PRISM: Polynomial Representations for Interaction-Structured Motor Control

Seung Hyun Lee
University of Michigan, Ann Arbor
Computer Science and Engineering

Stella X. Yu
University of Michigan, Ann Arbor
Computer Science and Engineering

Abstract: Robot policies are typically MLPs mapping observations to actions. Yet robot observations are physical variables, and many action-relevant cues arise not from individual variables but from their interactions; power, inertial effects, contact, slip, and compliance depend on products among observable signals. We introduce PRISM, a policy representation that makes polynomial interactions among observable physical variables explicit, learnable, and compact. Rather than listing all polynomial terms, PRISM uses a factorized polynomial module to expose higher-order interaction features efficiently. In reinforcement learning, it keeps the standard MLP backbone but applies a gradually activated element-wise polynomial function after it. In imitation learning, it replaces linear proprioceptive conditioning in Diffusion Policy with a polynomial layer trained end-to-end. Across humanoid locomotion and contact-rich manipulation, PRISM improves performance over standard MLP policies and larger MLPs with matched capacity, showing that interaction structure cannot be replaced by capacity alone. It also yields sensorless compliant behavior without force, wrench, tactile input, contact labels, or admittance control. These results suggest that polynomial representations should become a standard architectural choice for embodied motor control.

Keywords: Polynomial interaction, Motor control, Sensorless compliance

1 Introduction

Every learned robot policy must answer the same question: Given what the robot can observe, what action should it take next? In modern imitation and reinforcement learning (RL), this mapping is almost always implemented by an MLP inside the policy network. The choice is so standard that it is rarely questioned. In image recognition, the convention is natural: Deep visual features are passed through MLP layers to classify semantic categories. In robotics, however, the policy input is not an arbitrary feature vector. It is an observation of physical variables: joint positions, joint velocities, commands, action history, IMU/base orientation, actuator signals, RGB images, proprioception, and other quantities that the deployed robot senses in order to choose an action.

This distinction matters because many action-relevant physical quantities are not directly given as entries in this observation vector. The original entries are first-order variables: positions, velocities, commands, actuator signals, and histories. Control, however, often depends on their products or higher-order combinations. Joint power is the product of torque and joint velocity; kinetic, Coriolis, and centrifugal effects depend on velocity products; and slip, impact, contact impulse, and compliance can emerge from coupled relations among proprioception, commands, and recent actions (Fig. 1). These cues are thus not absent from the policy input. Rather, they are latent in polynomial interactions among observable variables. Standard MLP actors, however, receive only first-order variables, leaving these action-relevant physical quantities implicit rather than directly available.

A plain MLP can approximate such polynomial structure in principle [1, 2], but it does not present this structure to the policy directly. Its affine layers first form weighted sums of the original first-order coordinates, whereas many physical cues arise from second-order products or richer higher-

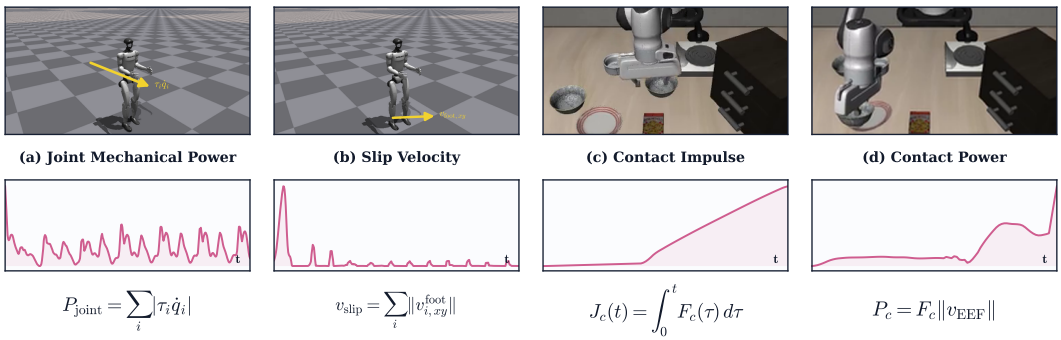

Figure 1: **Policies should expose multiplicative interactions among proprioception, actions, commands, and motion variables.** We show representative rollout signals from locomotion tasks and manipulation tasks; each curve plots the corresponding physical quantity over rollout time. These crucial physical quantities are not directly provided as policy inputs, but they can be formed through higher-order interactions of observable variables (a) $P_{\text{joint}} = \sum_i |\tau_i \dot{q}_i|$ denotes joint mechanical power, where τ_i and $\dot{q}_i$ are joint torque and velocity. (b) $v_{\text{slip}} = \sum_i \|v_{i,xy}^{\text{foot}}\|$ denotes foot slip velocity, where $v_{i,xy}^{\text{foot}}$ is planar foot velocity.(c) $J_c(t) = \int_0^t F_c(\tau) d\tau$ denotes accumulated contact impulse, integrating contact-force magnitude F_c over time. (d) $P_c = F_c \|v_{\text{EEF}}\|$ denotes contact power, where v_{EEF} is end-effector velocity.

order combinations. Increasing network width adds capacity, but it does not change this input basis: Torque–velocity, velocity–velocity, state–velocity, and action–state couplings remain implicit. Thus, if these interactions matter for control, simply enlarging the MLP is not equivalent to giving the policy an interaction basis. A standard actor must discover these relations internally from data rather than receive them in a form aligned with the underlying physics.

Our first insight is to make these interactions explicit and learnable. Rather than hand-engineering proxies for power, slip, contact, or motion coupling, we make polynomial interactions among observable physical variables directly available for action prediction. This gives the policy a learnable interaction basis in which such quantities can emerge from data. A degree-2 feature, for example, represents a learned quadratic interaction over the original inputs.

Our second insight is to make this interaction basis compact. A direct feature-lifting approach would append all coordinate products to the observation, but this quickly becomes inefficient as the input dimension and polynomial degree grow [3, 4]. PRISM instead uses a factorized polynomial module inspired by multiplicative neural interactions [5, 6, 7, 8]: Learned projections first form compact latent factors, whose element-wise polynomial products provide higher-order interaction features at manageable cost. In this view, the standard MLP actor is the first-order special case, while PRISM extends it with efficient learnable higher-order structure. In RL, PRISM keeps the MLP backbone but applies a gradually activated element-wise polynomial function after it, exposing higher-order interactions within the policy. In contact-rich imitation learning, PRISM replaces the standard linear proprioceptive conditioning layer in Diffusion Policy [9] with a polynomial interaction layer trained end-to-end with the policy.

