PRISM: PRior-guided Imagination Sampling in world Models

Yuhai Wang[1], Jiawei Xia[2], Rongxuan Zhou[1], Xiao Hu[1], Yongliang Shi[3], Jing Du[4], and Yang Ye[1]

[1]*Northeastern University*
[2]*University of California, Berkeley*
[3]*Qiyuan Lab*
[4]*University of Florida*
https://.com

Abstract: A learned world model provides a powerful physical intuition for evaluating future states. But its effectiveness in continuous control also depends critically on how candidate actions are generated for model-based planning. Rather than solely asking how accurately a model can simulate the future, we ask: which candidate actions are worth evaluating in the first place? Existing planners typically search arbitrarily, or use expert demonstrations only to initialize a sampling mean—discarding the expert's state-conditioned confidence. Properly guiding this search requires a robust action prior, yet current approaches often rely on independent visual encoders or large-scale VLMs to obtain one. We argue that this architectural bloat is unnecessary: the exact same data—and the learned representations of the world model itself—inherently encode the agent's action intuition. We introduce **PRISM**, a task-agnostic framework that extracts both from a single dataset while maintaining strict architectural simplicity. Building on a standard JEPA-style latent world model, PRISM attaches a lightweight MLP directly to its frozen encoder to predict a state-conditioned Gaussian prior. At plan time, PRISM fuses this prior into the planner's sampling distribution via a precision-weighted Product-of-Gaussians update. This parameter-free, closed-form integration steers the sampling process, making the prior confident where it is and ceding control where it is not. PRISM improves success rates by 35 percentage points over vanilla world-model-based MPC on Cube and 32 percentage points on PushT, without introducing significant inference overhead.

Keywords: World Models, Joint-Embedding Predictive Architecture (JEPA), Sampling-Based Planning, Action Prior, Product of Gaussians

1 Introduction

Embodied AI is poised to reshape manufacturing, healthcare, and household assistance, but progress on these settings hinges on agents that produce physically-grounded actions driving the environment toward a desired state. End-to-end vision–language–action policies have shown impressive generalization on this problem [1, 2, 3], yet they require internet-scale data, are opaque at the action level, and remain expensive to deploy. World models offer a complementary path: a separately learned dynamics model lets the agent *imagine* the consequences of candidate actions before acting [4, 5, 6]. Recent latent world models [5, 7, 8] have made this paradigm increasingly competitive on continuous-control tasks.

Among latent world models, JEPA-style architectures are notable for their simplicity, focusing on representation learning. They predict in the embedding space rather than reconstruct pixels, avoid the instabilities of generative training, and support lightweight architectures without value heads or reward predictors. For example, LeWM [8] couples a JEPA encoder and action-conditional predictor

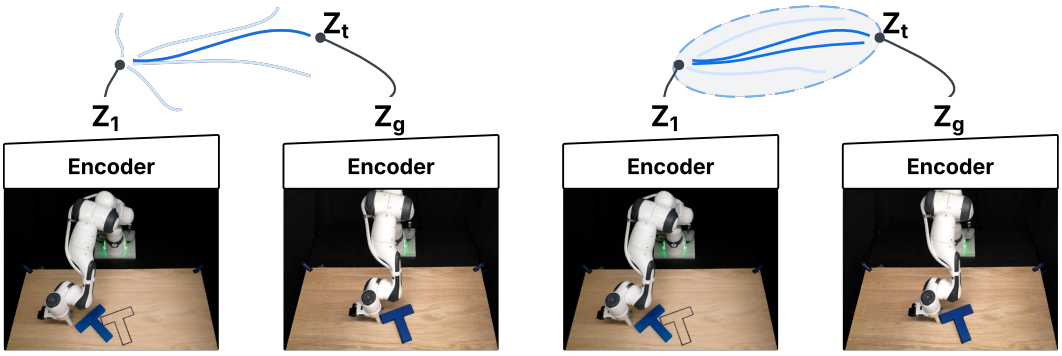

Figure 1: **Why a learned action prior matters.** A sampling-based planner proposes candidate action sequences (top) in the latent space of a shared visual encoder (bottom). *Left (prior planners):* an uninformed, isotropic initialization sprays candidates broadly, so most of the budget is wasted and only a few (dark blue) reach the goal. *Right (PRISM):* a lightweight prior read from the *same* encoder concentrates the initialization into a narrow, goal-directed distribution (dashed ellipse)—focusing the search where useful actions lie, at no added perceptual cost.

with sampling-based model predictive control: at every step, it draws candidate action sequences from a Gaussian distribution, rolls each through the predictor, scores by embedding-space MSE to a goal observation, and iteratively refits the distribution to the low-cost samples. Because this cost is purely visual—LeWM has no value or reward head—the entire optimization rests on the sampler: an uninformed Gaussian initialization may need hundreds of samples per step before the cost surface guides it toward a viable action.

What should the planner's sampling distribution be initialized with? In practice, the planner—built on either model predictive path integral control (MPPI) [9] or the cross-entropy method (CEM) [10]—initializes this distribution with a zero-mean, fixed-variance Gaussian and refines it iteratively. Yet the same dataset that trains the predictor (a *physical intuition* for what state an action leads to) also encodes an *action intuition*—which actions drive the state toward a goal. Existing approaches commonly obtain this action prior from a separate pretrained model, such as VLM-derived priors [11], thereby ignoring the implicit structure that ties it to the world model's learned latent physical representations. We hypothesize instead that building the action prior directly from the world model's latent representation informs the planner's initialization and improves sampling efficiency, without the architectural and computational redundancy of a separate prior model.

We propose **PRISM** (**PR**ior-guided **I**magination **S**ampling in world **M**odels): it builds the action prior from the same dataset and the same JEPA encoder the world model already uses through a lightweight MLP head that outputs a Gaussian over the upcoming actions. The action prior is fused with the planner through a closed-form Product-of-Gaussians (PoG) update [12] (Figure 1) to ensure the model falls back to a prior-free planner if the action prior approximation is highly uncertain. Without retraining or hand-tuning, PRISM improves multi-seed success of two visual manipulation tasks (Cube, PushT) by $+23$ to $+35$ pp over the same world model with an uninformed initialization (*vanilla MPPI*) at matched compute, with the largest gains at small sample budgets. Meanwhile, this performance lift comes at little/no computational overhead in inference. The same training-and-fusion pipeline transfers to two real-robot setups (Franka PushT, ARX X5 cube) (Sec. 4.4).

