KEEPING JEPA WORLD MODELS PLANNABLE WHEN LITTLE OF THE FRAME MOVES

Anonymous authors
Paper under double-blind review

ABSTRACT

Specifying a goal in language rather than as a goal frame is a natural interface for planning with a latent world model, but testing it needs scenes in which language must discriminate between several objects. We build SLIM, a pushing benchmark with several small objects and paired visual and language goals on identical scenes. On SLIM a LeWM world model that solves PushT succeeds on under 1% of trials, although a scripted controller with simulator state solves every tier. Probes locate the failure in the encoder: its latent is nearly action-insensitive, neither pusher nor object positions can be decoded from it, and rollouts are no better than copying the current latent forward. One inverse-dynamics auxiliary loss, applied to encoder latents and to predicted latents through a shared head discarded at test time, restores every probe and raises success from 0.003 to 0.35 (0.16 on the hard pushing tier, where a goal-agnostic policy scores zero), and improves PushT at twice the trained horizon. Controls attribute the repair to the gradient into the encoder, and a response sweep shows that the vanilla model plans once enough of the frame responds to actions. A cheap action-sensitivity probe, computable without environment access, acts as an empirical necessary condition: all configurations below its threshold failed to plan. On the repaired latent, a small language-goal head plans from sentences without retraining the world model: it reaches 0.84 on navigation (visual-goal oracle 1.00), follows the named zone when it is swapped with a decoy, and degrades gracefully to unseen nouns, while choosing which object to push from language remains open.

1 INTRODUCTION

Visual planning agents in the wild rarely have a pixel-precise goal frame to plan toward; a *language description* of what to achieve is a far more natural interface. We set out to build such a planner on a JEPA-style latent world model (LeCun, 2022; Assran et al., 2023), a sample-efficient recipe for visual control (Maes et al., 2026b; Sobal et al., 2025; Zhou et al., 2025) that predicts latents rather than pixels (Ha & Schmidhuber, 2018; Hafner et al., 2023).

Two prerequisites must hold. (1) The benchmark must be visually diverse enough that language can *discriminate*; PushT (Chi et al., 2023), with a single object and a single goal region, is not rich enough to support linguistically diverse instructions. (2) The latent must be *plannable*: the encoder must keep action-relevant variance so a sampling-based planner has a descent direction. We built SLIM (small-object, language-instructed, multi-object pushing) for the first prerequisite and found that the second fails when little of the frame responds to actions.

A vanilla JEPA world model (LeWM, Maes et al. 2026b) that reaches 98% success on PushT succeeds on 3 of 1200 SLIM trials. Probes locate the failure in the encoder: the latent is nearly action-insensitive ($1500\times$ below the repaired model), neither the pusher nor the objects can be decoded from it, the predictor's rollout error exceeds the true latent motion tenfold, and the planner's cost landscape is flat. We trace this to how little of the frame responds to actions (Fig. 1): the pusher rarely touches one of several objects, which then move 24–$50\times$ less than PushT's T-block under sampled actions, so a representation that ignores all bodies costs the prediction loss little, and the isotropic regulariser (SIGReg) prevents global collapse without choosing *which* variance survives. Consistent with this account, doubling the object radius changes nothing, whereas making more of the frame respond to actions lifts the vanilla model from 0.005 to 0.530.

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

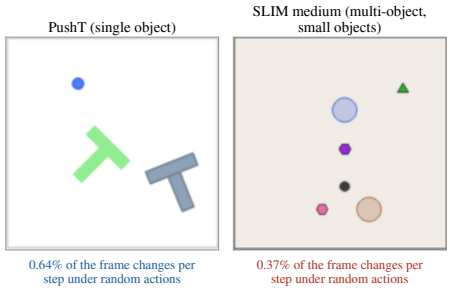


Figure 1: **Why JEPA world models collapse on SLIM.** Under the same random actions, 0.64% of the frame changes per predictor step on PushT (left), where the large T-block moves whenever it is touched, and 0.37% on SLIM (right; the same scene as in Fig. 2), where the pusher rarely hits one of several objects. When so little of the frame responds to actions, the prediction objective can be minimised by an action-insensitive encoder (Section 5.4).

We repair the latent with a single **inverse-dynamics auxiliary loss (I)**: one shared inverse-dynamics model (IDM) head predicts the action from consecutive encoder latents and from (current, predicted) latent pairs, and is discarded at test time. LeWM+I lifts SLIM success from 0.003 to 0.348 ± 0.016 (mean $\pm$ SE over three training seeds) and PushT success at twice the trained horizon by $+30$ pp. Controls show that *the gradient into the encoder carries the effect*: the encoder-side term alone recovers most of the gain, the predictor-side term alone none, and together they are best (Section 5.3).

