92.50%, showing that the improvement is not explained by increased model capacity alone.

Method	Interaction	#Parameters	Episode Length ↑	Linear Error ↓	Yaw Error ↓	Survival Rate ↑
MLP	✗	0.926M	1340.8 ± 39	0.4607 ± 0.011	0.2817 ± 0.008	51.00 ± 3.7
MLP-L	✗	1.321M	1349.9 ± 29	0.4738 ± 0.028	0.3086 ± 0.010	52.25 ± 2.9
PRISM	✓	1.321M	$\mathbf{2233.4 \pm 23}$	$\mathbf{0.2099 \pm 0.008}$	$\mathbf{0.1221 \pm 0.005}$	$\mathbf{92.50 \pm 2.4}$

Table 2: **PRISM achieves higher success rates than ACT across all three manipulation tasks.** We compare ACT [35] and PRISM on three ManiSkill3 tasks [8], using CroSTAta recovery demonstrations [37]. Success rates (%) use the final 100K checkpoint, one training seed per task, and five evaluation seeds. Average is the mean success rate across the three tasks. PRISM achieves higher success on all three tasks, increasing the average from 63.5% to 67.6%. These gains under matched training conditions support the benefit of the representation across different manipulation tasks.

Method	StackCube	TwoRobotStackCube	G1TransportBox	Average
ACT [35]	58.8 ± 6.3	47.0 ± 3.2	84.8 ± 4.9	63.5
PRISM	$\mathbf{68.8 \pm 5.8}$	$\mathbf{48.2 \pm 2.8}$	$\mathbf{85.8 \pm 5.1}$	**67.6**

Table 3: **PRISM improves humanoid reference-motion tracking.** We compare BFM-Zero [34] and its variants at aligned checkpoints after 9.6M environment steps, using one training seed. Nominal uses dynamic friction coefficients of 1.0 and the original robot masses. Params are in millions; bold marks the lowest tracking EMD.

Method	#Parameters	Nominal EMD ↓
MLP	32M	1.104
MLP-L	49M	1.090
PRISM	32 M	**1.050**

Table 4: **PRISM improves mean success in multi-task manipulation.** We compare SmolVLA [36] and its variants on LIBERO at aligned 80K-update checkpoints. Success (%) is mean ± SD over three training seeds. The larger control replaces the linear state projection with a larger MLP. Params are in millions; Δ denotes percentage-point gains over Baseline. Bold marks the highest mean.

Method	#Parameters	LIBERO Average ↑	Δ
SmolVLA	450M	62.00 ± 1.67	–
Larger Model	456M	62.57 ± 3.14	+0.57
PRISM	451M	$\mathbf{63.82 \pm 2.59}$	**+1.82**

Comparison with increased baseline capacity. For Humanoid-Gym, BFM-Zero, and SmolVLA, we next test whether increasing baseline capacity reproduces these gains. The controls widen the Humanoid-Gym actor and BFM-Zero main network, and replace SmolVLA's linear state projection with a larger MLP. The Humanoid-Gym control matches the reported parameter count of PRISM, whereas the BFM-Zero and SmolVLA controls use more parameters. Nevertheless, these controls do not match the reported performance of PRISM (Tables 1, 3, and 4).

4.2 Practical Benefits for Robot Control

We examine the execution behavior of PRISM in contact-rich simulation and its practical benefits on a physical SO-101 [38] robot.

Policy representation versus execution-time correction. We investigate whether execution-time contact corrections applied to a fixed baseline policy can achieve similar task performance to PRISM. We compare against two simulation adaptations of Minimalist Compliance Control (MCC) [32]. MCC-Sensorless applies force-proxy corrections to the translation commands of a frozen ACT policy, whereas MCC-Oracle uses simulator contact forces. **Fig. 4 compares contact-force** traces from a representative StackCube rollout in ManiSkill3, aligned to the first hand–cube contact. In this example, PRISM rapidly reduces contact force following contact and subsequently completes the task. Importantly, PRISM receives no force, wrench, or tactile feedback, and uses no explicit admittance controller. **Table 5 compares induced-drop recovery** on 44 shared validated failure states. PRISM achieves the highest observed recovery rate (52.3%), compared with 40.9% for ACT, 13.6% for MCC-Sensorless, and 34.1% for MCC-Oracle, which uses simulator-force feedback. PRISM also has the lowest mean capped recovery time (3.11 seconds) and terminal placement

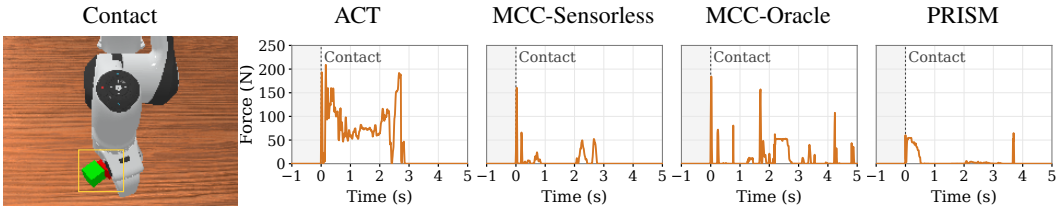


Figure 4: **PRISM exhibits compliance-like contact behavior in the illustrated recovery.** The leftmost image illustrates hand–cube contact during recovery from an induced object drop in StackCube [8]. The remaining panels compare hand–cube contact force (F, orange) for ACT [35], MCC-Sensorless [32], MCC-Oracle [32], and PRISM, respectively, from the same initial state. All force traces are aligned to first contact at $t = 0$ and use identical axes. In this example, PRISM rapidly relaxes contact force and first completes recovery 0.85 seconds after contact; the low-load phase includes object release. Execution continues for five seconds after contact, including after task completion. Contact force is recorded only for evaluation and is not provided to the PRISM actor.

Table 5: **PRISM improves failure recovery without explicit admittance control.**

Method	Recovery↑	Time(s)↓	Error(mm)↓
ACT [35]	40.9%	3.38	141.7
+MCC-Sensorless	13.6%	4.50	123.5
+MCC-Oracle	34.1%	3.71	105.4
PRISM	**52.3%**	**3.11**	**96.2**

Note. All methods use the same 44 validated drop states and final 100K checkpoints. Recovery is task completion rate. Time and placement error include failures; failed trials receive a capped recovery time of 5 s. MCC variants [32] are simulation adaptations.

Table 6: **PRISM improves real-robot SO-101 tape-pickup success.**

Method	Success↑	Motion(°)↑	Error(°)↓
ACT [35]	68.0%	31.1 ± 2.9	27.4 ± 2.9
PRISM	**94.0%**	$\mathbf{38.5 \pm 1.5}$	$\mathbf{24.3 \pm 4.2}$

Note. Both policies use the same 50 demonstrations and are evaluated over 50 hardware rollouts each. Joint motion measures movement during the first 3 seconds; larger values indicate more initial movement. Tracking error measures the difference between commanded and measured joint angles; lower values indicate closer tracking.

error (96.2 mm). These summaries include all 44 attempts; unrecovered cases are assigned the 5 seconds horizon for recovery time.

SO-101 tape pickup. To assess benefits on hardware, we compare ACT [35] and PRISM on a physical SO-101 robot using the same demonstrations and hardware protocol. The task requires lifting the tape roll at least 5 cm above the table and maintaining that height for 3 seconds. We report task success, arm joint-space travel during the first 3 seconds, and rollout-averaged command–state error. Table 6 shows higher pickup success for PRISM, accompanied by greater early joint-space travel and lower mean command–state error.

4.3 Generalizing to Out-of-distribution Physical Conditions

We test whether PRISM retains its performance advantage under changes in physical dynamics. All evaluations use fixed trained checkpoints without retraining. We report 60-second survival on flat terrain in Humanoid-Gym, 20-second supported survival on random rough terrain in mjlab [39], and reference-motion tracking Earth Mover's Distance (EMD) for BFM-Zero. Nominal dynamics provide the reference condition within each evaluation protocol.

Locomotion under dynamics changes. We compare PRISM with a parameter-matched MLP in Humanoid-Gym and with a standard MLP in mjlab. We vary friction, robot-link mass/inertia, and payload one factor at a time, including values outside their respective training ranges. PRISM achieves higher survival than a parameter-matched MLP on flat terrain in Humanoid-Gym and a standard MLP on random rough terrain in mjlab [39] across all displayed conditions (Fig. 5).

Across six settings beyond each simulator's training ranges, PRISM achieves mean survival of 92.3% versus 78.3% for a parameter-matched MLP on flat terrain in Humanoid-Gym, and 87.1% versus 80.0% for a standard MLP on random rough terrain in mjlab. In Humanoid-Gym, these settings span friction coefficients of 1.5–1.8, mass scaling factors of 0.8–1.2, and payloads of 3.6–4.8 kg. The corresponding ranges in mjlab are 0.2–1.5, 0.6–1.4, and 6–8 kg.

We evaluate the BFM-Zero policies under low friction and increased link mass, using the same checkpoints as in the nominal evaluation. Table 7 shows that PRISM retains lower tracking EMD than both the original and larger-capacity BFM-Zero backbones under both dynamics changes.

Robustness to manipulation failures. We evaluate whether PRISM recovers from induced object drops in ManiSkill3 StackCube [8]. Using ACT [35] and its PRISM variant trained on the same CroSTAta recovery demonstrations [37], we measure regrasp-and-lift and subsequent task completion within 100 control steps. PRISM achieves higher regrasp (75.0% vs. 68.2%) and recovery success (52.3% vs. 40.9%) than ACT. Further details are provided in Appendix B.1.