2 Related Work

World models. World models predict the consequences of candidate actions, enabling agents to plan via *imagined* rollouts rather than environment interaction [4, 5]. Two design choices dominate. *Generative* world models such as DreamerV3 [5] reconstruct pixels and pair dynamics with reward and value heads. *Embedding-space* world models such as DINO-WM [7] and LeWM [8] predict

only in a learned latent space and score plans by embedding-MSE to a goal observation. JEPA-style architectures are the prototypical embedding-space WMs: they side-step generative training instabilities, support lightweight backbones, and admit goal-conditional planning without auxiliary heads. We build on LeWM specifically because its absence of value or reward heads makes the planner's initialization the only locus where a learned prior can act—any improvement must come from the planner itself rather than from a value network absorbing the prior knowledge.

World models in sampling-based MPC. Pairing a world model with model predictive control (MPC) gives the model a concrete role: it serves as the imagination engine that scores candidate trajectories the planner proposes. In continuous action spaces, the dominant proposal mechanism is sampling-based MPC—MPPI [9] or CEM [10]—which iteratively draws hundreds of candidate action sequences from a Gaussian distribution, evaluates each via the WM, and refits the distribution to the elite samples. TD-MPC2 [6] pairs a generative latent WM with CEM-style sampling; LeWM and DINO-WM use the same pattern with embedding-MSE costs. In every case, the quality of the iterative refinement is bounded by two ingredients: the world model that scores the rollouts, and the sampling distribution that supplies them. The literature has invested heavily in the former; the latter is typically a zero-mean Gaussian with fixed isotropic variance. Biased-MPPI [13] provides a general derivation that admits arbitrary sampling distributions via KL-bias re-weighting; it is validated on low-dimensional state-space tasks (pendula, ships, wheeled vehicles) and does not provide a closed-form Gaussian instance suitable for direct integration with a learned head.

Action priors for planner sampling. A line of work seeks to inform the planner's sampling step with learned action priors. SPiRL [14] learns skill priors from offline data and uses them to regularize hierarchical RL exploration, operating at the policy level rather than the planner's per-step initialization. Probabilistic movement primitives (ProMP) [15] use Gaussian-product algebra to combine motion primitives in the open-loop, demonstration-only setting; PRISM applies the same algebra online inside a closed-loop planner. More recent systems inject a prior directly into MPPI's initialization. VLMPC [11] and Traj-VLMPC [16] use a VLM (GPT-4V) per env step to propose a low-dimensional Gaussian prior with a hand-set variance. PiJEPA [17] samples N_π trajectories from a fine-tuned Octo diffusion policy and uses their empirical mean and (clamped) standard deviation as the MPPI initialization. Behavior-cloning policies offer a harder alternative: deploy a BC controller outright, optionally combining with planning when the BC proposal looks suspect [18, 19, 20]. PRISM is closer to a soft, distributional version of these ideas: precision arithmetic does the weighting at every step rather than a hard switch.

In short, the embedding-space world-model setting generally lacks an action prior and integration mechanism that are *cheap to obtain* (i.e., additional perceptual system or inference-time policy network) and *robust* (i.e., fallback mechanism when action prior is less informative or even misleading), and no existing source provides both at once.

3 Method

3.1 Preliminaries: planning with a JEPA world model

World model (physical intuition). JEPA-style world model [8] learns a frozen encoder $h_\psi : \mathcal{O} \to \mathbb{R}^d$ and an action-conditional predictor f_θ that rolls an embedding forward under an H-step action sequence. Given a goal observation o_g with embedding $z_g = h_\psi(o_g)$, planning at state $z_t = h_\psi(o_t)$ minimizes a purely visual cost with reward-free

$$\mathrm{cost}(a_{t:t+H}) = \left\| f_\theta(z_t, a_{t:t+H}) - z_g \right\|_2^2. \tag{1}$$

Sampling-based planner. At each environment step, the MPC solver, MPPI [9], maintains a Gaussian over the H-step action sequence, $\mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$, initialized at $\mu{=}0$ with a fixed scale $\sigma{=}\sigma_\pi$. For J iterations it draws N candidates $a^{(i)} \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$, scores each by (1), and updates *only the mean* by softmax-weighted averaging,

$$\mu \leftarrow \textstyle\sum_i w_i\, a^{(i)}, \qquad w_i \propto \exp\!\big(-\mathrm{cost}^{(i)}/\lambda\big), \tag{2}$$

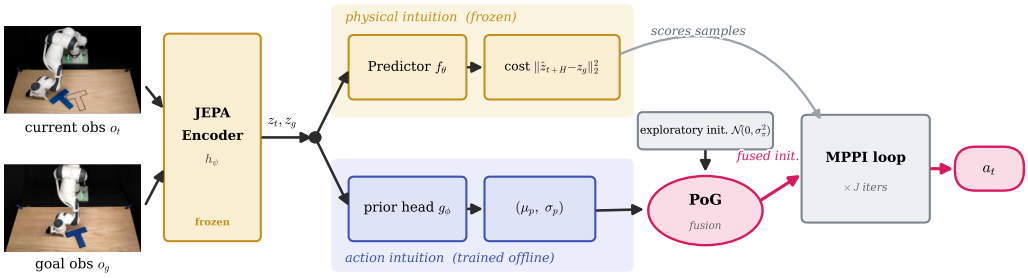

Figure 2: **PRISM architecture.** A single *frozen* JEPA encoder h_ψ embeds the current and goal observations, from which PRISM reads two intuitions: a *physical intuition* (top—the frozen predictor f_θ scores rolled-out actions against the goal) and an *action intuition* (bottom—a $\sim$1M-parameter MLP head g_ϕ giving a Gaussian prior over the next actions). A closed-form product of Gaussians (PoG) fuses this prior with the planner's default initialization; the fused initialization drives the otherwise-unmodified MPPI loop (mean updated, σ fixed). Only g_ϕ is trained, and only offline.

holding the sampling covariance σ^2 fixed across all J iterations. This fixed-covariance update is the defining difference from CEM, which additionally refits σ^2 to the elite samples—a distinction that becomes central in Sec. 4.3.

The gap. The initialization $\mathcal{N}(0, \sigma_\pi^2)$ is uninformed, so at small sample budgets N the planner spends iterations rediscovering, from scratch, action directions the demonstrations already exhibit. PRISM aims to supply that missing structure directly at the sampling initialization phase.

3.2 Action intuition: a Gaussian prior on frozen features

PRISM learns the action intuition from the *same* dataset and the *same* frozen encoder h_ψ (Figure 2)the world model uses. An *action-intuition head* g_ϕ maps the current and goal embeddings to a Gaussian over the next H-step action sequence, in the StandardScaler-normalized action space the planner operates in:

$$g_\phi : [z_t, z_g] \in \mathbb{R}^{2d} \longmapsto (\mu_p,\ \sigma_p) \in \mathbb{R}^A \times \mathbb{R}^A_{>0}. \tag{3}$$

It is a 3-layer GELU MLP (hidden width 512, $\approx$1M parameters). Because it consumes cached h_ψ features rather than pixels, it introduces no second vision encoder and runs in sub-millisecond time—about 1% of the world model's parameter budget.