With a plannable latent we return to language: a small head maps the current latent and a frozen sentence embedding of the goal to a goal latent for the planner's cost, leaving the world model untouched. Attending over the encoder's patch tokens, it reaches 0.84 on navigation against the visual-goal oracle's 1.00, ends closer to the named zone than to a swapped decoy in all 300 decoy episodes, and keeps 84% of its success with unseen nouns; on the pushing tiers it reaches about a fifth of the oracle.

Contributions.

- SLIM, a multi-object 2D pushing benchmark with four difficulty tiers and paired visual and language goals on identical scenes, which exposes the regime in which little of the frame responds to actions, at PushT-level compute (Section 3).
- A mechanistic account of *when and why* LeWM collapses (one model-free and four model-internal probes, and a response-to-actions sweep), an inverse-dynamics auxiliary loss on encoder and predicted latents that repairs it, and controls (encoder-vs-predictor disentangle, shuffled-action null, frozen-encoder cap) that pin the repair to encoder shaping (Sections 4 and 5).
- A language-goal head that plugs into the planner without retraining the world model, approaches the visual-goal oracle on navigation, grounds the named zone in a decoy test and degrades gracefully to unseen nouns (Section 6).

2 Related Work

JEPA world models for control. Joint-embedding predictive architectures (LeCun, 2022; Assran et al., 2023; Bardes et al., 2024; Assran et al., 2025) predict embeddings instead of pixels and regularise against collapse (Bardes et al., 2022; Balestriero & LeCun, 2025). For control, DINO-WM (Zhou et al., 2025) trains a predictor over a frozen DINOv2 encoder (Oquab et al., 2024), PLDM (Sobal et al., 2025) plans in a VICReg-regularised JEPA latent on state-based navigation, and LeWM (Maes et al., 2026b) trains a ViT encoder end-to-end with the SIGReg regulariser of LeJEPA (Balestriero & LeCun, 2025) (see also Terver et al., 2026). Inverse models as auxiliary objectives go back to learning to poke (Agrawal et al., 2016) and curiosity (Pathak et al., 2017); the controllable-latent-state literature uses them to keep what the agent controls and discard exogenous noise, and shows that one-step inverse models can miss parts of the controllable state (Lamb et al., 2023; Efroni et al., 2022; Islam et al., 2023; Wang et al., 2022; Levine et al., 2024; Yeom et al.,

2026). Our one-step IDM makes the pusher decodable; that it also restores the objects, which the action does not determine, is an empirical finding, and the weak pushing tiers are consistent with the limits of one-step inverse models.

Concurrent work. Concurrently with and independently of this work, several 2026 preprints add an inverse-dynamics objective on consecutive encoder latents to LeWM-style models. DA-LeWM (Wang et al., 2026a) adds an MSE IDM head at weight 0.1 with a goal-conditioned action head and studies convergence speed and cost geometry on PushT, Reacher, Cube and TwoRoom, where LeWM already plans. ATM (Chen, 2026) also tries a head on the predicted latent, but as a substitute for the encoder term rather than a shared head with stop-gradient, and finds it a shortcut. SMWM (Ivashkov et al., 2026), Delta-JEPA (Zhang et al., 2026b), AC-MTM (Boylan & Hokamp, 2026) and Liu et al. (2026) *replace* SIGReg by an MSE, latent-difference or contrastive IDM and see collapse only once the regulariser is removed; AC-MTM reports IDM *failing* where actions barely move a variable (PushT block orientation), whereas in our setting the whole latent collapses and IDM repairs it. None of these works studies scenes in which little of the frame responds to actions, reports a shuffled-action null, a frozen-encoder control or an encoder-vs-predictor disentangle, or plans toward language goals. ACPC (An et al., 2026) and ActSWM (Gan et al., 2026) target action sensitivity as a post-hoc screen and as a rollout constraint, respectively. Other latent-shaping objectives (Wang et al., 2026b; Bai & Xiong, 2026; Li et al., 2026b; Zeng et al., 2026; Toso et al., 2026) and object-centric latents (Nam et al., 2026; Daniel et al., 2026; Spieler et al., 2026; Zhang et al., 2026a) are complementary.

Language goals for latent planning. Language has been used as a learned reward for visual model-predictive control (Nair et al., 2021), as a goal space shared with goal images for policies (Lynch & Sermanet, 2021; Mees et al., 2022a), and to generate subgoal images (Black et al., 2024). DynaLang (Lin et al., 2024) predicts language tokens inside a Dreamer world model. Closest to our head is LAGO (Barbeau et al., 2026), which maps a frozen sentence embedding to latent (sub)goals for CEM by co-training the predictor over a frozen DINOv2 encoder. Our head is a separate 1–5M-parameter module trained on a frozen world model and is tested under paraphrase, which LAGO, GWM (Li et al., 2026a) and SAGE (Cheng et al., 2026) do not report. PiJEPA (Chahe & Zhou, 2026) uses language as a policy prior for model-predictive path-integral (MPPI) control, not as the planner's goal.