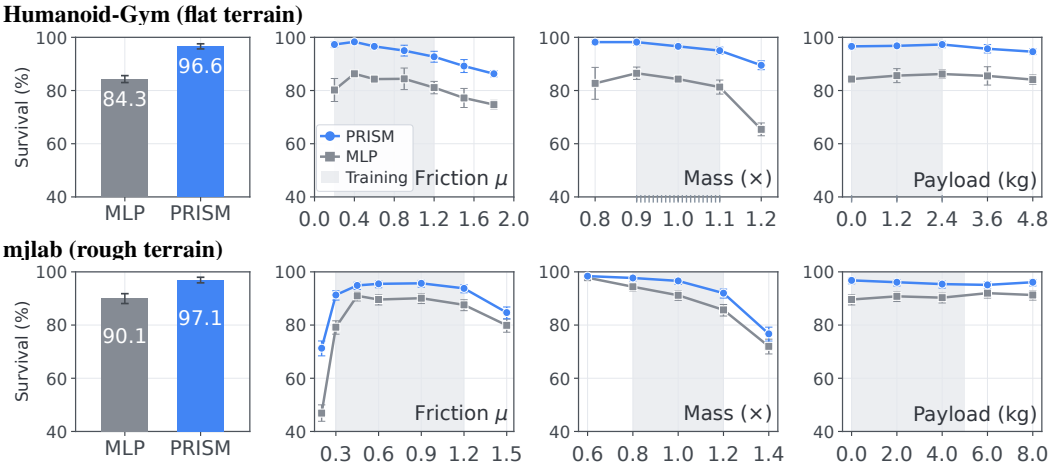


Figure 5: **Humanoid survival under dynamics changes.** Columns show survival under nominal dynamics and changes in friction, robot mass, and payload. Rows show flat terrain in Humanoid-Gym [7] (top) and random rough terrain in mjlab [39] (bottom). Shading marks training ranges or discrete-value envelopes. PRISM achieves higher survival than the MLP in all displayed conditions.

Table 7: **PRISM retains lower reference-motion tracking error under dynamics changes.** We report tracking EMD for BFM-Zero [34] and its variants at fixed checkpoints after 9.6M environment steps, using one training seed. Nominal uses terrain static and dynamic friction coefficients of 1.0 and the original link masses. Low-friction sets both coefficients to 0.20; Robot Mass scales link masses by 1.15. Each perturbation changes one factor from nominal. Mean averages the three conditions; bold marks the lowest EMD in each column. Both perturbations increase EMD for every model, but PRISM remains the lowest in each condition, showing that its tracking advantage persists beyond nominal dynamics.

Method	Nominal ↓	Low-friction ↓	Robot Mass ↓	Mean ↓
Baseline	1.104	1.582	1.121	1.269
Larger Model	1.090	1.589	1.114	1.264
PRISM	**1.050**	**1.548**	**1.073**	**1.224**

4.4 Learned Interactions and Action Prediction

We examine the information accessible from learned representations, the influence of multiplicative features on action prediction, and the impact of polynomial degree on policy performance.

Physical-response probing. We fit linear probes on frozen policy representations and evaluate predictions on held-out rollout windows with horizon $H = 5$. The targets include slip, joint power, contact impulse, and a contact-work proxy, computed from simulator states, contacts, and actions only for analysis. We report mean squared error (MSE) and Pearson correlation coefficient (PCC). Table 8 shows lower MSE and higher PCC for PRISM across the evaluated response proxies. The contact probes complement the contact-behavior analysis in Section 4.2 by assessing the linear accessibility of force-related response information.

Action sensitivity to multiplicative features. Linear predictability alone does not establish feature use in action prediction. We therefore remove individual latent products from the frozen policies while retaining the observations, affine path, learned weights, and other products. Table 9 highlights two structures: temporal and cross-joint factor dependence. In humanoid tracking, the leading factors emphasize left-hip velocity across consecutive observations and left–right hip velocities; removing their products changes predicted actions by 0.057 and 0.025, respectively. In TwoRobotStackCube, the leading factors span the two arms and an arm–gripper history, with corresponding

Table 8: **PRISM representations make mechanics-inspired responses more linearly predictable.** We conduct linear probing by freezing trained policy representations, fitting linear readouts, and evaluating them on held-out windows with horizon $H = 5$. Targets use simulator measurements only for analysis. Here, t is the window start and h indexes future samples; $\mathcal{F}$ contains the two feet, $c_{f,t}$ is one when foot f has vertical contact force above 5 N and zero otherwise, and $v^{\text{world}}_{f,xy,t}$ is its world-frame planar velocity. The symbols $\tau_{j,t}$ and $\dot{q}_{j,t}$ denote joint-j torque and angular velocity. For the contact targets, F_t is the scalar contact-load signal and Δx_t the spatial-displacement vector used to construct the proxy; norms are Euclidean and $\log$ denotes the natural logarithm. The contact rows retain the reported discrete proxy formulas, not calibrated continuous-time impulse or work estimates. MSE is mean squared error; PCC is the Pearson correlation coefficient. Lower MSE and higher PCC across all four targets indicate that mechanics-inspired response information is more linearly accessible in PRISM representations.

		MSE ↓		PCC ↑		Relative Gain	
Target	Formula	MLP	PRISM	MLP	PRISM	MSE	PCC
Slip velocity	$\sum_{h=1}^{H} \sum_{f\in\mathcal{F}} c_{f,t+h} \lVert v^{\text{world}}_{f,xy,t+h} \rVert_2$	0.711	**0.700**	0.501	**0.549**	**−1.5%**	**+9.6%**
Joint power	$\log\Big(1 + \sum_{h=1}^{H} \sum_{j=1}^{12} \lvert \tau_{j,t+h} \dot{q}_{j,t+h} \rvert\Big)$	0.328	**0.282**	0.820	**0.848**	**−14.0%**	**+3.4%**
Contact impulse	$\log(1 + \sum_{h\le H} F_{t+h})$	0.301	**0.241**	0.829	**0.861**	**−19.9%**	**+3.9%**
Contact work	$\sum_{h\le H} F_{t+h} \lVert \Delta x_{t+h} \rVert$	0.362	**0.319**	0.755	**0.780**	**−11.9%**	**+3.3%**

Table 9: **PRISM uses temporal and cross-joint features in action prediction.** We measure action changes after removing one latent product. For Humanoid-Gym, Δa is the mean L_2 distance between joint-target vectors after native action scaling, reported in radians. For ACT, it is the coordinate-wise mean absolute change in the first predicted normalized action. The two statistics have different units and are not directly comparable across domains. Products summarize dominant inputs to dense affine factors. With the observations and affine paths unchanged, removing these products alters actions in both tasks, showing that the learned multiplicative features contribute to action prediction.

Task Context	Top-ranked Interaction	Interaction Type	Δa
Humanoid velocity tracking	$\dot{q}_{\text{Lhippitch}}(t) \times \dot{q}_{\text{Lhippitch}}(t-1)$	Velocity–history	0.057
	$\dot{q}_{\text{Rhippitch}}(t) \times \dot{q}_{\text{Lhippitch}}(t)$	Cross-joint velocity	0.025
TwoRobotStackCube	$q_{\text{L5}}(t) \times q_{\text{R7}}(t)$	Cross-arm position	0.00338
	$q_{\text{L5}}(t-1) \times g_{\text{R2}}(t)$	Arm–gripper history	0.00270

action changes of 0.00338 and 0.00270. These interventions show that learned multiplicative features influence action generation in both domains. The displayed input products summarize dense factors, rather than identified physical mechanisms.

Polynomial degree. We compare affine, quadratic, and cubic representations under the same Humanoid-Gym setup (Appendix Table 21). Quadratic interactions capture most of the observed performance gain over the affine variant, while the cubic variant improves further. We use the quadratic representation as the default for its simpler construction.

5 Conclusion

PRISM shows that better robot control is not simply a matter of larger networks. Its compact polynomial layer learns multiplicative interactions among existing observations without prescribing robot dynamics or changing training objectives, sensing, or controllers. Across locomotion and manipulation, it improves performance, including on hardware, and retains its advantage under changes in friction, mass, and payload beyond the training ranges. These results show how changing the representation of observations can improve robot policies without redesigning the control system.

Limitations. PRISM relies on available observations to learn useful interactions; its benefits may be limited when essential information is missing.

Acknowledgements

This project was supported in part by NSF grant 2313151, with additional computing resources provided through NAIRR Pilot CIS240421 and CIS240431. We thank WHATs LAB in South Korea for generously lending robots that enabled our real-world manipulation experiments there.

References

[1] K. M. Lynch and F. C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[2] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[3] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations*, 2016.

[4] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[5] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[6] D. Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94: 103–114, 2017. doi:10.1016/j.neunet.2017.07.002.

[7] X. Gu, Y.-J. Wang, and J. Chen. Humanoid-gym: Reinforcement learning for humanoid robot with zero-shot sim2real transfer. *arXiv preprint arXiv:2404.05695*, 2024.

[8] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T.-K. Chan, et al. Maniskill3: Gpu parallelized robot simulation and rendering for generalizable embodied ai. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025.

[9] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[10] S. M. Jayakumar, W. M. Czarnecki, J. Menick, J. Schwarz, J. Rae, S. Osindero, Y. W. Teh, T. Harley, and R. Pascanu. Multiplicative interactions and where to find them. In *International conference on learning representations*, 2020.

[11] G. G. Chrysos, S. Moschoglou, G. Bouritsas, J. Deng, Y. Panagakis, and S. Zafeiriou. Deep polynomial neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 44(8):4021–4034, 2021.

[12] K. Usevich, R. Borsoi, C. Dérand, and M. Clausel. Identifiability of deep polynomial neural networks. *Advances in Neural Information Processing Systems*, 38:81809–81858, 2025.

[13] S. Maharaj and P. B. Nair. Deep learning with learnable product-structured activations. In *The Fourteenth International Conference on Learning Representations*.

[14] T. Xiao, W. Zhang, Y. Cheng, and J. Suo. Hope: High-order polynomial expansion of black-box neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46 (12):7924–7939, 2024.