Training. g_ϕ is trained offline on the dataset' $(z_t, z_g, a^\star_{t:t+H})$ tuples with the β-NLL loss [21] (β=0.5), which stabilizes the variance head against the marginal-mean collapse of plain Gaussian NLL. After training g_ϕ is frozen; no gradient flows through h_ψ or g_ϕ at plan time (loss and parameterization in Appendix B).

A single Gaussian is chosen to keep the fusion closed-form (Sec. 3.3); it cannot represent a genuinely multimodal action distribution. We show (Sec. 4.1) that even this unimodal prior is enough to dominate vanilla MPPI, and return to the multimodal ceiling in Sec. 5.

3.3 Precision-weighted fusion at the planner's initialization

At each step, before the first MPPI iteration, we fuse the planner's default initialization $\mathcal{N}(\mu_\pi, \sigma_\pi^2)$ with the prior $\mathcal{N}(\mu_p, (s\sigma_p)^2)$ by a per-coordinate product of Gaussians [12]. Writing precisions $\tau_\pi = \sigma_\pi^{-2}$ and $\tau_p = (s\sigma_p)^{-2}$,

$$\sigma^2_{\text{fused}} = (\tau_\pi + \tau_p)^{-1}, \tag{4}$$

$$\mu_{\text{fused}} = \sigma^2_{\text{fused}}\,(\tau_\pi \mu_\pi + \tau_p \mu_p), \tag{5}$$

with σ_{fused} clamped below at 0.05. The scalar s rescales the prior's standard deviation and is PRISM's *only* added hyperparameter (we use s=1, the head's σ_p as predicted; swept in App. E). We then run the *unmodified* MPPI loop initialized at $\mathcal{N}(\mu_{\text{fused}}, \sigma^2_{\text{fused}})$. Because MPPI updates only

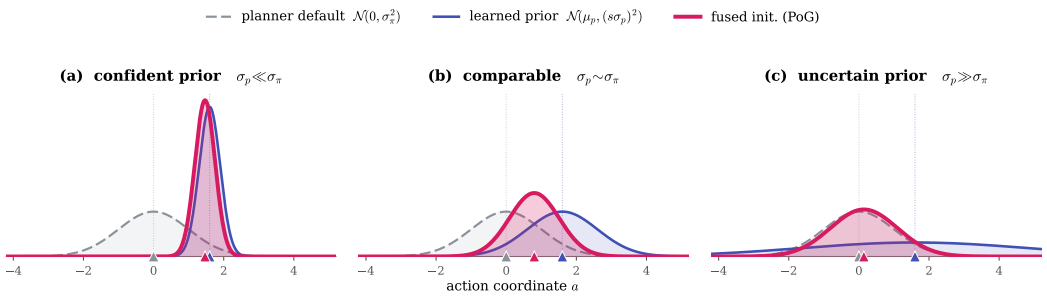

Figure 3: **PoG fusion is per-coordinate and confidence-aware.** On one action coordinate, the fused initialization (pink) is the product of the planner's default (grey, dashed) and the learned prior (indigo). **(a)** Confident head (σ_p small): the fused initialization concentrates at μ_p, narrowing the search. **(b)** Comparable uncertainty: precisions add, so it is narrower than either. **(c)** Uncertain head (σ_p large): the prior's precision vanishes and it reverts to the prior-free planner—graceful degradation (Sec. 3.3). We use s=1.

the mean and never refits the covariance (Sec. 3.1), the per-state σ_{fused} is the sampling width for *all* J iterations: the prior's confidence is not a one-shot initialization but persists through the entire optimization. Figure 3 illustrates the three regimes this induces.

Graceful degradation. The prior influences the initialization solely via its precision $\tau_p = (s\sigma_p)^{-2}$. When the prior is uncertain (e.g., out-of-distribution states where σ_p is large), $\tau_p \to 0$ and the initialization automatically reverts to vanilla MPPI: $(\mu_{\text{fused}}, \sigma_{\text{fused}}) \to (\mu_\pi, \sigma_\pi)$ (Fig. 3c). This per-coordinate fallback requires no learned gates or tuning, ensuring an unreliable prior cannot derail the planner. Furthermore, inflating the global scale s smoothly recovers the prior-free baseline, bounding the cost of mis-tuning (App. E).

Precision preservation. Conversely, when the prior is confident (σ_p is small), τ_p dominates and narrows the search around μ_p (Fig. 3a). Crucially, this confidence persists throughout optimization: the PoG fusion integrates σ_p into σ_{fused}, and MPPI's fixed-covariance update maintains it across all J iterations. Unlike warm-starting (which discards σ_p) or CEM (which overwrites it after one iteration), PRISM preserves the prior's per-state confidence end-to-end (Sec. 4.3).

Novelty. While the Product-of-Gaussians update is classical [12] and represents the Gaussian instance of Biased-MPPI [13], PRISM's core contributions are: (i) extracting the prior directly from the frozen world model encoder at zero added perceptual cost; (ii) a parameter-free initialization that preserves per-state precision and ensures graceful degradation; and (iii) leveraging MPPI's fixed-covariance update to maintain this precision throughout planning, avoiding the variance collapse seen in CEM. Computationally, PRISM adds only a sub-millisecond $O(A)$ overhead (one g_ϕ forward pass and one elementwise fusion) with no inference-time optimization (Appendix A).

4 Experiments

We evaluate PRISM on two simulated visual tasks (PushT, Cube) and preliminary real-robot platforms (Sec. 4.4). Section 4.1 compares PRISM against behavior-cloning baselines, vanilla MPPI, and a DINOv2 encoder ablation. Subsequent ablations isolate the role of the prior's variance (Sec. 4.2) and the choice of planner (MPPI vs. CEM, Sec. 4.3).

Environments and datasets. We evaluate on two goal-conditioned visual manipulation tasks [8] where the planner must match a target goal image (Figure 6). **PushT** (A=2, max length 246): A standard contact-rich 2D benchmark [20] requiring a planar pusher to align a T-block with a target pose. **Cube** (A=5, length 201): The `cube-single` robotic arm task from OGBench [22]. The

Table 1: **Main result.** Success rate (mean±std, in %) over 3 seeds $\{0, 1, 42\}$ at N=50 episodes per seed; *Learned-prior params* is the trainable parameter count of the learned action prior (PRISM's MLP head vs. DP's UNet), and the planning rows additionally use the shared frozen LeWM world model. Behavior-cloning policies are K-independent (no planning). All MPPI variants use J=30 iterations, H=5 plan-steps, action block 5. PRISM-MPPI is our method (PoG fusion with prior scale s=1, σ frozen across iterations).