We introduce PRISM, a policy representation that makes polynomial interactions among observable physical variables explicit, learnable, and compact. We evaluate PRISM on humanoid locomotion and contact-rich manipulation. In Humanoid-Gym [10], PRISM improves command tracking, episode length, and survival over both a standard MLP actor and a larger MLP with matched capacity, showing that interaction structure cannot be replaced by parameter count alone. In LIBERO [11], PRISM improves Diffusion Policy [9] success under the same RGB, proprioceptive, action, and low-level-control interface. It also yields compliance-like contact behavior without force, wrench, tactile input, contact labels, or an explicit admittance controller, outperforming Minimalist Compliance Control [12] in the deployable sensorless setting. Linear probing further shows that PRISM makes mechanics-inspired quantities such as slip, joint power, contact impulse, and contact work more accessible from policy representations.

We make three contributions. **1)** We identify a limitation of default MLP policies: They receive first-order physical variables, while many action-relevant cues arise from multiplicative interactions. **2)** We introduce PRISM, a compact polynomial policy representation that makes these interactions explicit and learnable without enumerating all monomials. It uses factorized element-wise polynomial interactions that can be gradually activated in reinforcement learning and can replace linear proprioceptive conditioning in Diffusion Policy. **3)** We show that this simple change improves performance without deployment-time sensing overhead, yielding broad gains in humanoid locomotion, contact-rich manipulation, and physical-quantity probing. These results suggest that **polynomial interaction modeling should become a standard architectural choice for embodied motor control.**

2 Related Work

Feature lifting for nonlinear representations. Feature lifting projects inputs into higher-dimensional spaces to make complex relationships linearly accessible [3, 4]. While deep MLPs are universal approximators [1, 2], they lack structural inductive biases for coordinate interactions, requiring excessive data to learn multiplicative mechanics. PRISM introduces a low-degree polynomial proprioceptive lift, directly exposing latent physical interactions to shallow control heads without the sample inefficiencies of deep networks.

Polynomial and multiplicative neural networks. Multiplicative operations enrich representable function classes [5], enabling factorized polynomial models to capture high-order correlations [6, 7, 8, 13]. Recent works have shown physics-informed locomotion [14, 15, 16] can guide policy learning by tracking complex physical properties. However, these paradigms typically require heavy auxiliary estimators or structured loss formulations at training time to enforce physical consistency. In contrast, PRISM embeds physical structure directly into the network architecture itself through a low-degree coordinate mapping. To our knowledge, PRISM is the first to introduce factorized polynomial lifts as a structural inductive bias for modern deep embodied control.

Robotic control with proprioceptive data. Real-world deployment often dictates strict sensor constraints [17], making proprioception vital for robust locomotion [18, 19, 20]. To compensate for missing exteroceptive data, standard paradigms rely on privileged learning and history encoders to estimate environmental parameters [21], or employ explicit force sensing [22, 23, 24, 25, 26] and specialized compliance controllers for manipulation [27, 28, 12, 20]. In humanoid locomotion, managing unengineered terrain and dynamic impacts requires tight coupling between sensory observation and action spaces [29, 30]. Rather than reconstructing unobserved quantities via heavy state estimators, PRISM demonstrates that vital physical cues are already implicitly embedded within the multiplicative relationships of deployable signals, enabling force-free compliance and balance recovery.

3 Polynomial Representations for Interaction-Structured Motor Control

We introduce PRISM, a compact policy representation that exposes low-degree interactions among deployment-available robot observations. The goal is not to hand-design physical quantities such as power, slip, contact impulse, or compliance. Instead, PRISM provides a learnable interaction basis in which such quantities can emerge from products of observable variables. Many control-relevant cues are not separate sensor readings, but latent multiplicative relations among proprioception, commands, actions, and motion variables.

Fig. 2 shows that PRISM can be inserted into reinforcement and imitation learning pipelines with minimal changes to the policy interface. For humanoid locomotion, we add an element-wise polynomial module to the PPO [31] actor while keeping the same sensor-minimal input and low-level PD controller. For contact-rich manipulation, we replace the linear proprioceptive conditioning layer in Diffusion Policy with a degree-2 polynomial interaction layer. In both settings, PRISM changes only the policy representation; it adds no deployment-time sensing or privileged physical input.

We consider policies that act using only deployment-available observations. Let $\pi_\theta(a_t \mid o_t)$ denote a policy mapping observation o_t to continuous action a_t. The actor does not observe contact forces, interaction wrenches, terrain friction, object mass, external perturbations, simulator states, or contact

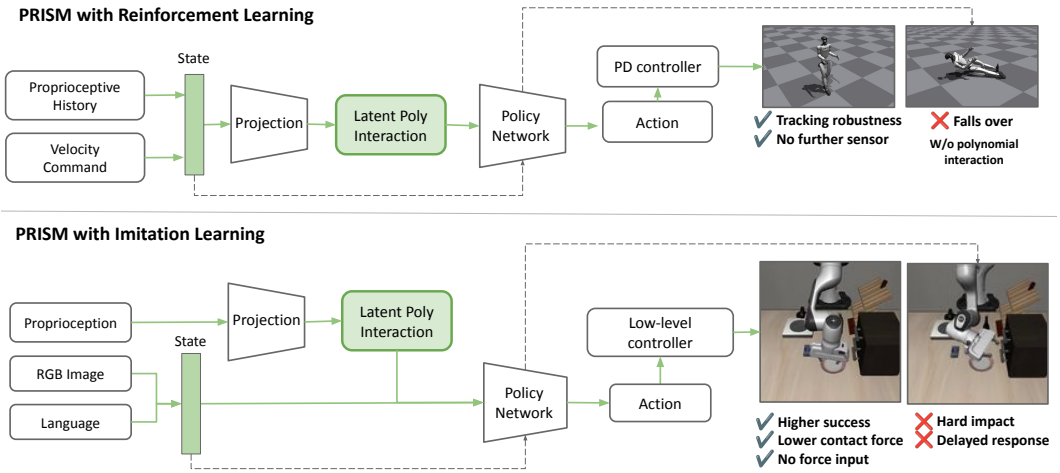

Figure 2: **PRISM exposes latent polynomial interactions within existing robot policy pipelines.** In reinforcement learning, PRISM augments the actor with a polynomial interaction branch over deployment-available locomotion observations, while keeping the same action interface and low-level PD controller. In imitation learning, PRISM replaces the standard linear proprioceptive conditioning pathway with a degree-2 latent polynomial interaction layer, which is then combined with visual and language-conditioned policy features for action prediction. Across both locomotion and manipulation, PRISM changes only the policy representation: it does not require force, wrench, tactile input, contact labels, or an explicit admittance controller at deployment.

labels. These variables affect the dynamics, but are unavailable at test time. We study whether their effects can be better captured by exposing interactions among variables the robot already observes.