[15] H. Li, Y. Chai, B. Lv, L. Ruan, H. Zhao, Y. Zhao, and J. Luo. Physics-informed neural network predictive control for quadruped locomotion. *arXiv preprint arXiv:2503.06995*, 2025.

[16] H. Hu, L. Feng, S. Chen, T. Zheng, D. Jiang, W. Chen, C. Zhang, G. Yang, and Y. Jin. Quietwalk: Physics-informed reinforcement learning for ground reaction force-aware humanoid locomotion under diverse footwear. *arXiv preprint arXiv:2604.23702*, 2026.

[17] C. Guo, G. L'Erario, G. R. M. Leonori, M. Lorenzini, A. Ajoudani, and D. Pucci. Physics-informed learning for human whole-body kinematics prediction via sparse imus. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4472–4479. IEEE, 2025.

[18] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[19] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[20] C. Zhang, N. Rudin, D. Hoeller, and M. Hutter. Learning agile locomotion on risky terrains. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11864–11871. IEEE, 2024.

[21] P. Zhi, P. Li, J. Yin, B. Jia, and S. Huang. Learning a unified policy for position and force control in legged loco-manipulation. *arXiv preprint arXiv:2505.20829*, 2025.

[22] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[23] O. Azulay, Z. Xu, A. Scheffer, and S. X. Yu. Vigor: Visual goal-in-context inference for unified humanoid fall safety. *arXiv preprint arXiv:2602.16511*, 2026.

[24] K.-Y. Lin and S. X. Yu. Let humanoids hike! integrative skill development on complex trails. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22498–22507, 2025.

[25] X. Zhang, Z. Kou, C. Qin, M. Huang, E. Ristani, A. Kumar, L. Chen, K. He, A. Boularias, and L. Guan. Glove2hand: Synthesizing natural hand-object interaction from multi-modal sensing gloves. *arXiv preprint arXiv:2603.20850*, 2026.

[26] Z. Liu, C. Chi, E. Cousineau, N. Kuppuswamy, B. Burchfiel, and S. Song. Maniwav: Learning robot manipulation from in-the-wild audio-visual data. *arXiv preprint arXiv:2406.19464*, 2024.

[27] R. Wang, H. Geng, T. Li, F. Wang, G. Anumanchipalli, T. Darrell, B. Li, P. Abbeel, J. Malik, and A. A. Efros. The sound of simulation: Learning multimodal sim-to-real robot policies with generative audio. *arXiv preprint arXiv:2507.02864*, 2025.

[28] X. Zhu, B. Huang, and Y. Li. Touch in the wild: Learning fine-grained manipulation with a portable visuo-tactile gripper. *Advances in Neural Information Processing Systems*, 38: 153783–153812, 2026.

[29] S. Jiang, H. Li, R. Ren, Y. Zhou, Z. Wang, and B. He. Kaiwu: A multimodal manipulation dataset and framework for robot learning and human-robot interaction. *IEEE Robotics and Automation Letters*, 2025.

[30] B. Xu, H. Weng, Q. Lu, Y. Gao, and H. Xu. Facet: Force-adaptive control via impedance reference tracking for legged robots. *arXiv preprint arXiv:2505.06883*, 2025.

[31] G. B. Margolis, M. Wang, N. Fey, and P. Agrawal. Softmimic: Learning compliant whole-body control from examples. *arXiv preprint arXiv:2510.17792*, 2025.

[32] H. Shi, S. Hu, Y. Hou, W. Wang, K. Liu, and S. Song. Minimalist compliance control. *arXiv preprint arXiv:2603.00913*, 2026.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[34] Y. Li, Z. Luo, T. Zhang, C. Dai, A. Kanervisto, A. Tirinzoni, H. Weng, K. Kitani, M. Guzek, A. Touati, et al. Bfm-zero: A promptable behavioral foundation model for humanoid control using unsupervised reinforcement learning. *arXiv preprint arXiv:2511.04131*, 2025.

[35] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[36] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[37] G. Minelli, G. Turrisi, V. Barasuol, and C. Semini. Crostata: Cross-state transition attention transformer for robotic manipulation. *arXiv preprint arXiv:2510.00726*, 2025.

[38] R. Knight, P. Kooijmans, R. Cadene, S. Alibert, M. Aractingi, D. Aubakirova, A. Zouitine, R. Martino, S. Palma, C. Pascal, et al. Standard open so-100 & so-101 arms. *URL https://github. com/TheRobotStudio/SO-ARM100*, 2025.

[39] K. Zakka, Q. Liao, B. Yi, L. L. Lay, K. Sreenath, and P. Abbeel. mjlab: A lightweight framework for gpu-accelerated robot learning. *arXiv preprint arXiv:2601.22074*, 2026.

[40] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[41] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[42] J. Kim, J. Kong, and J. Son. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International conference on machine learning*, pages 5530–5540. PMLR, 2021.

[43] B. Zhang and R. Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

[44] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal. Robust motion in-betweening. *ACM Transactions on Graphics (TOG)*, 39(4):60–1, 2020.

[45] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A Task Performance and Capacity Controls

This appendix follows the experimental organization of the main text: task performance and capacity controls, practical robot control, generalization to changed physical conditions, and representation analysis. We use the main-text partition $o_t = (x_t, c_t)$, where x_t is the observation stream processed by PRISM and c_t contains the remaining inputs. Each affine factor $A_k(x_t) = W_k x_t + b_k$ maps $\mathbb{R}^d$ to a common latent space $\mathbb{R}^m$, with $W_k \in \mathbb{R}^{m\times d}$ and $b_k \in \mathbb{R}^m$. The symbol g denotes the feature mapping between the polynomial block and the remaining policy, as illustrated in the main architecture figure. Its projection, normalization, and activation operations are specified for each backbone below. Scalar scales are broadcast over latent coordinates; channel-wise scales are written explicitly with $\odot$. The polynomial degree describes the feature before nonlinear operations, not the complete policy.

A.1 Humanoid-Gym locomotion

For the flat-terrain Unitree G1 comparison in Humanoid-Gym [7] (Table 1), Tables 10, 11, and 12 report the training settings, reward weights, and actor parameter counts. The actor input $x_t \in \mathbb{R}^{705}$ concatenates 15 frames of 47-dimensional proprioceptive and command observations.

Polynomial encoder and actor. Humanoid-Gym uses a residual parameterization of the affine-plus-product representation. To connect its implementation to the affine factors in the main text while retaining its actual projections, define

$$\ell_t = \mathcal{H}(x_t), \qquad p_{0,t} = \mathcal{B}_0(\ell_t), \tag{8}$$

$$A_1^{\mathrm{HG}}(x_t) = \mathcal{B}_1(\ell_t), \qquad A_2^{\mathrm{HG}}(x_t) = \mathcal{S}_0(p_{0,t}),$$

$$r_t = A_1^{\mathrm{HG}}(x_t) \odot A_2^{\mathrm{HG}}(x_t), \qquad \psi_2^{\mathrm{HG}}(x_t) = p_{0,t} + r_t, \tag{9}$$

$$z_t = \mathrm{ELU}\big(\mathcal{R}(x_t) + \beta_{\mathrm{poly}}\mathcal{P}(\psi_2^{\mathrm{HG}}(x_t))\big)\,. \tag{10}$$

All six calligraphic maps are learned affine maps, including biases. The maps $\mathcal{H}$ and $\mathcal{R}$ take 705 inputs to 256 outputs; $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{S}_0$, and $\mathcal{P}$ each map 256 features to 256 features. Thus both A_1^{HG} and A_2^{HG} are affine in x_t, and r_t contains products of learned affine observations. The superscript distinguishes this implementation from the compact construction: its affine base is $p_{0,t}$, and it additionally retains $\mathcal{R}(x_t)$. These paths and projections are included in the reported parameter counts, rather than identified with a bare pair of $705 \to 256$ factors.

The scalar β_{poly} increases from zero to one over 500 proximal policy optimization (PPO) [33] updates and scales the polynomial branch relative to the raw-input affine path. It is a scheduled buffer, not a learned interaction scale or the generalized advantage estimation (GAE) [40] parameter λ. At zero scale, the encoder becomes $z_t = \mathrm{ELU}(\mathcal{R}(x_t))$. The 256-dimensional output feeds the actor MLP $256 \to 512 \to 256 \to 128 \to 12$, with exponential linear unit (ELU) activations [3] after each hidden layer and no input or hidden LayerNorm. Its final projection predicts the mean of a Gaussian distribution over joint-position targets. The native action interface and low-level proportional–derivative (PD) controller are retained.

Training and performance measures. In Table 10, K_p and K_d are proportional and derivative gains, while v_x, v_y, and ω_z are forward, lateral, and yaw-rate commands. The gain pairs (K_p, K_d) are $(100, 2)$ for hip yaw, roll, and pitch, $(150, 4)$ for knees, and $(40, 2)$ for ankles, symmetrically on both legs. The discount factor is γ and the GAE parameter is λ. Episode length counts control steps, and survival is the percentage of episodes reaching the 24-second horizon. At control sample t, the velocity errors are

$$e_t^{\mathrm{lin}} = \|v_{xy,t}^{\mathrm{cmd}} - v_{xy,t}^{\mathrm{base}}\|_2, \qquad e_t^{\mathrm{yaw}} = |\omega_{z,t}^{\mathrm{cmd}} - \omega_{z,t}^{\mathrm{base}}|.$$

Here v_{xy} denotes planar velocity and ω_z denotes yaw rate; the errors are measured in m/s and rad/s, respectively. The evaluator averages sampled errors within episodes and then aggregates episodes; these are not squared-error or root-mean-square-error metrics. The mean and standard deviation in

Table 10: Humanoid-Gym [7] training configuration for Table 1. Ten physics steps per control tick give a control rate of 100 Hz.