Method	Learned-prior params	PushT			Cube		
BC-only (head μ alone)	1.0M	31 ± 5			66 ± 4		
Diffusion Policy [20]	19.3M	41 ± 10			77 ± 5		
Planning with World Model		K=32	K=64	K=128	K=32	K=64	K=128
Vanilla MPPI (DINO-WM-style) [7]	0	4 ± 2	3 ± 2	5 ± 1	45 ± 4	49 ± 6	41 ± 3
PRISM-MPPI (DINO-WM-style)	1.0M	13 ± 6	10 ± 7	15 ± 5	57 ± 7	65 ± 5	67 ± 3
Vanilla MPPI (LeWM) [8]	0	59 ± 5	61 ± 7	57 ± 6	46 ± 2	44 ± 5	44 ± 4
PRISM-MPPI (ours)	1.0M	$\mathbf{82 \pm 4}$	$\mathbf{86 \pm 6}$	$\mathbf{89 \pm 4}$	$\mathbf{79 \pm 2}$	$\mathbf{78 \pm 3}$	$\mathbf{79 \pm 6}$

world models and prior heads share the same expert dataset (18.7k trajectories for PushT, 10k for Cube). The prior head is trained offline via the β-NLL loss of Section 3.2 (dataset details in App. C).

Methods compared. We evaluate against two planner-free baselines: **BC-only** (executing the prior's mean μ_p directly) and **Diffusion Policy (DP)** [20] (a 19.3M-parameter multimodal UNet1D). For planning methods, we compare: **Vanilla MPPI** (default zero-mean initialization); **Warm-start MPPI** (initializes mean to μ_p, but discards σ_p for the default σ_π); **PRISM-CEM** (PoG initialization followed by CEM's adaptive-σ refitting, to ablate our fixed-σ design); and our full method, **PRISM-MPPI** (PoG fusion with fixed σ). To evaluate encoder generalization, we also test a **DINO-WM-style baseline** [7], swapping our from-scratch ViT-tiny for a frozen DINOv2-base.

Evaluation protocol. We report success rate (SR) over N=50 episodes, mean±std across seeds $\{0, 1, 42\}$, at three MPPI sample budgets $K \in \{32, 64, 128\}$; multi-seed differences use paired statistics (matched seeds). Planner hyperparameters and the per-task success threshold are in Appendix C.

4.1 Main result

Table 1 reports the headline comparison across all baselines, our method, and the encoder ablation, on both tasks and at three planner budgets $K \in \{32, 64, 128\}$.

Initialization over budget. Performance gains stem from informed initialization rather than large sampling budgets. PRISM-MPPI consistently dominates across all budgets; remarkably, our K=32 variant outperforms vanilla MPPI at K=128 by +25 pp on PushT and +35 pp on Cube.

Behavior-cloning baselines. Neither behavior-cloning policy matches planning performance. The BC-only approach collapses to 31% on PushT, as its unimodal Gaussian fails to capture the multimodal demonstration distribution, though it fares better on the unimodal Cube task (66%). An expressive Diffusion Policy improves PushT to 41% but still trails all planning methods. This confirms that simply executing a prior is insufficient; the primary advantage lies in using it to *initialize* the planner.

Encoder agnosticism and ablation. Finally, PoG fusion provides value independent of the visual representation, though absolute performance heavily depends on the encoder. Swapping our task-trained ViT-tiny for a frozen DINOv2-base yields competitive results on Cube (67% at K=128) but causes a collapse on PushT (10–15%). This discrepancy arises because DINOv2's CLS token encodes global semantics rather than the fine 2D spatial coordinates PushT requires (Appendix F). Despite this absolute drop, PRISM maintains a +7 to +26 pp relative lift over vanilla MPPI on the frozen DINOv2 encoder, confirming the fusion mechanism itself is encoder-agnostic.

Table 2: **Variance ablation (LeWM encoder).** A component ladder: adding the prior's mean (warm-start), then its per-state precision (PoG fusion). Success rates (SR) are mean±std over seeds $\{0, 1, 42\}$. Inference time (ms/plan) is measured on Cube (K=128) using an RTX 5090.

	PushT SR (%)			Cube SR (%)			Time (ms)
Method	K=32	K=64	K=128	K=32	K=64	K=128	(K=128)
Vanilla MPPI (no prior)	59 ± 5	61 ± 7	57 ± 6	46 ± 2	44 ± 5	44 ± 4	210.1 ± 1.2
Warm-start (μ_p only)	75 ± 3	69 ± 13	66 ± 1	51 ± 1	57 ± 4	55 ± 3	210.7 ± 2.2
PRISM-MPPI (μ_p+σ_p)	82 ± 4	86 ± 6	89 ± 4	79 ± 2	78 ± 3	79 ± 6	211.6 ± 4.8

Table 3: **Planner ablation: frozen vs. adaptive σ.** PRISM-CEM initializes identically to PRISM-MPPI but refits σ from elites each iteration. SR mean±std over seeds $\{0, 1, 42\}$, N=50.

	PushT			Cube		
Method	K=32	K=64	K=128	K=32	K=64	K=128
PRISM-CEM (σ adaptive)	43 ± 1	87 ± 6	91 ± 3	67 ± 9	81 ± 3	85 ± 6
PRISM-MPPI (σ frozen)	82 ± 4	86 ± 6	89 ± 4	79 ± 2	78 ± 3	79 ± 6

4.2 Variance Ablation: Warm-start vs. PoG Fusion

To isolate the benefit of the prior's per-state precision σ_p, we compare PRISM against a *warm-start* baseline. This baseline initializes MPPI with the prior's mean μ_p but discards σ_p in favor of the default variance σ_π. Table 2 ablates these components: no prior, mean-only, and full fusion.

While the prior's mean alone improves upon vanilla MPPI (+8–16 pp on PushT, +5–13 pp on Cube), it fails to scale with the sample budget K. On PushT, warm-start performance actually declines from 75% to 66% as K increases, as extra samples drift around an uncalibrated mean. In contrast, PRISM's performance climbs ($82\% \rightarrow 89\%$). By incorporating per-state precision, PRISM outperforms mean-only warm-starting across all budgets and tasks (+21–28 pp on Cube, +7–23 pp on PushT). Ultimately, the variance term is what enables robust scaling with compute.