For humanoid locomotion, the actor observes proprioceptive and command signals:

$$o_t^{\text{loc}} = [q_t, \dot{q}_t, a_{t-1}, \omega_t, r_t, v_t^{\text{cmd}}],$$

where q_t and $\dot{q}_t$ are joint positions and velocities, a_{t-1} is the previous action, ω_t is the base angular velocity, r_t is the projected gravity vector, and v_t^{cmd} is the target velocity command. The action a_t specifies target joint positions relative to a nominal pose, which are tracked by a fixed low-level PD controller. All locomotion methods therefore share the same observation space, action space, and low-level controller; they differ only in the actor representation.

For contact-rich manipulation, the policy receives visual observations and robot proprioception:

$$o_t^{\text{manip}} = [I_t^{\text{agent}}, I_t^{\text{wrist}}, s_t^{\text{proprio}}],$$

where I_t^{agent} and I_t^{wrist} are RGB images from the agent-view and wrist cameras, and s_t^{proprio} denotes the robot's internal proprioceptive state. The policy predicts relative end-effector actions $a_t \in \mathbb{R}^7$, consisting of 3D translation, 3D rotation, and a 1D gripper command, executed by the same operational-space controller as the baseline Diffusion Policy. Thus, PRISM does not modify the controller or introduce force, wrench, tactile, or contact supervision.

3.1 Latent Polynomial Representation

A standard MLP can represent nonlinear functions through depth and nonlinear activations, but products among physical variables remain implicit. PRISM makes such interactions directly available through a learned latent polynomial lift. Given a deployable input x_t, we project it into two latent feature spaces with learned affine maps and multiply the resulting factors element-wise:

$$\psi_2(x_t) = (W_u x_t + b_u) \odot (W_v x_t + b_v).$$

Each dimension of $\psi_2(x_t)$ is therefore a learned quadratic interaction over the original input variables. This gives the policy direct access to interaction features that can represent

mechanics-relevant couplings, such as velocity–velocity, state–velocity, action–state, or command–proprioception relations, without hand-engineering proxies for power, slip, contact, or compliance. The factorized form avoids enumerating all raw monomials while preserving the inductive bias that many physical cues arise from products among observable variables.

3.2 PRISM for Reinforcement Learning

For reinforcement learning, PRISM replaces the first actor projection with a polynomial encoder. Given the actor input x_t, the encoder first computes the latent quadratic feature $\psi_2(x_t)$ defined in Sec. 3.1. It then projects this feature to the actor dimension:

$$z_t = \mathrm{ELU}\left(W_p \psi_2(x_t) + b_p\right),$$

where W_p and b_p are learned parameters, and ELU denotes the ELU nonlinear activation [32]. The actor MLP maps z_t to the mean of a Gaussian action distribution and is trained end-to-end with PPO [31]. During locomotion training, the critic may use privileged simulator information for value estimation, but the deployed actor uses only deployment-available observations.

3.3 PRISM for Imitation Learning

For contact-rich imitation learning, we incorporate PRISM into Diffusion Policy [9] by replacing its standard linear proprioceptive conditioning layer with the latent polynomial representation in Sec. 3.1. The policy receives RGB observations from the agent-view and wrist cameras, together with proprioception s_t, and predicts a sequence of relative end-effector actions.

Instead of mapping s_t through a linear layer, PRISM computes the polynomial feature $\psi_2(s_t)$. This feature is projected to the standard conditioning dimension, concatenated with visual embeddings, and passed to the diffusion U-Net. The full policy, including the polynomial proprioceptive layer, is trained end-to-end with the standard noise-prediction imitation objective. At deployment, it uses no force measurements, wrench estimates, contact labels, tactile signals, privileged physical parameters, or admittance controller.

4 Experimental Results

We design our experiments to answer three primary questions: (1) Is PRISM effective at improving sensor-minimal locomotion over standard MLP actors, and do these gains stem from structured interaction modeling rather than simply expanding model capacity? (2) Can polynomial features be seamlessly integrated into RGB-based visuomotor policies to induce compliance-like behavior in manipulation tasks without requiring explicit force, wrench, or contact inputs? (3) Do the learned polynomial features inherently capture underlying physical quantities, and can task-vital cross-terms be effectively discovered in a purely data-driven manner?

4.1 Experimental Setups

Baselines. We evaluate PRISM in two complementary settings. First, we use Humanoid-Gym [10] to evaluate sensor-minimal humanoid locomotion, where the actor must maintain balance and track commanded velocities using only proprioceptive and command-related observations. Second, we evaluate a polynomial module on LIBERO [11] imitation-learning tasks.

For locomotion, we compare PRISM against a standard MLP actor, a larger-capacity MLP actor, and a polynomial actor. The larger MLP control tests whether the performance gain can be explained by parameter count alone. For imitation-learning experiments, we compare PRISM against the Diffusion Policy [9] and Minimalist Compliance Control (MCC) [12] paired with the diffusion policy. The Diffusion Policy baseline directly executes the base visuomotor policy. MCC-Sensorless evaluates the same base checkpoint with a delayed and noisy contact-force proxy inside a compliant low-level correction. All methods use the same demonstrations, RGB observations, proprioceptive state, relative end-effector action space, and low-level controller. MCC-Sensorless applies a noisy