Item	Setting
Actor observation	15×47 proprioceptive/command history
Critic observation	3×73 privileged history
Action	12-dimensional joint-position target, scale 0.25
Simulation	$\Delta t = 0.001$ s, control decimation 10
PD gains	$K_p = \{100, 150, 40\}$, $K_d = \{2, 4, 2\}$
Linear commands (m/s)	$v_x \in [-0.3, 0.6]$, $v_y \in [-0.3, 0.3]$
Yaw-rate command (rad/s)	$\omega_z \in [-0.3, 0.3]$
Randomization	Friction, mass, pushes, delay, action noise
Learning algorithm	PPO [33], learning rate 10^{-5}
Discount γ / GAE λ	0.994 / 0.9
Rollout / epochs / minibatches	60 / 2 / 4
Training iterations	3001
PRISM degree / latent width	2 / 256

Table 1 are across five training seeds, not individual evaluation episodes. Table 11 lists the nonzero native reward weights used for this comparison.

Table 11: Active reward weights for our Humanoid-Gym [7] experiments. Zero-weight terms are omitted; functional forms follow the released experiment configuration.

Term	Scale	Term	Scale
Linear velocity	1.2	Yaw velocity	1.1
Hard tracking	0.5	Low-speed regulation	0.2
Upright orientation	1.0	Base height	0.2
Velocity consistency	0.5	Base acceleration	0.2
Joint-pose tracking	1.6	Default joint pose	0.5
Feet air time	1.0	Feet clearance	1.0
Contact number	1.2	Feet distance	0.2
Knee distance	0.2	Collision	-1.0
Contact force	-0.01	Foot slip	-0.05
Action smoothness	-0.002	Torque	-10^{-5}
Joint velocity	-5×10^{-4}	Joint acceleration	-10^{-7}

Capacity controls. The baseline MLP uses hidden widths $[512, 256, 128]$; MLP-L uses $[816, 352, 160]$. Table 12 reports actor-only counts, separately from the main comparison's parameter totals. The exact actor counts are 527,244 for MLP, 922,092 for MLP-L, and 921,996 for PRISM, including affine biases. They follow from the complete encoder above and the corresponding actor MLPs, without counting the critic or Gaussian standard deviations. The main-table totals additionally include the shared critic $219 \to 768 \to 256 \to 128 \to 1$ (398,849 parameters) and 12 learned action standard deviations. The resulting totals are 926,105, 1,320,953, and 1,320,857, respectively, matching the displayed 0.926M, 1.321M, and 1.321M.

Table 12: Actor parameter counts for the Humanoid-Gym [7] comparison. All actors receive the same 705-dimensional observation history. Counts refer to the actor only and are rounded; MLP-L and PRISM match at the reported precision.

Method	Input dimension	Actor parameters
MLP	705	527K
MLP-L	705	922K
PRISM	705	922K

Fig. 6 supplements the aggregate results with learning curves and an illustrative rollout.

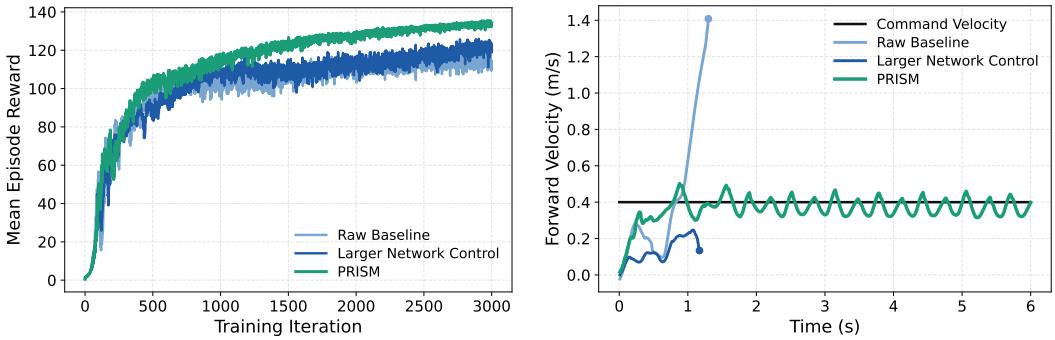


Figure 6: **PRISM improves learning performance and illustrated command tracking.** Left: Humanoid-Gym [7] training curves with shaded standard error across training seeds. The legend entries Raw Baseline and Larger Network Control denote MLP and MLP-L, respectively. Right: realized forward velocities under a fixed command (black). The displayed PRISM rollout remains near the commanded velocity, whereas the baseline traces end after falls, illustrating the tracking and survival differences summarized in Table 1.

A.2 ACT on ManiSkill3 recovery demonstrations

We train ACT [35] and its PRISM variant on three ManiSkill3 tasks [8], using the released CroSTAta recovery demonstrations [37]. We use the demonstrations, not the CroSTAta policy model. Each task has its own pair of policies, trained with batch size 64 and seed 1000 for 100K updates. We evaluate the final 100K exponential-moving-average (EMA) weights, without checkpoint selection on evaluation results. Table 13 specifies the data splits and native sensor/action interfaces.

Table 13: Task-specific ACT [35] data and native ManiSkill3 [8] interfaces. Train and Dev. give trajectory counts; Qpos and Action give the measured-joint and action dimensions for one time step. G1TransportBox uses the native `UnitreeG1TransportBox-v1` task.

Task	Train	Dev.	Qpos	Action	RGB views
StackCube	900	110	9	8	Base + wrist
TwoRobotStackCube	900	143	18	16	Base
G1TransportBox	900	113	25	25	Base + head

Sensor inputs and supervision. The actor receives two chronological frames of RGB images and measured robot joint positions. At reset, the first frame is duplicated. Native RGB is resized to 128×128 with bilinear interpolation and normalized with the pretrained ResNet18 [41] image statistics. Joint histories are concatenated to dimensions 18, 36, and 50 for StackCube, TwoRobotStackCube, and G1TransportBox. Joint and action normalizers are fitted only on the 900 training trajectories; action statistics use expert-valid labels. Object poses, relative object positions, task state, contact flags, forces, and privileged simulator state are excluded from actor inputs. The expert mask excludes artificial failure actions from supervised targets while retaining failure observations in histories. Invalid and terminal-padding targets are zeroed before the conditional variational autoencoder (CVAE) [42] action embedding and masked from posterior attention. The L_1 loss averages only expert-valid action components; the Kullback–Leibler (KL) loss averages only samples with at least one valid expert target.

Backbone and execution. Both methods use the same pretrained ResNet18 visual backbone and the transformer architecture used by ACT [35], with hidden width 256 and feedforward width 512, two encoder layers, four decoder layers, eight attention heads, and a 32-dimensional CVAE latent. Training predicts 16 native action vectors beginning at the current observation time. During evaluation, we execute the first predicted action and infer again at each 20-Hz control tick, without temporal ensembling. This shared execution schedule was fixed after a separate execution diagnos-

tic and before evaluation on the fresh seed bank. Inference uses a zero prior latent with the policy in evaluation mode. Actions retain the native normalized `pd_joint_delta_pos` controller interface. For TwoRobotStackCube, the flat 16-dimensional action preserves the left/right eight-component Panda blocks; for G1TransportBox, all 25 components control native joint deltas.

Proprioceptive conditioner. In the main-text notation, $x_t \in \mathbb{R}^d$ is the normalized two-frame joint history, with $d = 18$, 36, and 50 for StackCube, TwoRobotStackCube, and G1TransportBox, respectively. The baseline uses an affine projection followed by root mean square normalization (RMSNorm) [43]. For PRISM, the degree-2 construction is

$$A_i(x_t) = W_i x_t + b_i, \qquad i \in \{1, 2\}, \tag{11}$$

$$\psi_2(x_t) = A_1(x_t) + s_2\big(A_1(x_t) \odot A_2(x_t)\big), \tag{12}$$

$$z_t = g(\psi_2(x_t)) = \mathrm{RMSNorm}(W_p \psi_2(x_t) + b_p)\,. \tag{13}$$

Here, $\odot$ denotes element-wise multiplication, $W_i \in \mathbb{R}^{256\times d}$, $b_i \in \mathbb{R}^{256}$, $W_p \in \mathbb{R}^{256\times 256}$, and $b_p \in \mathbb{R}^{256}$. Thus $A_i(x_t)$, $\psi_2(x_t)$, and z_t are 256-dimensional. Both conditioners use a trainable RMSNorm gain initialized to one and a numerical stabilizer $\epsilon = 10^{-6}$. The scalar interaction scale s_2 is learned and initialized to 0.01. At initialization, A_1 copies the baseline affine projection, $W_p = I_{256}$, and $b_p = 0$, where I_{256} is the identity matrix. With these initial parameters, setting $s_2 = 0$ would reproduce the baseline encoding; the actual initialization uses $s_2 = 0.01$. The visual, latent, transformer, and action-head initialization is shared across methods.

Table 14: Shared ACT [35] training and deployment settings.

Item	Setting
RGB size / observation history	128×128 / 2 frames
Hidden / feedforward width	256 / 512
Encoder / decoder / attention heads	2 / 4 / 8
CVAE latent / KL weight	32 / 10
Prediction queries / execution	16 / first action at each 20-Hz tick
Temporal ensembling	Disabled
Batch / training seed / updates	64 / 1000 / 100K
Optimizer	AdamW [**?**], FP32
Learning rate / backbone rate	10^{-4} / 10^{-5}
Weight decay	10^{-4}
Scheduler	StepLR at 66,666 updates, factor 0.1
Checkpoint	Final 100K EMA; configured power 0.75
Conditioner width / RMSNorm epsilon	256 / 10^{-6} for both methods

Native evaluation. Each method and task uses five evaluation groups (2200–2204) with 100 attempts each. Group index $r \in \{0, \ldots, 4\}$ uses reset seeds $1200000 + 1000r + i$, where $i \in \{0, \ldots, 99\}$; its displayed group ID is $2200 + r$. The seed bank was fixed before the reported evaluation outcomes. Both methods share seeded native initial states and geometry signatures. Success is native `info.success` at any executed tick within 100 control steps (5 seconds); every scheduled attempt, including a runtime error, remains in the denominator. The success rates in Table 2 are mean $\pm$ sample standard deviation (SD) across these five groups, using one trained checkpoint per method and task. These SDs do not measure training-seed variability. Training-geometry disjointness is not asserted because there is no verified mapping from evaluation reset IDs to the demonstration geometry.