Computation-wise, this performance lift comes at no noticeable inference cost. As shown in Table 2, execution times across vanilla MPPI, warm-start, and PRISM-MPPI are statistically indistinguishable ($\approx$ 210–212 ms/plan). The theoretical overhead of PRISM consists solely of one prior-head forward pass (< 0.05 ms) and a closed-form elementwise fusion. This microsecond-level addition is entirely subsumed by the natural GPU variance of the shared MPPI loop (std $\pm$1–5 ms). Thus, integrating the action prior incurs no/little computational overhead at inference.

4.3 Why MPPI and not CEM: the σ-frozen design

PRISM-CEM is identical to PRISM-MPPI except that CEM refits σ from the top-K elites each iteration instead of holding it fixed. The two diverge most where samples are scarce: at K=32 on PushT (Table 3), PRISM-MPPI achieves $82 \pm 4\%$ while PRISM-CEM falls to $43 \pm 1\%$ — a paired difference of $+38.7 \pm 3.1$ pp (t= + 21.9, p≈0.002, $3/3$ matched seeds). As the budget grows the adaptive refit recovers and pulls level (e.g. 91% vs. 89% on PushT at K=128), because more elites eventually estimate σ reliably. The point is not that frozen σ dominates everywhere, but that it is *robust where compute is scarce*: PRISM-MPPI never suffers the low-budget collapse — precisely the regime a fast planner targets.

Mechanism. At low K with a biased prior, CEM's top-K elites cluster around a wrong mean and its refit collapses σ toward zero in that direction; MPPI's fixed covariance instead carries the head's per-state confidence through all J iterations. A σ-floor sweep confirms this collapse is structural, not a floor artifact (Appendix D).

4.4 Real-world preliminary

We deploy PRISM-MPPI on two real-robot platforms (Figure 4) to demonstrate that our training-and-fusion pipeline transfers to hardware without modification. While a matched vanilla-MPPI base-

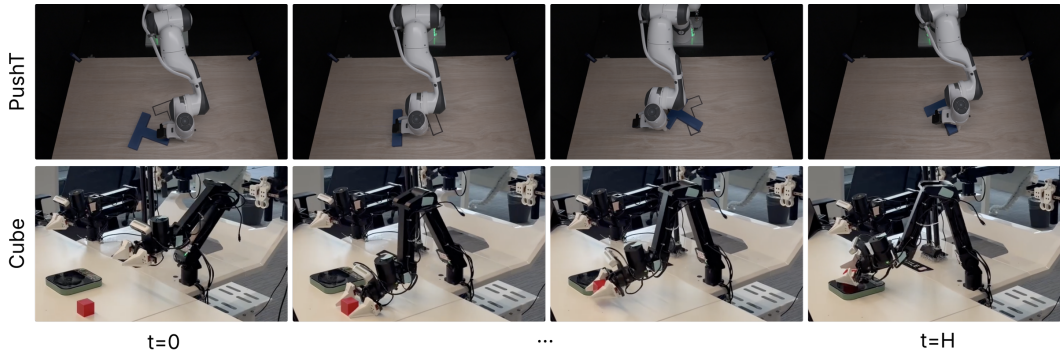

Figure 4: **Real-robot rollouts.** Keyframes (left→right) of example PRISM-MPPI episodes on hardware. *Top:* Franka PushT. *Bottom:* ARX X5 single-arm cube manipulation (Agilex Mobile ALOHA, third-person RealSense D435i). Success rates are reported in Sec. 4.4; a matched vanilla-MPPI baseline on hardware is in progress.

line to isolate the prior's contribution is currently in progress, we report preliminary single-method results. On the first setup, a Franka Research 3 (FR3) arm performs planar T-block pushing observed via a single third-person RealSense D455; trained on Meta Quest 3 teleoperation data and running real-time planning on an RTX 5080, PRISM-MPPI succeeds in 35% of 20 trials. On the second setup, an ARX X5 arm on an Agilex Mobile ALOHA platform performs single-cube manipulation. Using a single RealSense D435i with no wrist or egocentric cameras, joystick-teleoperation training data, and a local RTX 4090, the system achieves a 45% success rate over 20 trials. (Appendix G)

5 Discussion

PRISM's PoG fusion leverages the prior's predicted variance to outperform mean-only warm-starts: it narrows the search when confident and automatically reverts to the prior-free baseline when uncertain, guaranteeing graceful degradation. Furthermore, PRISM excels under restricted budgets or multimodal datasets. While a 19.3M-parameter Diffusion Policy (DP) matches PRISM on unimodal tasks with generous budgets (Cube, $K{=}128$), DP collapses on the multimodal PushT task (41% vs PRISM's 89%). Crucially, PRISM achieves this robust performance using only a 1.0M-parameter head on the frozen world model, bypassing the computational bloat of large standalone policies.

Limitations. First, PRISM's prior is bound by its training data: it requires task-specific demonstrations (limiting zero-shot generalizability) and relies on near-expert data to be useful, though our asymptote guarantee ensures graceful degradation on sub-expert datasets. Second, the reliability of a purely local action prior may degrade in highly complex or long-horizon tasks where temporal memory is required. Third, our unimodal Gaussian head under-fits genuinely multimodal experts; while PRISM still beats vanilla MPPI and matched-compute DP in these regimes, a mixture-of-Gaussians head could raise the ceiling. Finally, our controlled comparisons are in simulation; the matched real-robot baseline is currently in progress (Sec. 4.4).

6 Conclusion

We introduced PRISM, a closed-form, precision-weighted fusion of a learned action prior—read from the world model's own frozen encoder—into the planner's sampling distribution. It is parameter-free, adds no inference-time optimization, and provably degrades to vanilla MPPI when the prior is uninformative, yet improves multi-seed success by up to $+35$ pp at matched compute, with the largest gains at small sample budgets. PRISM exemplifies a broader principle: using precision arithmetic to integrate lightweight priors into a planner's initialization. In future work, we plan to extend this fusion framework to alternative prior sources and planning algorithms.

Acknowledgments

If a paper is accepted, the final camera-ready version will (and probably should) include acknowledgments.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[4] D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[5] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[6] N. Hansen, H. Su, and X. Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

[7] G. Zhou, H. Pan, Y. LeCun, and L. Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

[8] L. Maes, Q. Le Lidec, D. Scieur, Y. LeCun, and R. Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

[9] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[10] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

[11] W. Zhao, J. Chen, Z. Meng, D. Mao, R. Song, and W. Zhang. VLMPC: Vision-language model predictive control for robotic manipulation. In *Robotics: Science and Systems (RSS)*, 2024.