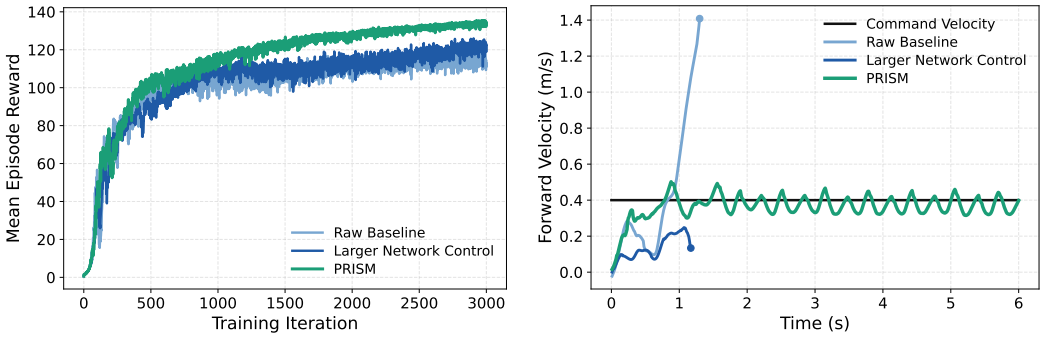

Figure 3: **PRISM improves both training stability and locomotion behavior.** Locomotion results on Humanoid-Gym [10]. Left figure shows training curves with shaded standard error across seeds. Residual polynomial actors learn more reliably and reach higher performance than baselines. Right figure shows forward command-tracking traces under a fixed forward command, where the black line denotes the command and others denote realized forward velocity. PRISM follows the target more closely and remains stable throughout the rollout.

Table 1: **PRISM improves humanoid locomotion beyond standard parameter scaling.** Performance comparison on Humanoid-Gym [10]. The larger MLP baseline has nearly the same number of trainable parameters as PRISM, but fails to close the performance gap. This indicates the performance leap is driven by structured polynomial interaction modeling rather than increased model capacity. Bold indicates the best result.

Method	Polynomial Interaction	Params	Episode Length ↑	Lin. Error ↓	Yaw Error ↓	Survival Rate ↑
MLP Baseline	✗	0.926M	1340.8	0.4607	0.2817	51.00
Larger MLP	✗	1.321M	1349.9	0.4738	0.3086	52.25
PRISM	✓	1.321M	**2233.4**	**0.2099**	**0.1221**	**92.50**

sensorless wrench correction at execution time, following the practical setting where true force measurements are not available. We also evaluate MCC-Oracle, which utilizes simulator contact force, included strictly as a non-deployable force-access ablation to provide a performance upper bound.

Metrics. For locomotion, we report episode length, linear-velocity tracking error, yaw-velocity tracking error, and survival rate. For LIBERO [11], we report success rate, smoothness, position error, and orientation error. Simulation results are averaged over five random seeds. For reinforcement-learning experiments, all methods use the same observation interface, action space, reward function, low-level controller, and PPO [31] training setup. For imitation-learning experiments, the base visuomotor policy and PRISM conditioning are trained end-to-end.

4.2 Comparison against Baselines

Humanoid locomotion. We first evaluate PRISM on the Humanoid-Gym [10], where the actor must stabilize a humanoid and follow the commanded velocities from only proprioceptive and command observations. Fig. 3 visually validates these gains. PRISM features exhibit tighter convergence bounds across training seeds. In closed-loop evaluation under fixed forward commands, PRISM tracks target velocities with minimal oscillation, while MLP alternatives suffer from severe compounding velocity drift and early falls. Table 1 shows that PRISM achieves the best episode length, linear-velocity tracking error, and survival rate. Crucially, the Larger MLP Control fails to bridge the performance gap with the baseline, indicating that the benefits of PRISM do not arise from scaling parameter counts. Instead, the performance leap occurs when multiplicative interactions are explicitly exposed. Degree 3 performs best, but we use Degree 2 as the default because it captures most of the gain with a simpler architecture (see Appendix).

Force-free polynomial conditioning in manipulation. We evaluate whether PRISM can induce contact-rich compliance without adding complex admittance controllers. Fig. 4 visualizes a representative rollout with a heavy emphasis on contact. MCC tracking displays destabilizing force spikes upon initial contact. PRISM maintains a low and stable contact-force profile throughout the rollout. Our approaches quickly before contact and then reduces end-effector speed after contact,

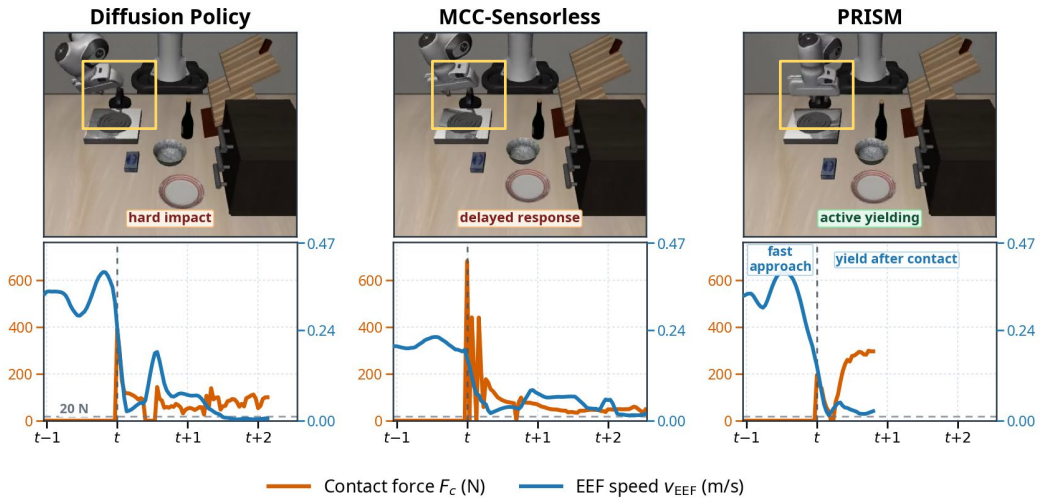

Figure 4: **PRISM exhibits compliance-like contact regulation without force input.** The top row visualizes a representative LIBERO [11] rollout for the *turn on the stove* task, with the contact area highlighted in yellow. The bottom row plots contact force (F_c, orange) and end-effector speed (v_{EEF}, blue), temporally aligned to the initial contact time t. While standard Diffusion Policy [9] also executes a fast approach, it suffers from a *hard impact* upon collision. Similarly, MCC-Sensorless [12] exhibits a *delayed response* and fails to adapt its motion in time, driving contact forces past 600 N. In contrast, PRISM demonstrates emergent compliance: it maintains a *fast approach* in free space for high efficiency, but uniquely transitions to *active yielding* immediately upon contact. This instantaneous velocity suppression absorbs the impact energy and regulates contact forces without requiring force, wrench, or tactile sensing.