Evaluation uses ManiSkill [8] version 3.0.0b21, with single-environment CPU PhysX and a Vulkan renderer bound to the allocated NVIDIA GPU. The renderer is initialized before policy CUDA. Both methods use this same backend, sensor preprocessing, and controller. This compatibility path differs from CUDA-physics execution and is not a numerical reproduction of an author-reported benchmark score.

Table 15: **PRISM improves mean StackCube success under the shared evaluation protocol.** ACT [35] and its PRISM variant use the final 100K EMA checkpoint; each group contains 100 native ManiSkill3 [8] attempts. Mean and SD are computed across evaluation groups, not training seeds. The mean rises from 58.8% to 68.8% with the same reset-seed bank.

Method	2200	2201	2202	2203	2204	Mean $\pm$ SD
ACT	62	61	50	55	66	58.8 ± 6.3
PRISM	79	64	67	67	67	$\mathbf{68.8 \pm 5.8}$

A.3 BFM-Zero and SmolVLA

Table 16 specifies the architecture and evaluation settings for the reference-motion and multi-task manipulation comparisons. The BFM-Zero [34] capacity control widens the six-layer main network from 2048 to 2560 units; the SmolVLA [36] control replaces the linear state projection with a three-layer MLP of width 2048. Tables 3 and 4 count deployment-time BFM-Zero modules and the full SmolVLA policy, respectively.

BFM-Zero reference-motion tracking. The history-state encoder uses the degree-2 representation, a two-layer projection, and RMSNorm [43], while retaining the original core network [34]. These post-polynomial operations constitute the backbone-specific mapping g. All variants are trained from scratch with the native replay setup and motion data. Nominal evaluation disables dynamics randomization, sets static and dynamic terrain friction to 1.0, and retains the original link masses. We use the upstream tracking Earth Mover's Distance (EMD) evaluator on the LAFAN motion set [44]. The nominal and changed-dynamics comparisons use the same checkpoints after 9.6M environment steps, with one training seed.

BFM-Zero encoder configuration. The encoder configuration uses the 29-degree-of-freedom G1 robot. The observation configuration is `humanoidverse/config/obs/bfm_zero_obs.yaml`. Its history stream stacks four frames of base angular velocity (3), projected gravity (3), joint positions (29), joint velocities (29), and previous actions (29), giving $4 \times 93 = 372$ inputs. It specifies two 372-dimensional affine factors and an *element-wise* interaction scale:

$$\psi_2(x_t) = A_1(x_t) + s_2 \odot A_1(x_t) \odot A_2(x_t), \qquad s_2 \in \mathbb{R}^{372}.$$

Each component of s_2 is learned and initialized to 0.01. The backbone-specific mapping g is a $372 \to 372 \to 372$ projection with final RMSNorm [43] ($\epsilon = 10^{-6}$). The remaining observation streams retain their original interfaces. The scenario driver invokes `humanoidverse.tracking_eval` with the 29-degree-of-freedom LAFAN data, 128 evaluation environments, and the fixed-friction/mass overrides for each scenario. These dimensions describe the encoder configuration; the reported training budget and evaluation conditions are summarized in Table 16.

SmolVLA. We train on 40 LIBERO [9] tasks using the `HuggingFaceVLA/libero` release associated with SmolVLA [36]. The 16-layer visual-language backbone is initialized from SmolVLM2-500M-Video-Instruct and frozen; the action expert and proprioceptive conditioner are trained from scratch. The polynomial conditioner uses RMSNorm [43] within g. Training uses AdamW [45] with learning rate 10^{-4}, 100 warm-up steps, and cosine decay over a 100K-update schedule. The main comparison uses aligned 80K-update checkpoints and reports mean $\pm$ SD over three training seeds. Table 17 instead gives a single-seed suite breakdown; it is not the three-seed summary in Table 4.

B Practical Benefits for Robot Control

B.1 Controlled failure recovery on StackCube

Table 5 evaluates recovery from 44 shared physically induced drop states in ManiSkill3 StackCube [8]. ACT [35] and PRISM use the same CroSTAta recovery demonstrations [37] and the final

Table 16: BFM-Zero [34] and SmolVLA [36] implementation settings. The SmolVLA suite breakdown is a single run, distinct from the main three-seed summary.

Item	BFM-Zero	SmolVLA
Data / benchmark	40 LAFAN motions [44]	40 LIBERO tasks [9]
PRISM input	History-state stream	Proprioception
Baseline architecture	Main network: width 2048, six layers	Linear state projection
Larger control	Main network: width 2560, six layers	State MLP: width 2048, three layers
Reported checkpoint	9.6M environment steps	80K updates
Batching	512 parallel environments	Batch size 64
Training seeds	Single seed: 1000	Main: three seeds; breakdown: 1000
Evaluation	40 motions, 128 environments	50 rollouts/task, 2000 total

Table 17: **PRISM achieves the highest mean success in the single-seed SmolVLA breakdown.** SmolVLA [36] variants are evaluated on LIBERO [9] with seed 1000 and the protocol in Table 16. Avg. averages the four suites; bold marks each column's highest value. PRISM leads on three suites and the overall average, while the larger model leads on Object.

Method	Spatial	Object	Goal	Long	Avg.
SmolVLA	69.8	57.2	81.2	45.8	63.50
Larger Model	64.6	**65.6**	85.0	44.4	64.90
PRISM	**70.0**	57.4	**85.4**	**53.4**	**66.55**

100K checkpoints described in Sec. A.2. This controlled-drop pilot is distinct from ordinary task success and from an official CroSTAta recovery evaluation protocol.

Failure-state construction. A fixed baseline ACT checkpoint generates the setup trajectories. When cubeA is grasped, lifted at least 3 cm above its resting height, and at least 8 cm from cubeB in the horizontal plane for three consecutive control ticks, we hold the arm and open the gripper. A valid failure state is unfinished, ungrasped, settled on the table, and inside the workspace. We take the first 44 valid states from a fixed candidate order, without selecting by either policy's recovery outcome.

Each ACT/PRISM evaluation obtains its failure state by replaying the same recorded setup actions from the same native reset, rather than restoring a simulator snapshot. We verify the physical state, controller targets, counters, and two-frame sensor history before policy execution. Floating physical quantities use absolute tolerance 10^{-6}; recorded actions, integer counters, and RGB history are compared exactly. Object positions and forces are not provided to the ACT or PRISM actor.

Recovery criteria and aggregation. After handoff, each policy has 100 control ticks (5 seconds) to recover, with its history initialized from the last two setup observations and prediction at every 20-Hz tick. Recovery success is native StackCube completion after the drop. Regrasp-and-lift requires grasping cubeA and lifting it by at least 3 cm for three consecutive ticks; this is reported separately and is not counted as task completion. Every scheduled state remains in the denominator, including handoff mismatches and runtime errors.

The time entry in Table 5 is averaged over all 44 attempts. Successful attempts contribute their time from handoff to first task completion; unsuccessful attempts contribute the 5-second cap. Terminal placement error is also averaged over all attempts, including failures. Thus the time metric incorporates failed recoveries rather than measuring speed only on successful trials. The pilot reports proportions, not five-seed standard deviations. Its baseline-generated failure distribution and small sample size limit the scope of the comparison. For StackCube, terminal placement error is

$$e_p = 1000 \left\| p_A - p_B - [0 \quad 0 \quad 0.04]^\top \right\|_2 \quad \text{(mm)},$$

where p_A and p_B are the three-dimensional world-frame centers of cubeA and cubeB at first completion or the 5-second cap. The 4-cm offset is twice the native cube half-size of 2 cm. This is

task-placement error, not end-effector command-tracking error. The native stacking goal does not define a target cube yaw; we therefore do not assign an orientation error to this goal.

Execution-time correction baselines. MCC-Sensorless and MCC-Oracle are simulation adaptations of Minimalist Compliance Control [32] applied to a frozen ACT policy. MCC-Sensorless uses force-proxy corrections; MCC-Oracle uses simulator contact-force measurements. Both are evaluated on the same 44 validated states. The translation correction acts on a Cartesian target derived from the native ACT joint target, and is mapped back to the native joint-action interface as described below. PRISM changes the learned representation rather than adding an execution-time admittance controller, and receives no force, wrench, or tactile feedback. The sensorless wrapper estimates external torque as the negative native PD-drive torque, rather than reading simulator contact force. The drive model is

$$\tau_{\mathrm{PD}} = \mathrm{clip}\big(K_p(q^{\mathrm{target}} - q) - K_d\dot{q},\, \pm\tau_{\mathrm{lim}}\big),$$

using the native controller gains and saturation limits. Here q, $\dot{q}$, and q^{target} are the measured arm-joint positions, velocities, and current PD-drive targets; K_p and K_d are the native proportional and derivative gains. The clipping operation uses the corresponding joint-torque limits τ_{lim} componentwise. The robot links have gravity disabled in this simulation controller. Five 100-Hz physics samples are averaged per 20-Hz control tick, and an exponential moving average uses new-sample weight 0.1. Let $\bar{\tau}$ be this filtered PD-drive torque vector and $J_p \in \mathbb{R}^{3\times d_{\mathrm{arm}}}$ the world-frame translational Jacobian for the d_{arm} controlled arm joints. The proxy force is

$$\hat{f} = (J_p J_p^\top + 10^{-3} I_3)^{-1} J_p(-\bar{\tau}),$$

where I_3 is the 3×3 identity matrix. Jacobians use central finite differences with a 10^{-3}-rad step. The oracle instead uses the previous control tick's mean net hand–environment contact impulse divided by the physics interval. This net force differs from the sum of contact magnitudes plotted in Fig. 4.