[12] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[13] E. Trevisan and J. Alonso-Mora. Biased-MPPI: Informing sampling-based model predictive control by fusing ancillary controllers. *IEEE Robotics and Automation Letters*, 9(6):5871–5878, 2024.

[14] K. Pertsch, Y. Lee, and J. J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020.

[15] A. Paraschos, C. Daniel, J. Peters, and G. Neumann. Using probabilistic movement primitives in robotics. *Autonomous Robots*, 42(3):529–551, 2018.

[16] J. Chen, W. Zhao, Z. Meng, D. Mao, R. Song, W. Pan, and W. Zhang. Vision-language model predictive control for manipulation planning and trajectory generation. *arXiv preprint arXiv:2504.05225*, 2025.

[17] A. Chahe and L. Zhou. Policy-guided world model planning for language-conditioned visual navigation. *arXiv preprint arXiv:2603.25981*, 2026.

[18] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[19] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[20] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[21] M. Seitzer, A. Tavakoli, D. Antic, and G. Martius. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. In *International Conference on Learning Representations (ICLR)*, 2022.

[22] S. Park, K. Frans, B. Eysenbach, and S. Levine. OGBench: Benchmarking offline goal-conditioned reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

[23] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

A Planning algorithm

Algorithm 1 gives one PRISM-MPPI planning step, following the data flow of Fig. 2. PRISM adds one g_ϕ forward pass and one elementwise fusion per environment step—$O(A)$ overhead, sub-millisecond, with no inference-time optimization; the MPPI loop is otherwise byte-for-byte the baseline planner.

Algorithm 1 PRISM-MPPI planning step at environment time t

Require: frozen h_ψ, predictor f_θ, frozen head g_ϕ, prior scale s, horizon H, iterations J, samples N

$z_t \leftarrow h_\psi(o_t),\ z_g \leftarrow h_\psi(o_g)$

$(\mu_p, \sigma_p) \leftarrow g_\phi(z_t, z_g)$ ▷ action-intuition prior over the next H actions

$(\mu_{\text{fused}}, \sigma_{\text{fused}}) \leftarrow \mathrm{PoG}\big((\mu_\pi, \sigma_\pi),\ (\mu_p, s\sigma_p)\big)$ ▷ Eqs. 4–5

$\mu \leftarrow \mu_{\text{fused}}$ ▷ σ_{fused} held fixed for all iterations

for $j \leftarrow 1$ to J **do**

  Draw $\{a^{(i)}\}_{i=1}^{N} \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma^2_{\text{fused}}))$

  $\text{cost}^{(i)} \leftarrow \|f_\theta(z_t, a^{(i)}) - z_g\|_2^2$

  $\mu \leftarrow \sum_i w_i\, a^{(i)}, \quad w_i \propto \exp(-\text{cost}^{(i)}/\lambda)$ ▷ mean only; σ_{fused} fixed

end for

return first action of μ

B Prior-head training

Architecture (Fig. 5). The action-intuition head g_ϕ is a 3-layer Multi-Layer Perceptron (MLP). It maps the concatenated current and goal embeddings, $[z_t;\ z_g] \in \mathbb{R}^{2D}$, to a per-element Gaussian distribution over an action sequence of length $H \times B$.

The specific dimensions are defined as follows: D is the JEPA embedding dimension (e.g., D=192 for LeWM [8]'s ViT-tiny encoder); H=5 is the planning horizon; B=5 is the action block size (representing the frame-skip from the world-model rate to the environment rate); and A is the raw environment-step action dimension (A=5 for the Cube task, A=2 for PushT).

The MLP architecture follows a straightforward progression: $\mathrm{Linear}(2D, 512) \rightarrow \mathrm{GELU} \rightarrow \mathrm{Linear}(512, 512) \rightarrow \mathrm{GELU} \rightarrow \mathrm{Linear}(512, 2HBA)$.

The output of the final layer is split evenly to form the mean and pre-activation precision. The mean head μ_p is returned unchanged. The standard-deviation head is computed as $\sigma_p = \mathrm{softplus}(\cdot) + 0.05$, where the 0.05 minimum variance floor matches the planner-side σ-floor detailed in Appendix D. Finally, both μ_p and σ_p are reshaped into $\mathbb{R}^{H \times B \times A}$ tensors.

This lightweight design results in total parameter counts of approximately 0.59M for Cube and 0.51M for PushT—constituting only about 1% of the total JEPA world-model budget.

Targets. To construct the training targets, we process the expert demonstrations into state-action-goal tuples.

- **Action Sequences:** For each valid starting frame t (chosen such that a full window fits and the boundary-NaN action at the episode end is excluded), we extract the next $HB = 25$ consecutive expert environment-step actions. These actions are normalized using the same StandardScaler applied during evaluation and reshaped into a tensor of dimension (H, B, A).
- **Goal Embeddings:** The goal embedding z_g is the JEPA embedding of the demonstration episode's final frame. This employs hindsight goal sampling [23], where the actual outcome serves as a self-consistent goal label, eliminating the need for external task-goal annotations.

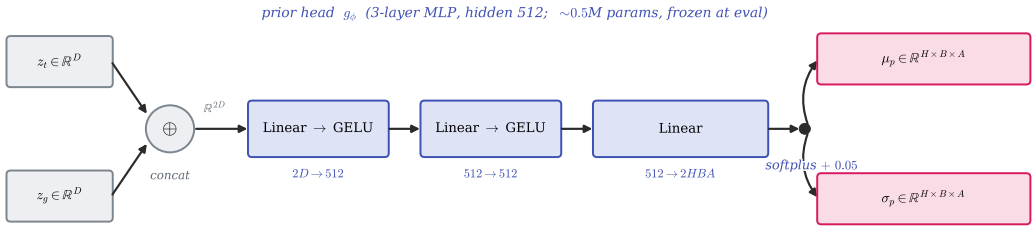

Figure 5: **Prior-head architecture.** A 3-layer MLP (hidden 512) maps the concatenated current/goal embeddings $[z_t;\, z_g] \in \mathbb{R}^{2D}$ to a per-element Gaussian over an $H \times B \times A$ action sequence. The mean head μ_p is returned unchanged; the standard-deviation head σ_p is a softplus with a 0.05 floor (matching Appendix D). Only g_ϕ is trained; the JEPA encoder h_ψ is frozen at every stage.

Consequently, the action-intuition head learns the distribution $p(a_{t:t+HB} \mid z_t, z_g^{\text{train}})$ based purely on demonstration-consistent (state, goal) pairs.

Train–Deploy Goal Distribution Shift. At deployment, the evaluator-specified goal image z_g^{deploy} generally comes from a different distribution than the training goals z_g^{train}. PRISM-MPPI elegantly handles this shift by decoupling the prior's prediction from the final executed action.