Table 2: **PRISM maximizes manipulation task success without deployable hardware overhead.** Average task success across LIBERO [11] tasks. PRISM preserves the deployable execution of standard imitation learning while outperforming explicit sensorless compliance controllers. MCC [12]-Oracle uses direct simulator force and is included strictly as a non-deployable force-access ablation. We report scores for each category in Appendix.

Method	Success rate (%) ↑	Smoothness ↓	Position Error ↓	Orientation Error ↓
Diffusion Policy [9]	63.8	0.0098	<u>0.0267</u>	0.0314
MCC-Sensorless [12]	47.8	0.0100	0.0319	0.0318
MCC-Oracle [12]	<u>64.5</u>	**0.0095**	0.0268	<u>0.0313</u>
PRISM	**91.0**	<u>0.0096</u>	**0.0253**	**0.0272**

keeping the force response lower without force, wrench, or tactile input. This shows that polynomial proprioceptive conditioning functions as an implicit force-free compliance mechanism, allowing the policy to seamlessly contextualize the boundaries of the environment. Table 2 shows that PRISM achieves the highest task success rate across the completed LIBERO [11] tasks, including spatial, long, object, goal. PRISM improves successful-rollout position and orientation errors while maintaining the same observation and action interface as the base imitation policy. This suggests that polynomial proprioceptive conditioning helps the policy represent contact-relevant execution context, rather than simply damping actions.

4.3 Latent Physical Interaction Analysis

To verify PRISM captures physical structure rather than merely overfitting raw observations, we train linear probes on frozen policy representations to predict future mechanics-inspired responses. Table 3 demonstrates that PRISM makes these physical proxies significantly more linearly recoverable than baselines, improving predictions for both locomotion (slip, power) and manipulation (contact impulse, work) despite strictly relying on sensor-minimal inputs.

Table 3: **PRISM representations make Newtonian quantities more linearly predictable.** We freeze trained policy representations and train ordinary linear probes on held-out rollout windows with horizon $H = 5$. Targets are mechanics-inspired quantities computed only for analysis from simulator states, contacts, and actions. Lower MSE and higher Pearson correlation coefficient (PCC) indicate better linear predictability. Gains are relative changes from Diffusion Policy [9] to PRISM.

		MSE ↓		PCC ↑		Relative Gain	
Target	Formula	Diffusion [9]	PRISM	Diffusion [9]	PRISM	MSE	PCC
Slip velocity	$\sum_{h\le H} \|v^{\text{foot}}_{xy,t+h}\|$	0.711	**0.700**	0.501	**0.549**	**−1.5%**	**+9.6%**
Joint power	$\log(1+\sum_{h\le H} \|\tau^{\top}_{t+h}\dot{q}_{t+h}\|)$	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	$\sum_{h\le H} F_{t+h}\|\Delta x_{t+h}\|$	0.362	**0.319**	0.755	**0.780**	**−11.9%**	**+3.3%**

Table 4: **Emergent task-specific interaction proxies discovered by PRISM.** We quantify latent factor importance by measuring the mean absolute change in the predicted action vector (Δa) when individual degree-2 polynomial factors are ablated. Terms are named by the dominant input variables in their corresponding affine factors. These interaction proxies are not manually specified; they emerge from the polynomial structure learned by the policy.

Task Context	Learned Interaction	Emergent Role	Δa
Humanoid velocity tracking	$\dot{q}_{\text{Lhippitch}}(t) \times \dot{q}_{\text{Lhippitch}}(t-1)$	velocity-memory coupling	**0.057**
	$\dot{q}_{\text{Rhippitch}}(t) \times \dot{q}_{\text{Lhippitch}}(t)$	joint-velocity coupling	0.025
	$\dot{q}_{\text{Lanklepitch}}(t) \times \dot{q}_{\text{Rhiproll}}(t-3)$	velocity-memory coupling	0.009
	$\dot{q}_{\text{Lhiproll}}(t) \times q_{\text{Lhiproll}}(t)$	joint state-velocity coupling	0.008

We quantify learned interactions by ablating one degree-2 latent factor at a time. For each factor, we set only that multiplicative latent feature to zero, while keeping the raw inputs, learned weights, and all other factors unchanged. The policy deviation is measured as the mean absolute change in the predicted action. To name each factor, we inspect the two affine branches that form it and report the dominant input-space product induced by their largest weights. Thus, each term is an input-space interpretation of a learned latent factor, not a hand-coded variable.

Table 4 shows that the highest-impact locomotion factors involve velocity memory, cross-joint velocity, and state–velocity interactions. Ablating these factors directly shifts the joint-position commands, indicating that PRISM uses learned quadratic structure to drive control.

5 Limitations

PRISM assumes that key action-relevant cues can be captured by low-degree polynomial interactions among deployment-available observations. It may be less effective when failures depend on long-horizon history, unobserved contact geometry, material properties, or higher-order dynamics. PRISM also cannot compensate for missing sensory coverage: If critical information is absent from vision or proprioception, failures may still occur. Our experiments focus on humanoid locomotion and contact-rich manipulation with fixed controllers. Future work should study adaptive polynomial degree, temporal interactions, broader robot morphologies, and more diverse real-world contacts, objects, and terrain.

6 Conclusion

We presented PRISM, a compact policy representation that makes polynomial interactions among observable physical variables explicit and learnable. Across humanoid locomotion and contact-rich manipulation, PRISM improves over standard and larger MLP baselines, supports sensorless compliant behavior, and requires no additional deployment-time sensing or privileged physical inputs. These results suggest that polynomial representations should become a standard architectural choice for embodied motor control.

References

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

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

[3] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[4] B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[5] 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.

[6] A. Som, H. Choi, K. N. Ramamurthy, M. P. Buman, and P. Turaga. Pi-net: A deep learning approach to extract topological persistence images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 834–835, 2020.

[7] K. Usevich, R. Borsoi, C. Dérand, and M. Clausel. Identifiability of deep polynomial neural networks. *Advances in Neural Information Processing Systems*, 38:81809–81858, 2026.

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

[9] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

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

[32] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 4(5):11, 2015.

[33] R. Cadene, S. Aliberts, F. Capuano, M. Aractingi, A. Zouitine, P. Kooijmans, J. Choghari, M. Russi, C. Pascal, S. Palma, et al. Lerobot: An open-source library for end-to-end robot learning. *arXiv preprint arXiv:2602.22818*, 2026.

[34] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[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.