The translational correction uses virtual mass $m_v = 1\,\mathrm{kg}$, stiffness $K_v = 100\,\mathrm{N/m}$, damping $B_v = 20\,\mathrm{N\,s/m}$, and control interval $\Delta t_{\mathrm{ctrl}} = 0.05\,\mathrm{s}$:

$$\dot{\delta}_{k+1} = \dot{\delta}_k + \frac{\Delta t_{\mathrm{ctrl}}}{m_v}(f_k - K_v\delta_k - B_v\dot{\delta}_k),$$
$$\delta_{k+1} = \delta_k + \Delta t_{\mathrm{ctrl}}\dot{\delta}_{k+1}.$$

Here k indexes 20-Hz control ticks, $\delta_k \in \mathbb{R}^3$ is the world-frame translation offset in meters, and $\dot{\delta}_k$ is its velocity. The input f_k is the sensorless proxy or the oracle net force. Both states start at zero. The offset norm is capped at 3 cm, and outward radial velocity is removed when the cap is reached. The corrected Cartesian target uses forward kinematics [1] of the nominal ACT joint target plus δ_k, with its orientation unchanged. Five damped least-squares inverse-kinematics iterations use the six-dimensional pose error, damping 0.1, and native joint limits.

Let $D \in \mathbb{R}^{d_{\mathrm{arm}}\times d_{\mathrm{arm}}}$ be the diagonal matrix of native arm-joint delta scales, and let a^{ACT} be the arm portion of the normalized ACT action. The conversions are

$$q^{\mathrm{nom}} = q + Da^{\mathrm{ACT}}, \qquad a^{\mathrm{corr}} = D^{-1}(q^{\mathrm{corr}} - q),$$

followed by componentwise clipping to $[-1, 1]$. The vector q^{corr} is the corrected joint target returned by inverse kinematics; gripper commands are unchanged. With zero force and offset, the wrapper preserves the nominal action. These baselines are simulation adaptations, not reproductions of the original hardware telemetry or policies trained with force inputs.

B.2 Contact-aligned force measurements

Fig. 4 complements the aggregate recovery results with an illustrative hand–cube contact diagnostic. A contact image is shown on the left, followed by four force-only panels for ACT [35], the MCC-Sensorless and MCC-Oracle adaptations [32], and PRISM. All traces start from the same validated induced-drop state (failure-bank case 2) and use identical force and time axes.

Contact-load definition. Let k index physics samples with duration $\Delta t_k = 0.01\,\mathrm{s}$. Let $\mathcal{C}_k$ be the set of contact points between the hand and either task cube, and let $J_{k,c}$ be the contact-impulse vector at point $c \in \mathcal{C}_k$. The plotted scalar load is

$$F_k^{\mathrm{hand}} = \frac{1}{\Delta t_k} \sum_{c \in \mathcal{C}_k} \|J_{k,c}\|_2. \tag{14}$$

Impulses are measured in N s and the resulting load in N. Summing the impulse magnitudes retains opposing finger-force contributions rather than cancelling them in a resultant vector. For each method, $t = 0$ is the first post-handoff 100-Hz sample at which hand–cubeA force reaches 1 N. The alignment event uses cubeA, whereas the plotted load includes contacts with either task cube.

Table 18: Contact-recording protocol for the illustrated induced-drop recovery on ManiSkill3 StackCube [8] in Fig. 4. The common failure bank is used for aggregate recovery statistics; the figure is a single contact-aligned example, not a bank average.

Item	Setting
Recovery evaluation	44 shared validated drop states
Illustrated state	Failure-bank case 2
Plotted contact pair	Hand–cubeA and hand–cubeB
Physics / policy rate	100 Hz / 20 Hz
Force / impulse units	N / N s
Contact alignment	First post-handoff hand–cubeA force ≥ 1 N
Trace window	1 s before and 5 s after first contact
ACT / PRISM actor inputs	RGB images and measured joint positions
Oracle wrapper feedback	Simulator contact force

Visualization and interpretation. The curves show the measured one-second pre-contact prefix and five seconds of actual policy execution after contact, without smoothing, padding, or scene resets. Execution continues after first task completion only for this illustration. In contrast, each attempt in Table 5 ends at first completion or the 5-second horizon measured from handoff. The selected case was the first earlier ACT-failure/PRISM-recovery contrast with actual contact for both policies; it was not selected for low contact force. Replay agreement and image/trace provenance are retained with the figure.

In this example, PRISM rapidly relaxes contact force and first completes recovery 0.85 seconds after contact, without force input to the actor or an explicit admittance controller. This is consistent with the compliance-like behavior described in the main text. The low-load phase includes object release, and the continued rollout later manipulates the object again. The example does not isolate sustained force-induced yielding or estimate mechanical compliance from a force–displacement relation.

B.3 SO-101 tape pickup

Table 6 compares ACT [35] and PRISM on the physical SO-101 robot [38]. Both policies are trained on the same 50 demonstrations and evaluated in 50 hardware rollouts each, with the same hardware protocol. A rollout succeeds when the tape roll is lifted at least 5 cm above the table and remains at that height for 3 seconds. ACT completes 34 of 50 attempts and PRISM completes 47 of 50, corresponding to 68% and 94% success.

Execution measurements. Consistent with the main text, joint motion describes arm joint-space travel during the first 3 seconds of execution. Joint tracking error describes the rollout-averaged discrepancy between commanded and measured arm-joint angles. These summaries are reported in degrees; neither is an angular-speed measurement. More initial travel describes an execution pattern rather than a separate task-success criterion. The higher pickup success for PRISM is accompanied by lower reported mean tracking error, connecting improved physical task completion with closer command following without attributing success to motion magnitude alone.

Collection and reset workflow. The hardware collection utilities use a calibrated SO-101 leader to teleoperate a calibrated SO-101 follower. Both policy launchers read camera inputs and reset settings from a common experiment configuration. A collection manifest specifies object-placement conditions, repetitions, and shuffled order. Between recordings, the operator resets the tape roll to the next listed condition and returns the arm to a saved ready pose; the utility also supports automatic ready-pose motion. Autonomous evaluation begins from that pose and uses episodic execution with return to the initial position between attempts.

C Generalizing to Out-of-distribution Physical Conditions

The dynamics evaluations use fixed trained checkpoints without retraining. Humanoid-Gym and mjlab measure survival under their respective protocols; BFM-Zero measures reference-motion tracking EMD. These metrics and evaluation horizons are not pooled across backbones.

C.1 Humanoid-Gym dynamics generalization

The flat-terrain G1 study in Humanoid-Gym [7] (Fig. 5) uses a separate training and evaluation protocol from Table 1. It compares PRISM with a parameter-matched MLP and evaluates 60-second survival, rather than the 24-second survival in the main policy-performance table. Final checkpoints are evaluated without selection on the dynamics sweeps. This comparison therefore evaluates the representation together with its training recipe. Table 19 lists the settings.

Table 19: **Humanoid-Gym dynamics-generalization protocol.** Experiments use Humanoid-Gym [7]. Parameter pairs list matched MLP / PRISM; actor–critic totals include action-distribution parameters. Sensor-noise scales are ordered as joint position, joint velocity, angular velocity, and orientation, in rad, rad/s, rad/s, and rad, respectively. $\mathcal{U}[a, b]$ denotes a uniform distribution on $[a, b]$.

Setting	Training	Evaluation
PPO [33] budget	3,001 updates; 4,096 environments	Final checkpoint
Actor-mean parameters	922,092 / 921,996	Fixed
Actor–critic total parameters	1,320,953 / 1,320,857	Fixed
History / action dimension	15×47 / 12	Unchanged
Polynomial degree / warm-up	2 / 500 updates, $\beta_{\text{poly}} : 0 \to 1$	$\beta_{\text{poly}} = 1$
Effective friction μ	$\mathcal{U}[0.2, 1.2]$	$\{0.2, 0.4, 0.6, 0.9, 1.2, 1.5, 1.8\}$
Ground / shape friction	0.2 / $\mathcal{U}[0.2, 2.2]$	Both set to μ
Robot-mass multiplier	$0.90, 0.91, \ldots, 1.10$	$\{0.8, 0.9, 1.0, 1.1, 1.2\}$
Backpack mass (kg)	$\{0, 1.2, 2.4\}$	$\{0, 1.2, 2.4, 3.6, 4.8\}$
XY replacement bound b (m/s)	0.2	$\{0, 0.1, 0.2, 0.4, 0.6\}$
Angular replacement bound (rad/s)	0.4	Disabled
Replacement interval	4 s	4 s for $b > 0$
Sensor-noise SD	$(0.03, 0.30, 0.06, 0.018)$	Disabled
Previous-action blend weight	$\mathcal{U}[0, 0.5]$	Disabled
Multiplicative action-noise SD	0.02	Disabled
Forward command (m/s)	$\mathcal{U}[-0.3, 0.6]$	Unchanged
Lateral command (m/s)	$\mathcal{U}[-0.3, 0.3]$	Unchanged
Target heading (rad)	$\mathcal{U}[-3.14, 3.14]$	Unchanged
Heading gain / yaw-rate clip	0.5 / $[-1, 1]$ rad/s	Unchanged
Planar-command dead zone	Norm ≤ 0.2 m/s	Unchanged
Command resampling	8 s	8 s
Episode horizon	24 s	60 s
Training / evaluation seeds	1 training seed (seed 1)	5 evaluation seeds (301–305)
Episode quota	—	2 per environment; 100 environments per seed

Training distribution. The simulator combines contact friction as $\mu_{\text{eff}} = (\mu_{\text{ground}} + \mu_{\text{robot}})/2$, where the subscripts denote effective, ground, and robot-shape friction. Discrete robot-mass and backpack settings are assigned through balanced, shuffled Cartesian-product combinations shared across methods. Link scaling changes the original robot links' masses and inertias, excluding the separately attached backpack, which has its own mass, center of mass, and inertia. Sensor noise is applied before history stacking, excluding command and previous-action channels. The action-blending weight is resampled at reset. The scheduled branch scale β_{poly} in Eq. 10 increases from zero to one over 500 updates and is fixed to one during evaluation. This schedule is distinct from the learned ACT interaction scale s_2.