1. Prior Initialization: The prior is queried with z_g^{deploy} to generate the parameters (μ_p, σ_p). Crucially, these are used *only* to initialize the MPPI sampling distribution via a Product-of-Gaussians (PoG) fusion with the planner's default $\mathcal{N}(0, \sigma_\pi^2)$.

2. MPPI Optimization: Throughout the iterations, the sampling standard deviation σ remains frozen at this PoG-fused value (the signature of PRISM-MPPI). For each sampled action candidate a_{cand}, the planner:

- Rolls the action forward through the world model from the current state z_t.
- Computes the cost as the squared distance between the predicted final-step embedding and z_g^{deploy}.
- Updates the sampling mean by softmax-reweighting the candidates based on their negative costs.

3. Robustness Guarantee: Because candidate costs are evaluated against the actual deployment goal z_g^{deploy}, the prior's training distribution only biases the *initial* sampling. The final deployed action is entirely determined by MPPI re-weighting. This decoupling makes PRISM-MPPI highly robust to the $z_g^{\text{train}} \to z_g^{\text{deploy}}$ distribution shift. In contrast, direct-execution policies that bypass the planner—both BC-only and Diffusion Policy in our ablations—suffer a 15–20 percentage point drop in success rate under the exact same shift.

Loss. The head is trained with the β-NLL loss [21] (β=0.5),

$$\mathcal{L}(\phi) = \mathbb{E}_{(z_t, z_g, a^\star)} \left[\text{sg}\big((\sigma_p^2)^\beta\big) \left(\frac{(a^\star - \mu_p)^2}{2\sigma_p^2} + \log \sigma_p \right) \right], \tag{6}$$

where $(\mu_p, \sigma_p) = g_\phi(z_t, z_g)$, $\text{sg}(\cdot)$ is the stop-gradient operator, and the expectation is averaged elementwise over all HBA output components and over the minibatch. The $\sigma_p^{2\beta}$ reweighting interpolates between plain Gaussian NLL (β=0) and MSE (β=1) and stabilizes the variance head against NLL's tendency to collapse onto the marginal mean when μ_p is hard to fit.

Optimization. AdamW (learning rate 3×10^{-4}, weight decay 10^{-4}, (β_1, β_2)=$(0.9, 0.999)$), batch size 256, 50 epochs. The learning rate follows a 1,000-step linear warm-up and then a cosine decay to zero over the full schedule. Episodes are split 90:10 into train/val at the episode level (Cube:

9,000/1,000; PushT: 16,817/1,868); we report results using the lowest-val-NLL checkpoint. After training, g_ϕ is frozen; no gradient flows through h_ψ or g_ϕ at plan time. Each task trains in under 5 minutes on a single RTX 5090 once the JEPA encoder features are cached.

C Implementation and experimental details

Datasets. The Cube dataset contains 10,000 trajectories collected by a scripted oracle with 100.00% trajectory success rate (verified empirically over the full 2,010,000 frames). The PushT dataset contains 18,685 trajectories collected by a Diffusion-Policy [20] expert with 99.46% trajectory success rate (fraction of episodes terminating before the timeout horizon 246).

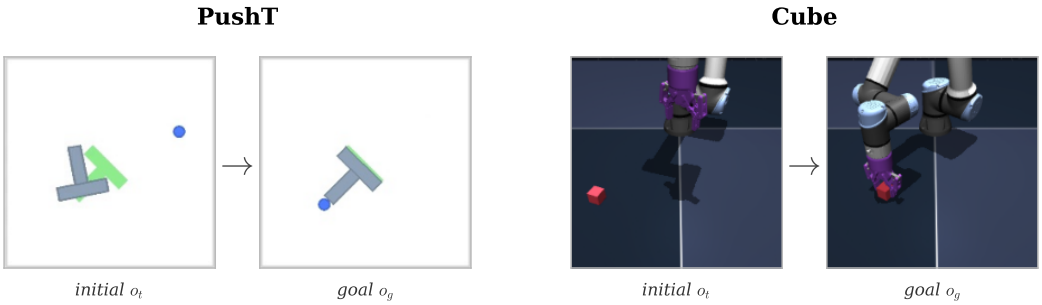

Figure 6: **Simulation tasks (goal-conditioned).** The planner receives a goal observation o_g and must drive the scene to it. *PushT:* push the grey T-block to align it with the green target pose; the blue disk is the pusher. *Cube:* a robot arm manipulates a single cube (OGBench's `cube-single`). Each pair shows an initial observation o_t and the goal observation o_g that conditions the planner.

Baselines. Diffusion Policy uses a 19.3M-parameter conditional UNet1D backbone with DDIM sampling (K=10), matched visual preprocessing (224×224 ImageNet-norm), and matched goal sampling at train and eval. The DINO-WM-style encoder ablation swaps our from-scratch ViT-tiny for a frozen DINOv2-base, keeping the LeWM recipe (projector, predictor, action encoder, SIGReg loss) byte-for-byte identical, with a learnable 768→192 projector to match the predictor capacity.

Planner hyperparameters. All MPPI variants use J=30 iterations, planning horizon H=5 plan-steps, action block 5 (frame-skip), and softmax temperature 0.5; σ_{fused} is floored at 0.05 and PRISM-MPPI uses prior scale s=1. Success uses the LeWM-paper threshold for each task.

Compute. All experiments run on a single NVIDIA RTX 5090 (32 GB) under CUDA 12.8. A full multi-seed cell (3 seeds × 50 episodes) takes approximately 1–2 minutes per (task, method, K) with LeWM and 1.5–2 minutes with the frozen DINOv2 encoder. Prior-head training takes under 5 minutes per task once the JEPA feature cache is built.

D Robustness to the variance floor

Robustness checks of the σ-floor (sweeping it in $\{0.05, 0.10, 0.20, 1.0\}$) confirm that the low-budget PRISM-CEM collapse (Sec. 4.3) is structural—driven by the elite-set variance refit—and not an artifact of the floor value.

E Asymptote behavior under prior-scale sweep

The PoG asymptote (Section 3.3) predicts that as the prior scale $s \to \infty$, PRISM-MPPI reduces to vanilla MPPI. We verify this by sweeping $s \in \{0.1, 0.3, 1, 2, 3, 10, 30, 100, 10^4\}$ on both tasks (multi-seed $\{0, 1, 42\}$, K=128; Figure 7). At s=10^4 the mean SR is within ∼2 pp of vanilla MPPI

on both tasks (Cube 43% vs 44%; PushT 59% vs 57%), confirming the asymptote. The curve is single-peaked near $s \approx 0.3$–1 and decays to the prior-free baseline by $s \gtrsim 10$; we use $s{=}1$ (the head's σ_p as predicted), which sits on the near-peak plateau. Because PRISM-MPPI degrades smoothly to vanilla MPPI above a task-dependent saturation point, the cost of mis-tuning s is bounded above by the prior-free baseline—unlike warm-start, where a bad μ_p persistently biases the planner with no corrective mechanism.