Appendix

A Experimental Details

Across locomotion and manipulation, all compared methods share deployable observations, actions, controllers, rewards or demonstrations, and evaluation protocol. PRISM changes only the representation used to expose polynomial interactions, isolating the architectural contribution from sensing, controller, reward, and baseline-tuning differences.

A.1 Humanoid Locomotion Protocol

We evaluate Unitree G1 locomotion in Humanoid-Gym [10]. The deployed actor receives only proprioceptive history, commanded velocity, previous actions, base angular velocity, and base orientation. It does not receive terrain friction, external pushes, mass randomization, stance masks, contact labels, or contact forces. The PPO critic [31] uses privileged simulator quantities during training only.

The actor observation is a 15-frame history of a 47-D frame: sinusoidal gait phase, commanded linear/yaw velocity, joint-position error, scaled joint velocity, previous action, scaled base angular velocity, and base roll/pitch/yaw. The critic receives a 3-frame history of a 73-D privileged frame, additionally including reference-pose error, base linear velocity, randomized push force/torque, friction coefficient, normalized base mass, stance mask, and contact mask.

Actions are 12-dimensional joint-position offsets around the default pose. The low-level PD controller uses action scale 0.25, control decimation 10, simulator timestep 0.001s, $K_p = \{100, 150, 40\}$ for hip/knee/ankle joints, and damping $\{2, 4, 2\}$ for hip/knee/ankle joints. Commands are resampled every 8s with $v_x \in [-0.3, 0.6]$, $v_y \in [-0.3, 0.3]$, and $\omega_z \in [-0.3, 0.3]$. Domain randomization includes friction in $[0.1, 2.0]$, base-mass offset in $[-5, 5]$ kg, random pushes every 4s, action delay, and action noise.

All locomotion methods use the same Humanoid-Gym reward with non-negative total reward clipping, domain randomization, action space, and low-level controller. Table 5 lists the active reward scales.

Here $e_v = v_{xy} - v_{xy}^{\mathrm{cmd}}$, $e_\omega = \omega_z - \omega_z^{\mathrm{cmd}}$, g_{xy} is the horizontal projected-gravity vector, c_i is measured foot contact, and s_i is the desired stance mask. The low-speed term is $R_{\mathrm{low}} = -2$ for sign mismatch, -1 if $|v_x| < 0.5|v_x^{\mathrm{cmd}}|$, 1.2 if $0.5|v_x^{\mathrm{cmd}}| \le |v_x| \le 1.2|v_x^{\mathrm{cmd}}|$, and 0 otherwise. The spacing score is $D(d; d_{\min}, d_{\max}) = \frac{1}{2}\left(e^{-100|\mathrm{clip}(d-d_{\min}, -0.5, 0)|} + e^{-100|\mathrm{clip}(d-d_{\max}, 0, 0.5)|}\right)$. Table 6 summarizes the shared PPO setup, and Table 7 reports the capacity-matched actor sizes.

A.2 LIBERO Manipulation Protocol

We evaluate contact-rich imitation learning on LIBERO [11] with task-specific LeRobot Diffusion Policy checkpoints [9, 33]. Each task has an independent policy trained only from its demonstrations. The reported average covers 40 policies across LIBERO-Spatial, Object, Goal, and Long. Inputs are agent RGB, wrist RGB, and proprioception: end-effector pose, gripper state, and robot joint state. Actions are 7D relative end-effector commands executed by LIBERO's relative operational-space controller. PRISM receives no force, wrench, tactile input, contact labels, object-state privilege, or simulator-only physical parameters.

Manipulation uses imitation learning, not environment reward. The Diffusion Policy [9] uses the standard denoising objective [34], $\mathcal{L}_{\mathrm{DDPM}} = \mathbb{E}\|\epsilon - \epsilon_\theta(a_k, k, g_t)\|_2^2$, where a_k is a noised demonstration action chunk, k is the diffusion step, and g_t is the global condition from RGB and proprioception. No reward, force/contact supervision, smoothness loss, or auxiliary physical loss is added. PRISM replaces the linear proprioceptive conditioner with a latent degree-2 polynomial kernel over the two-step state history, where W_ℓ and W_r produce learned 256-D factors. The kernel conditions the diffusion U-Net end-to-end. Table 8 summarizes the shared imitation-learning setup.

Table 5: **All locomotion methods use the same reward function.** The Humanoid-Gym reward is shared across baselines, so only the actor architecture changes. Positive entries are reward terms and negative entries are penalties; zero-weight terms are omitted.

Term	Reward expression	Scale
Command tracking and base stability		
Linear vel. tracking	$\exp(-5\|v_{xy} - v_{xy}^{\mathrm{cmd}}\|^2)$	1.2
Yaw vel. tracking	$\exp(-5(\omega_z - \omega_z^{\mathrm{cmd}})^2)$	1.1
Hard vel. tracking	$\frac{1}{2}(e^{-10\|e_v\|} + e^{-10\|e_\omega\|}) - 0.2(\|e_v\| + \|e_\omega\|)$	0.5
Low-speed regulation	$R_{\mathrm{low}}(v_x, v_x^{\mathrm{cmd}})\mathbf{1}_{\|v_x^{\mathrm{cmd}}\|>0.1}$	0.2
Upright orientation	$\frac{1}{2}(e^{-10(\|\mathrm{roll}\|+\|\mathrm{pitch}\|)} + e^{-20\|g_{xy}\|})$	1.0
Base height	$\exp(-100\|h - h^\star\|)$	0.2
Velocity consistency	$\frac{1}{2}(e^{-10v_z^2} + e^{-5\|\omega_{xy}\|})$	0.5
Base acceleration	$\exp(-3\|\dot{x}_t^{\mathrm{base}} - \dot{x}_{t-1}^{\mathrm{base}}\|)$	0.2
Gait and reference-pose shaping		
Joint-pose tracking	$\exp(-2\|q - q^{\mathrm{ref}}\|) - 0.2\,\mathrm{clip}(\|q - q^{\mathrm{ref}}\|, 0, 0.5)$	1.6
Default joint pose	$\exp(-100 d_{\mathrm{yaw/roll}}) - 0.01\|q - q^0\|$	0.5
Feet air time	$\sum_i \mathrm{clip}(T_i^{\mathrm{air}}, 0, 0.5)\mathbf{1}_{\mathrm{first\ contact}}$	1.0
Feet clearance	$\sum_i \mathbf{1}(\|h_i^{\mathrm{foot}} - h_{\mathrm{foot}}^\star\| < 0.01)\mathbf{1}_{\mathrm{swing}}$	1.0
Contact number	$\mathrm{mean}_i\,[1 \text{ if } c_i = s_i, \text{ else } -0.3]$	1.2
Feet distance	$D(d_{\mathrm{feet}}; 0.2, 0.5)$	0.2
Knee distance	$D(d_{\mathrm{knee}}; 0.2, 0.25)$	0.2
Safety and regularization		
Collision	$\sum_j \mathbf{1}(\|F_j^{\mathrm{body}}\| > 0.1)$	-1.0
Contact force	$\sum_i [\|F_i^{\mathrm{foot}}\| - 700]_+$	-0.01
Foot slip	$\sum_i \mathbf{1}_{c_i}\sqrt{\|v_{i,xy}^{\mathrm{foot}}\|}$	-0.05
Action smoothness	$\|a_t - a_{t-1}\|^2 + \|a_t + a_{t-2} - 2a_{t-1}\|^2 + 0.05\|a_t\|_1$	-0.002
Torque	$\|\tau\|^2$	-1.0×10^{-5}
Joint velocity	$\|\dot{q}\|^2$	-5.0×10^{-4}
Joint acceleration	$\|(\dot{q}_{t-1} - \dot{q}_t)/\Delta t\|^2$	-1.0×10^{-7}