Evaluation sweeps. Each sweep changes one factor from nominal effective friction $\mu = 0.6$, original link mass, no backpack, and no root-velocity perturbation. The nominal evaluation is reused across panels. Out-of-range settings refer to the individually varied training supports; a training boundary is not an unseen value. The six out-of-range conditions summarized in the main text comprise friction values $\{1.5, 1.8\}$, link-mass multipliers $\{0.8, 1.2\}$, and backpack masses $\{3.6, 4.8\}$ kg, varied one factor at a time.

The additional root-velocity sweep in the protocol replaces each world-frame XY velocity component with an independent draw from $\mathcal{U}[-b, b]$ on the native schedule, including reset-time scheduling; b is the replacement bound and $b = 0$ disables replacement. This is neither an applied force nor an additive velocity increment. Matching the training XY bound does not reproduce the full training distribution, since angular perturbations and observation/action noise are disabled during evaluation. Fig. 5 displays the friction, mass, and payload sweeps, not this additional velocity-replacement sweep.

Commands and survival. Yaw-rate commands are computed from heading error using the gain and clipping in Table 19, rather than sampled directly. Native resets and the initial zero-action step are retained. Survival uses the native timeout flag, including coincident timeout and physical-termination events, and does not impose a velocity-tracking threshold. The Humanoid-Gym panels report mean and sample SD over five evaluation seeds, not independently trained policies. Each seed uses 100 environments and two episodes per environment. Methods share evaluation settings and seeds, but their actions and reset times can differ.

C.2 mjlab rough-terrain survival

The lower row of Fig. 5 compares PRISM with a standard MLP in mjlab [39] on random rough terrain, reporting 20-second supported survival. Friction, robot mass, and payload are varied one factor at a time, using fixed trained checkpoints without retraining. This is a separate protocol from the 60-second flat-terrain Humanoid-Gym evaluation. Across the six out-of-range conditions summarized in the main text, mean survival is 87.1% for PRISM and 80.0% for the MLP. Those conditions span friction coefficients of 0.2–1.5, link-mass scaling factors of 0.6–1.4, and payloads of 6–8 kg.

Training configuration. Each method is trained for 30,000 PPO [33] updates with seed 1, 4,096 environments, and 24 control samples per update. Evaluation uses `model_29999.pt`. The physics interval is 0.005 s with decimation four, giving a 50-Hz policy and 20-second episodes. The actor and critic use ELU [3] MLPs with hidden widths $[512, 256, 128]$; the actor has RMSNorm [43] ($\epsilon = 10^{-6}$) after its final hidden activation and before its action projection for both methods. The actor receives 286 values: 99 robot/command values and 187 terrain-height samples; the critic receives 298 and there are 29 actions. In the main-text partition, $x_t \in \mathbb{R}^{99}$ is the robot/command stream and the terrain-height samples are part of c_t. Empirical observation normalization is shared across methods. PRISM transforms the 99 robot/command values with two $99 \to 256$ affine factors, a learned element-wise interaction scale $s_2 \in \mathbb{R}^{256}$ initialized componentwise to 0.01, and a $256 \to 99$ projection. The 187 height samples retain their original path into the actor. PPO uses Adam [45]

with initial learning rate 10^{-3}, adaptive KL target 0.01, clipping 0.2, five epochs, four minibatches, $\gamma = 0.99$, entropy weight 0.01, and maximum gradient norm 1.

Training uses the native mixed-terrain curriculum: flat terrain, ascending stairs, and descending stairs each have proportion 0.2; ascending slopes, descending slopes, random rough terrain, and waves each have proportion 0.1. Terrain patches are 8×8 m on a 10×20 grid, with a 20-m border and initial terrain level at most five. The generator's difficulty range is $[0, 1]$. The random-rough component has height-noise range $[0.02, 0.10]$ m, noise step 0.02 m, horizontal scale 0.1 m, and vertical scale 0.005 m. The figure evaluates random rough terrain only.

At startup, a shared nominal mask selects friction 0.6, mass/inertia scale 1, and zero payload with probability 0.2. Otherwise, friction, link-density scale, and payload are sampled independently from $\mathcal{U}[0.3, 1.2]$, $\mathcal{U}[0.8, 1.2]$, and $\mathcal{U}[0, 5]$ kg. Density scales original link mass and inertia together; the original robot mass is 33.341142 kg, excluding the payload. The payload is fixed to the torso at offset $(-0.15, 0, 0.20)$ m, with box dimensions $(0.20, 0.12, 0.16)$ m and no added collisions. Native center-of-mass, encoder, and push randomization remain active, including for the nominal-mask samples. The forward-command curriculum uses $[-1, 1]$ m/s initially, $[-1.5, 2]$ after 120K control steps, and $[-2, 3]$ after 240K.

Evaluation geometry and survival. Five paired layout seeds, 401–405, each provide 200 native resets, giving 1,000 attempts per method and condition. Episode seeds are $1000s + i$, with layout seed s and $i \in \{0, \ldots, 199\}$. Each layout uses a common certified 21×21 grid of 8×8-m patches (bounds ± 84 m), curriculum disabled, and fixed difficulty 0.5, retaining the native rough-terrain noise, platforms, and borders. Physical sweeps use command $(v_x, v_y, \omega_z) = (0.5, 0, 0)$ from rest, without a warm-up interval. Supported survival requires the full 20 seconds without native fall/termination and with both the root XY position and all 14 projected foot capsules inside the certified grid after every control step, including the final one. No velocity-tracking threshold is imposed, and failed attempts remain in the denominator. Error bars are pointwise Wilson 95% intervals [**?**] for the pooled success count out of 1,000, conditional on these fixed layouts and checkpoints; they are not training-seed standard deviations.

The six displayed out-of-range settings are friction 0.2 and 1.5, mass/inertia scales 0.6 and 1.4, and payloads 6 and 8 kg. The separately measured friction-0.1 condition is omitted from the displayed figure and this six-condition mean.

C.3 BFM-Zero dynamics changes

Table 7 uses the same 9.6M-step BFM-Zero [34] checkpoints and tracking EMD evaluator as the nominal comparison in Table 3. The low-friction condition sets both static and dynamic terrain friction coefficients to 0.20; the robot-mass condition multiplies link masses by 1.15. Each changes one factor from nominal. Link-mass scaling is not an attached payload. All other scenario and rollout settings are retained. Mean is the arithmetic average of the nominal, low-friction, and increased-mass EMD values. Both perturbations increase EMD for every model, but PRISM retains lower EMD than the original and larger-capacity baselines.

D Learned Interactions and Action Prediction

D.1 Physical-response probing

Table 8 evaluates whether mechanics-inspired response targets can be predicted linearly from frozen policy features. The four target names and formulas follow the main table: slip velocity, joint power, contact impulse, and contact work. Their simulator-derived targets are used only for analysis, not as policy inputs or auxiliary training objectives. Linear readouts are fitted on a fitting partition and evaluated on held-out windows with horizon $H = 5$.

Slip and joint-power targets. For Humanoid-Gym, $h \in \{1, \ldots, H\}$ indexes future control samples after the window start t. The 100-Hz control rate gives a sample interval of $0.01\,\mathrm{s}$. Let $\mathcal{F}$ contain the two feet, and define

$$c_{f,k} = \mathbf{1}\left[F^z_{f,k} > 5\,\mathrm{N}\right],$$

where $F^z_{f,k}$ is the vertical contact force for foot f at sample k, and $\mathbf{1}[\cdot]$ is the indicator function. The vector $v^{\mathrm{world}}_{f,xy,k} \in \mathbb{R}^2$ is that foot's planar world-frame velocity. With joint torque $\tau_{j,k}$ and joint angular velocity $\dot{q}_{j,k}$, the targets are

$$y^{\mathrm{slip}}_t = \sum_{h=1}^{H} \sum_{f \in \mathcal{F}} c_{f,t+h} \| v^{\mathrm{world}}_{f,xy,t+h} \|_2,$$

$$y^{\mathrm{power}}_t = \log\left(1 + \sum_{h=1}^{H} \sum_{j=1}^{12} |\tau_{j,t+h} \dot{q}_{j,t+h}|\right).$$

These are the contact-masked slip and per-joint absolute-power formulas in Table 8. Taking absolute values before summing prevents cancellation between positive and negative joint-power contributions. Neither expression contains a time-integration multiplier: they are sampled response proxies, not integrated displacement or energy.

Contact-response targets. The two contact rows retain the discrete expressions reported in Table 8:

$$y^{\mathrm{impulse}}_t = \log\left(1 + \sum_{h=1}^{H} F_{t+h}\right),$$

$$y^{\mathrm{work}}_t = \sum_{h=1}^{H} F_{t+h} \| \Delta x_{t+h} \|_2.$$

Here F_k denotes the scalar contact signal entering the target and Δx_k denotes the spatial displacement used in the contact-work proxy. In these expressions, Δx_k is not a difference between policy-input vectors x_t. The work target combines a contact scalar with a displacement magnitude rather than a signed force–displacement inner product. The impulse row is retained in its reported discrete form, without inserting an additional time-step multiplier. Consequently, these labels describe mechanics-inspired target proxies, not independently calibrated continuous-time impulse or work estimates. The locomotion collection and readout settings below apply to the slip and joint-power rows; they do not specify the contact-probe collection.