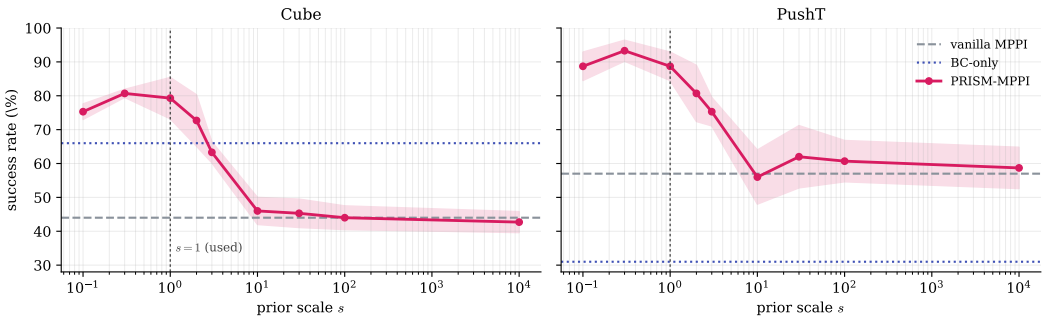

Figure 7: **Prior-scale sweep / asymptote.** Success rate vs. prior scale s (log axis) for PRISM-MPPI at $K{=}128$ (mean$\pm$std over seeds $\{0, 1, 42\}$), with vanilla-MPPI and BC-only reference lines. As $s \to \infty$ the prior's precision vanishes and PRISM-MPPI converges to vanilla MPPI; the SR peaks near $s \approx 0.3$–1 (we use $s{=}1$).

F Encoder-ablation analysis

We attribute the DINO-WM-style collapse on PushT (Sec. 4.1) to a known property of DINOv2's CLS-token aggregation: trained with random crops on natural images, the CLS token encodes *global semantics* rather than *fine 2D position*, which is exactly what PushT requires. Cube's 3D-rendered scenes (textured objects, lighting, articulated arm) align better with DINOv2's pretraining distribution, so the encoder transfers usefully there. The from-scratch JEPA training in LeWM avoids this mismatch by learning a task-adapted encoder; we read this as a concrete case for preferring task-adapted encoders when the visual domain is non-standard.

G Real-robot deploy details

Setup. We deploy PRISM-MPPI on two real-robot platforms (Franka FR3 for planar PushT and ARX X5 on an Agilex Mobile ALOHA for single-cube manipulation; main-paper results in Sec. 4.4). For concreteness, this appendix details the *Franka FR3 + PushT* setup as a representative example; the ARX X5 + cube pipeline follows the same training-and-fusion recipe with task-specific adjustments (RealSense D435i camera, joystick-teleoperation data, RTX 4090 host (Fig. 8b)).

The FR3 operates under Cartesian-impedance control with a single top-down RGB camera (Fig. 8a), resizing frames to 224×224 and forwarding them to the planner at 10 Hz to match the teleoperation rate used during training. The policy controls only the $(\Delta x, \Delta y)$ end-effector delta per tick; the z-height, wrist, and gripper are locked.

Dataset. We utilize a custom 411-episode teleoperation corpus collected via an expert human operator. To comply with double-blind review guidelines, the dataset is currently anonymized; the full snapshot will be publicly released on Hugging Face upon publication.

Hardware and Specifications. The dataset contains $93{,}728$ frames recorded at 10 Hz on the same Franka FR3 hardware used for deployment.

- **State Space:** Each frame is captured via a single top-down RGB camera and processed as a 224×224 uint8 RGB image.

(a) Franka FR3 + PushT (b) ARX X5 + cube

Figure 8: **Real-robot setups (red circle: single third-person RGB camera).** Both platforms use a single tripod-mounted RealSense camera as the only RGB sensor feeding the planner; no wrist or egocentric cameras are used. *(a)* Franka FR3 with a RealSense D455 viewing a wooden tabletop; the blue T-block must be aligned with the marked target outline. *(b)* ARX X5 on an Agilex Mobile ALOHA with a RealSense D435i viewing an office desk; the arm transports a single red cube.

- **Action Space:** The action is defined as the 2D end-effector positional delta $(\Delta x, \Delta y)$ in meters. The empirical per-tick standard deviation is $(8.4, 10.5)$ mm, with a peak magnitude of approximately 5 cm.

Heterogeneity and Training Splits. The episodes are intentionally heterogeneous in their terminal states:

- **Target-Completion Subset (First 200 episodes):** The operator pushes the T-block into a marked target region on the table.
- **Arbitrary-Stop Subset (Remaining 211 episodes):** The operator intentionally stops at arbitrary positions.

While the JEPA world model is trained on the full 411-episode corpus to maximize dynamics coverage, we restrict the prior-head training strictly to the first 200 target-completion episodes. This split is critical: applying hindsight goal sampling to the arbitrary terminal states of the remaining episodes would otherwise dilute the goal-conditioning, yielding a near-null prior.

Planner. Each call uses $K = 300$ MPPI candidates over $n_{\text{iters}} = 30$ iterations, with horizon $H = 3$ plan-steps of $B = 5$ env-ticks each (1.5 s of lookahead). We use $H = 3$ rather than the sim convention $H = 5$ to stay within the world model's high-fidelity rollout envelope on this dataset: the pred-to-identity error ratio is 0.15 at $H = 5$ but degrades to 0.33 at $H = 25$. The sampling σ is frozen at the PoG-fused value for all 30 iterations within a call (the PRISM-MPPI signature), then refreshed to the planner default $\sigma_\pi = 1.0$ before the next call.

Execution. The goal image is captured once per session with the T-block placed at the target position and reused across all calls of a trial. Each call returns $B = 5$ env-tick actions, denormalized via the StandardScaler stored with the prior, sent to the robot one per tick at 10 Hz; the planner replans every 0.5 s (receding-horizon shift B). A single call takes ~ 210 ms on an RTX 5090, well within the 0.5 s execution window; the prior-head forward adds < 0.5 ms over vanilla MPPI, so all three planner modes have effectively identical end-to-end latency.

Reproducibility. The deployed checkpoints and a standalone inference script supporting all three planner modes (`pog`, `warm_start`, `none`) are released on Hugging Face, depending only on PyTorch and standard libraries.