Table 6: **All locomotion policies are trained with the same PPO setup [31].** The optimizer, rollout schedule, discounting, and action interface are shared across methods.

Item	Setting
Algorithm	PPO [31]
Obs. / priv. obs.	15×47 / 3×73
Action / controller	12D joint target, PD, scale 0.25
LR / entropy	10^{-5} / 0.001
γ / GAE λ	0.994 / 0.9
Rollout / epochs / minibatches	60 / 2 / 4
Iterations	3001
PRISM degree / hidden	2 / 256

A.3 MCC Compliance Baselines

MCC-style controllers [12] are implemented inside the LIBERO [11] wrapper and applied only at evaluation time to the same trained Diffusion Policy [9] checkpoints. MCC-Sensorless estimates translational external wrench from actuator generalized forces and the end-effector Jacobian, without simulator contact force:

$$(J_p J_p^\top + \epsilon I)\hat{f}_t = J_p \tau_{\mathrm{ext}},$$

where J_p is the translational Jacobian, $\tau_{\mathrm{ext}} = -(qfrc_{\mathrm{actuator}} - qfrc_{\mathrm{bias}})$, and $\epsilon = 10^{-4}$. The noisy, filtered, delayed estimate corrects translation as

$$a_{t,1:3}^{\mathrm{MCC}} = a_{t,1:3}^{\mathrm{nom}} - k_f \hat{f}_{t-d}.$$

Table 7: **PRISM is compared against both a standard MLP and a parameter-matched larger MLP.** Capacities are read from the seed-2 checkpoints used in our evaluation.

Method	Input / poly latent	MLP hidden widths	Actor params	Total params
MLP Baseline	705-D obs.	$[512, 256, 128]$	527K	926K
Larger MLP	705-D obs.	$[816, 352, 160]$	922K	1.321M
PRISM	$705 \rightarrow 256$ poly	$[512, 256, 128]$	922K	1.321M

Table 8: **All manipulation policies are trained with the same Diffusion Policy setup [9].** PRISM changes only the proprioceptive conditioning representation.

Item	Setting
Benchmark	LIBERO task-specific policies
Observation	Agent RGB, wrist RGB, proprioception
Action / controller	7D relative EEF, LIBERO OSC
Policy	LeRobot Diffusion Policy [9, 33]
Image size / obs. steps	128×128 / 2
Horizon / action steps	16 / 8
DDPM train / inference steps	100 / 10
U-Net dims	$[128, 256, 512]$
Batch / optimizer	64 / AdamW
LR / weight decay	10^{-4} / 10^{-6}
AdamW betas	$(0.95, 0.999)$
Scheduler	Cosine LR schedule
Training / checkpoints	20K steps / every 10K
PRISM kernel	State history, latent quadratic, $256/256$
Evaluation	10 online rollouts per task, seed 0

The selected validation setting is $k_f = 0.015$, delay $d = 2$, EMA coefficient 0.90, noise std. 0.15, and per-axis correction clipping 0.05. MCC-Oracle uses MuJoCo contact force directly, with no artificial noise or delay, and is included only as a non-deployable force-access ablation. Table 9 lists the deployed MCC parameters, and Table 10 reports the validation sweep used to select MCC-Sensorless.

A.4 MCC-Sensorless Tuning

MCC-Sensorless is tuned on a validation split over force gain, delay, EMA filtering, force-estimate noise, and correction clipping. The selected configuration is fixed for held-out evaluation; MCC-Oracle is not selected by this sweep because it is a non-deployable force-access ablation.

A.5 Metrics and Contact Diagnostics

LIBERO success is the environment success signal. Smoothness is the mean squared difference between consecutive executed actions. Position/orientation errors are end-effector Euclidean distance and geodesic rotation error to the controller target. Contact force is the MuJoCo external-contact proxy in Newtons and is used only for evaluation; excess contact means force $>$ 20N. Success-conditioned smoothness and pose errors average only successful episodes. For contact diagnostics, rollouts are aligned at first contact time t_c; approach speed is pre-contact end-effector speed, speed drop is pre-minus-post-contact speed, impulse is $\sum_{t \geq t_c} F_t \Delta t$, and contact power is $F_t \| v_{\mathrm{EEF},t} \|$.

Table 9: **MCC baselines [12] are applied as evaluation-time compliance controllers.** Both variants use the same trained Diffusion Policy checkpoint and modify only the executed translation command.

Parameter	Sensorless	Oracle
Force source	$J_p,\ qfrc$	Simulator contact
Force gain k_f	0.015	0.015
Noise std.	0.15	0
Delay d	2 control steps	0
EMA coefficient	0.9	0.9
Regularization ϵ	10^{-4}	10^{-4}
Correction clip	0.05 / axis	0.05 / axis
Controller gain	$K_p = 50$	$K_p = 50$
Damping ratio	1.0	1.0
Correction dims	Translation	Translation
Action clipping	Enabled	Enabled
Deployable	Yes	No

Table 10: **MCC-Sensorless is tuned on a validation split before held-out evaluation.** We tune MCC evaluation-time parameters on a validation subset and select by success rate, then successful-episode contact impulse. † denotes metrics averaged over successful episodes only.