Locomotion feature collection. Both frozen locomotion policies are evaluated on a common observation bank generated by the PRISM policy. Collection disables observation noise, action noise, and action delay, while retaining friction and base-mass randomization and one-second pushes with XY bound $0.2\,\mathrm{m/s}$ and angular bound $0.4\,\mathrm{rad/s}$. A window can start only when root height exceeds $0.55\,\mathrm{m}$, absolute roll and pitch are below $0.35\,\mathrm{rad}$, episode age is at least ten control ticks, and no reset flag is present. Selected windows contain a future push with nonzero XY magnitude. The MLP readout uses the final 128-dimensional hidden activation; the PRISM readout uses the 256-dimensional encoder output z_t in Eq. 10.

Fitting and held-out evaluation. Locomotion windows are randomly partitioned 80/20 into fitting and evaluation subsets. Features and targets are standardized using only fitting-set means and standard deviations, with 10^{-6} added to each standard-deviation denominator. For standardized feature $\widetilde{\phi}_n \in \mathbb{R}^{d_\phi}$ and scalar target $\widetilde{y}_n$, the prediction is $\widehat{y}_n = w^\top \widetilde{\phi}_n + b$, where $w \in \mathbb{R}^{d_\phi}$ and $b \in \mathbb{R}$. The weights use ridge regularization and the intercept is unpenalized; the ridge strength and base split seed are run parameters. The split seed is $s + 100i$, where s is the base seed and i is the method index, so the two methods need not use identical held-out subsets. Splitting is at window level rather than trajectory level. This protocol therefore measures held-out-window prediction, not generalization to unseen trajectories or a matched-width/layer probe control.

Metrics and interpretation. The locomotion readouts report mean squared error (MSE) on standardized targets and Pearson correlation coefficient (PCC) on held-out windows. For the relative-change columns of Table 8, the reported percentages are

$$\Delta_{\mathrm{MSE}} = 100\frac{\mathrm{MSE}_{\mathrm{PRISM}} - \mathrm{MSE}_{\mathrm{base}}}{\mathrm{MSE}_{\mathrm{base}}}, \qquad \Delta_{\mathrm{PCC}} = 100\frac{\mathrm{PCC}_{\mathrm{PRISM}} - \mathrm{PCC}_{\mathrm{base}}}{\mathrm{PCC}_{\mathrm{base}}}.$$

Here base denotes the corresponding baseline column. Negative MSE changes and positive PCC changes indicate improvement. The main table reports both improvements for each of its four targets. This supports greater accessibility to the tested linear readouts; it does not by itself identify a physical mechanism or show that a policy uses the decoded quantity to generate actions. The diagnostic in Sec. D.2 addresses action dependence directly.

D.2 Action sensitivity to multiplicative features

Table 9 measures how predicted actions change when a complete latent product is removed from a frozen policy. The domain-specific statistics are the same as those in the main caption: mean joint-target L_2 distance for Humanoid-Gym [7] and coordinate-wise mean absolute change in the first normalized action for ACT [35].

Latent-product intervention. Let $r_t \in \mathbb{R}^m$ be the multiplicative contribution at the intervention location. Removing coordinate j gives

$$r_t^{(-j)} = r_t - [r_t]_j e_j, \tag{15}$$

where $[r_t]_j$ is the jth component and $e_j \in \mathbb{R}^m$ is the corresponding coordinate vector. For ACT, $r_t = s_2(A_1(x_t) \odot A_2(x_t))$ and $\psi_2^{(-j)}(x_t) = A_1(x_t) + r_t^{(-j)}$ is passed through the unchanged projection and RMSNorm in g. For Humanoid-Gym, Eq. 9 uses $r_t = A_1^{\mathrm{HG}}(x_t) \odot A_2^{\mathrm{HG}}(x_t)$ and $\psi_2^{\mathrm{HG},(-j)}(x_t) = p_{0,t} + r_t^{(-j)}$. Its affine base $p_{0,t}$, raw-input path $\mathcal{R}(x_t)$, projection $\mathcal{P}$, and remaining actor are retained. In both cases, the observations, learned weights, and all other latent products are fixed; the intervention removes one product coordinate, not its original input channels.

Humanoid-Gym statistic and evaluated candidates. Let $\widehat{a}_n$ and $\widehat{a}_n^{(-j)}$ be the original and intervened native action vectors for saved observation n. The diagnostic uses $N_{\mathrm{HG}} = 512$ observations and a seed-2 checkpoint. After the native 0.25-rad joint-target scaling, it reports

$$\Delta a_{\mathrm{HG}}^{(j)} = \frac{1}{N_{\mathrm{HG}}} \sum_{n=1}^{N_{\mathrm{HG}}} \left\| 0.25\big(\widehat{a}_n - \widehat{a}_n^{(-j)}\big) \right\|_2 .$$

The action-vector distance is in radians, not a coordinate-averaged absolute error. The evaluated candidate coordinates are 159, 194, 19, and 154. The displayed humanoid entries are selected within this candidate set; their ordering does not assert an exhaustive ranking of all 256 products. The main table reports action changes of 0.057 and 0.025 for its two displayed humanoid interactions.

ACT statistic and feature ranking. For each manipulation task, the analysis uses its final 100K EMA checkpoint with training seed 1000 and $N_{\mathrm{ACT}} = 128$ held-out RGB–joint-history windows. Four time-stratified observations are drawn from each of 32 development trajectories using sampling seed 20261016, without success-based filtering. Each of the 256 products is ablated separately before the fixed projection and RMSNorm. For action dimension d_a, the statistic is

$$\Delta a_{\mathrm{ACT}}^{(j)} = \frac{1}{N_{\mathrm{ACT}} d_a} \sum_{n=1}^{N_{\mathrm{ACT}}} \sum_{\ell=1}^{d_a} \left| \widehat{a}_{n,\ell} - \widehat{a}_{n,\ell}^{(-j)} \right| .$$

The action vectors are the first predicted normalized actions, before denormalization or clipping. Products are ranked within each task by this statistic. Table 9 displays the two highest-ranked TwoRobotStackCube products, with changes of 0.00338 and 0.00270; it does not display a pair for every analyzed task. The unablated and intervened outputs are computed with the same frozen policy operations. Larger values indicate greater output sensitivity, not better task performance.

Table 20: **Action-sensitivity statistics follow the domain-specific definitions in the main text.** The Humanoid-Gym [7] and ACT [35] diagnostics remove one complete latent product and retain the other policy operations. Humanoid-Gym measures joint-target vector distance in radians; ACT measures coordinate-wise absolute change in normalized action units. Their magnitudes are therefore not directly comparable across domains.

Item	Humanoid-Gym	ACT
Observation windows	512 saved observations	128 held-out windows per task
Training checkpoint	Seed 2	Final 100K EMA, seed 1000
Action statistic	Mean L_2 distance	Coordinate-wise mean absolute change
Action representation	After 0.25-rad target scaling	First normalized action, before clipping
Feature selection	Ranked within selected candidates	All 256 products ranked within task

Interaction labels. Each displayed product names dominant input coefficients in two dense affine factors; it is not an isolated raw-input monomial. For Humanoid-Gym, let $W_{\mathcal{H}}$, $W_{\mathcal{B}_0}$, $W_{\mathcal{B}_1}$, and $W_{\mathcal{S}_0}$ be the weight matrices of the corresponding affine maps. The effective raw-input factor matrices are $W_{\mathcal{B}_1}W_{\mathcal{H}}$ and $W_{\mathcal{S}_0}W_{\mathcal{B}_0}W_{\mathcal{H}}$. The label uses the largest-magnitude coefficient in the first factor and the largest coefficient in the second factor excluding the first selected input coordinate. For ACT, the label independently selects the largest-magnitude coefficient in each affine row on normalized joint histories; repeated coordinates are allowed.

The symbols $\dot{q}_{\mathrm{Lhippitch}}$ and $\dot{q}_{\mathrm{Rhippitch}}$ denote left and right hip-pitch velocities. In TwoRobotStackCube, q_{L5} and q_{R7} denote the measured angles of the fifth joint of the left seven-degree-of-freedom Panda arm and the seventh joint of the right arm. The symbol g_{R2} denotes measured displacement of the right gripper's second finger joint and is distinct from the feature mapping g. The index $t-1$ denotes the preceding observation in the input history. With the observations and affine paths unchanged, removing these products alters actions in both tasks, showing that the learned multiplicative features contribute to action prediction. This intervention does not isolate the physical role of a single raw-input product or establish a task-success benefit from it alone.

D.3 Polynomial degree

Table 21 compares affine (D1), quadratic (D2), and cubic (D3) constructions in the 256-wide Humanoid-Gym polynomial-actor family. As in the main method, degree is the maximum polynomial degree before the downstream nonlinear mapping, not the degree of the full policy. These are separate ablation runs, not the five-training-seed summary in Table 1.

Each degree increment adds two biased $256 \to 256$ affine maps, or 131,584 actor parameters. Adding the shared 398,849-parameter critic and 12 action standard deviations gives total counts of 1,189,273, 1,320,857, and 1,452,441 for D1, D2, and D3, respectively. Thus the comparison varies both degree and capacity. Most of the observed improvement occurs from D1 to D2, with a smaller additional improvement for D3; the quadratic construction remains the default for its simpler form.

Table 21: **Quadratic features capture most of the observed polynomial-degree gain.** D1, D2, and D3 denote affine, quadratic, and cubic feature constructions on Humanoid-Gym [7]. These runs are separate from the five-seed comparison in Table 1; bold marks the best reported value. Survival rises from 55.75% for D1 to 91.00% for D2, with a further increase to 92.50% for D3. Actor capacity also increases with degree, so this is not a parameter-matched isolation of polynomial degree.

Method	Actor params	Return ↑	Ep. Len. ↑	Lin. Err. ↓	Survival (%) ↑
PRISM-D1	790,412	75.26	1408.4	0.4685	55.75
PRISM-D2	921,996	131.78	2198.7	0.2254	91.00
PRISM-D3	1,053,580	**134.60**	**2296.5**	**0.2097**	**92.50**