Cfg.	k_f	Delay	EMA	Noise	Succ. (%) ↑	Imp.† ↓
c05	0.015	2	0.90	0.15	25.0	390.4
c04	0.015	1	0.90	0.15	25.0	494.0
c03	0.010	1	0.90	0.05	20.0	251.5
c07	0.010	1	0.70	0.15	15.0	324.6
c01	0.005	1	0.90	0.05	15.0	392.7
c06	0.020	1	0.90	0.15	15.0	423.6
c00	0.005	0	0.70	0.05	10.0	227.8
c08	0.015	1	0.70	0.05	10.0	248.8

B Additional LIBERO Results

B.1 LIBERO Category Breakdown

Table 11 reports the completed 20K-step LIBERO results across 40 task-specific policies. Each suite contains 10 tasks, with 10 online rollouts per task.

C Real-Robot Pilot and Additional Controls

C.1 SO-101 Real-Robot Pilot

We run a small SO-101 tabletop pilot [33] to evaluate PRISM on real hardware. ACT [35] and PRISM-ACT are trained from the same leader-arm teleoperation demonstrations and use the same front RGB stream, SO-101 joint state, joint-position delta command representation, and low-level servo interface. PRISM-ACT changes only the proprioceptive conditioner to a latent degree-2 polynomial kernel and uses no force, wrench, tactile, contact-label, or admittance input. On a randomized push-to-target task, PRISM-ACT succeeds in 8/10 matched trials and ACT in 6/10. Table 12 summarizes the pilot interface and success counts.

C.2 Evaluation-Time Perturbations

We perturb trained checkpoints only at evaluation time. Action delay uses a one-step FIFO buffer, so the action selected at t executes at $t + 1$. Proprioceptive noise is zero-mean Gaussian noise with std. 0.01 on the processed state tensor. Image corruption is zero-mean Gaussian noise with

Table 11: **PRISM improves success across LIBERO task families.** Results are averaged over task-specific policies and online evaluation rollouts. MCC-Oracle is a non-deployable force-access ablation. The last row reports absolute percentage-point gain over the standard Diffusion Policy.

Method	Spatial	Object	Goal	Long	Avg.
Diffusion	79.0	36.0	79.0	<u>61.0</u>	63.8
MCC-Sens.	67.0	31.0	54.0	39.0	47.8
MCC-Oracle	<u>81.0</u>	<u>37.0</u>	<u>82.0</u>	58.0	<u>64.5</u>
PRISM	**88.0**	**97.0**	**95.0**	**84.0**	**91.0**
PRISM – Diffusion	+9.0	+61.0	+16.0	+23.0	+27.2

Table 12: **The SO-101 pilot [33] keeps the same deployable interface for both policies.** The PRISM-ACT policy differs from ACT only by the latent polynomial proprioceptive conditioner. Current/load proxies are logged only for analysis and are not used by either policy.

Method	Task setting	Demo source	Observation	Success
ACT	Push to target	Leader teleop	Front RGB + joints	6/10
PRISM-ACT	Push to target	Leader teleop	Front RGB + joints	**8/10**

std. 0.05 on the processed image tensor, scaled to the valid image range and clipped; the combined setting uses delay 1, proprio noise 0.01, and image noise 0.03. The subset contains one contact-heavy task from each LIBERO family: `libero_spatial:1` (black bowl next to ramekin to plate), `libero_object:3` (bbq sauce to basket), `libero_goal:7` (turn on stove), and `libero_10:1` (cream cheese box and butter to basket). Table 13 reports success under each perturbation.

C.3 Physical-Probe Controls

All probing controls use the same held-out contact windows from the cross-suite diagnostic subset. We compare shuffled targets, random polynomial latents, and the learned PRISM linear latent before polynomial multiplication. All probes are ordinary least-squares linear probes with a fixed train/evaluation split; no ridge regularization or nonlinear probe is used. Table 14 reports the resulting probe controls.

D Ablations

D.1 Polynomial Degree Ablation

Table 15 isolates the effect of polynomial order in the locomotion actor. First-order lifting improves the representation only weakly, while introducing quadratic interactions produces the dominant performance gain. Cubic interactions add a small additional improvement, but we use degree 2 as the default because it captures the main benefit with a simpler interaction module.

Table 13: **PRISM remains robust under deployment-time perturbations.** Entries report success rate on the contact-heavy diagnostic subset under evaluation-only perturbations: one-step action delay, Gaussian proprioceptive noise, Gaussian image corruption, and their combination. All methods use the same trained checkpoints as the clean evaluation and are evaluated over 20 rollouts per condition.

Method	Clean	Delay	Prop.	Image	Comb.
Diffusion	<u>25.0</u>	<u>25.0</u>	<u>25.0</u>	<u>25.0</u>	10.0
MCC-Sens.	<u>25.0</u>	<u>25.0</u>	20.0	20.0	<u>15.0</u>
PRISM	**90.0**	**95.0**	**55.0**	**90.0**	**65.0**

Table 14: **Physical-response probing controls use the same held-out rollout windows.** All rows use the same held-out contact windows from a cross-suite LIBERO [11] diagnostic subset covering Spatial, Object, Goal, and Long, with ordinary least-squares linear probes.

	Contact impulse		Contact work	
Representation	MSE ↓	PCC ↑	MSE ↓	PCC ↑
Diffusion Emb.	2.742	0.451	13.373	0.235
PRISM Linear Latent	<u>0.807</u>	<u>0.688</u>	<u>0.315</u>	<u>0.831</u>
PRISM Poly. Latent	**0.378**	**0.825**	**0.290**	**0.860**

Table 15: **Humanoid locomotion benefits from higher-order interactions, not from first-order feature lifting alone.** Ablation on polynomial degree for Humanoid-Gym [10] locomotion. D1 corresponds to first-order lifting only, while D2 and D3 introduce quadratic and cubic interaction terms. All variants use the same PPO setup and actor backbone.

Method	Return ↑	Ep. Len. ↑	Lin. Err. ↓	Survival (%) ↑
PRISM-D1	75.26	1408.4	0.4685	55.75
PRISM-D2	131.78	2198.7	0.2254	91.00
PRISM-D3	**134.60**	**2296.5**	**0.2097**	**92.50**