Benchmarks. PushT (Chi et al., 2023), BlockPush and Sawyer variants are single-object; OGBench's visual scene tasks (Park et al., 2024) are multi-object at 64×64, with language labels added by LAGO. Language-Table (Lynch et al., 2023) and CALVIN (Mees et al., 2022b) pair multi-object tabletop manipulation with far richer language; what SLIM adds is a controlled regime in which little of the frame responds to actions, at PushT-level compute.

3 THE SLIM BENCHMARK

Design. Standard world-model benchmarks sit in a regime where sampled actions change a sizeable part of the frame, so a world model can look strong without ever being tested on scenes where little moves. SLIM is a controlled testbed for that regime at PushT-level compute: several small objects per scene with a controlled distractor count, tiers from solved to unsolved, simulator constants that can be tuned to vary how much of the frame responds to actions (Section 5.4), a scripted controller as a ceiling, and paired visual and language goals on identical scenes.

The arena is a top-down 2D scene rendered at 224×224 and simulated with pymunk. The pusher is a 14 px-radius disc driven by a 2D target velocity $(\mathrm{d}x, \mathrm{d}y) \in [-1, 1]^2$ through a PD controller; objects are 15 px-radius bodies drawn from four shapes and a colour palette, damped so they stop shortly after contact. Every scene has two coloured zones, one of which is the target. Four tiers differ in geometry: *easy* is a navigation tier (pusher to target zone; 0–2 distractor objects, nothing to push); *medium* places pusher, target object and zone collinearly; *hard* adds ~ 25 px lateral offsets so the push must be steered; *very hard* puts pusher, object and zone far apart along a diagonal. Push tiers contain 1–3 objects of distinct colour, one of them the target. The tier's template (pusher, target object, target zone) is placed at a random position in the arena, distractor objects and zones are drawn from the same position distribution as the target object and zone, and the scene is mapped

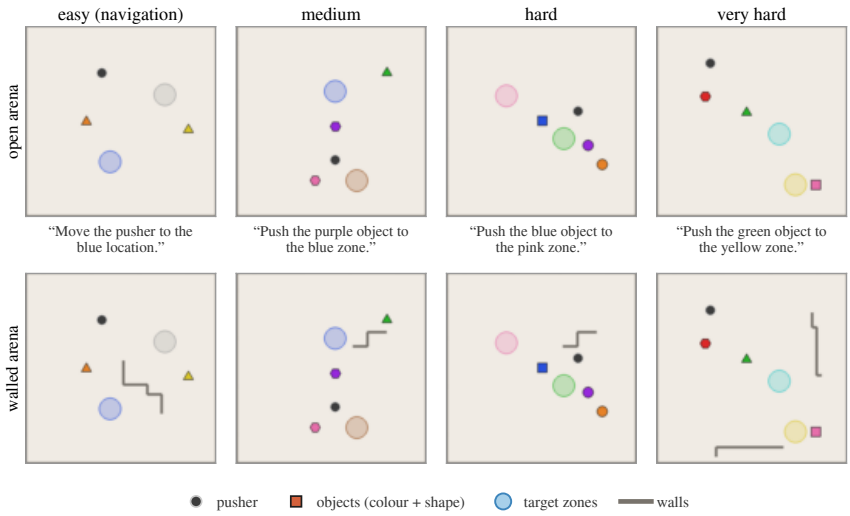


Figure 2: SLIM scenes for the four tiers (columns) in the open arena (top) and the identical configuration with procedurally generated wall chains added (bottom). The templated language goal of each scene is quoted under the top row; the easy tier asks only to move the pusher. Tiers differ in the geometry of pusher, target object and zone, not in object appearance.

through one of the eight symmetries of the square arena (quarter turns about the centre, optionally mirrored). This varies position and direction of every scene while keeping the template distances, and hence each tier's difficulty, fixed. A policy that ignores the goal and moves in one direction for a tuned number of steps solves 0.12 of easy and 0.20 of medium scenes (the symmetries map an axis-aligned template onto four directions, so one direction is aligned with about a quarter of the scenes) and 0 on hard and very hard (Table 1). Procedural wall chains are off in the headline results (Appendix B). Success requires, at the end of a 20-control-step episode, the pusher within 30 px of the target zone (easy) or the target object within 40 px of it (push tiers). Each scene comes with a templated natural-language goal ("push the purple object to the blue zone") and with a rendered goal frame, so visual-goal and language-goal planning are evaluated on identical scenes.

For training data we collect 10 000 episodes of 200 steps with a hand-coded primitive policy that mixes six manoeuvres (orbit, nudge, firm push, glancing hit, direction change, inter-object push) and never sees the goal.

Little of the frame responds to actions. Before any model is trained, a model-free probe already separates SLIM from PushT. Under a shared random-action distribution $\mathcal{N}(0, \sigma^2 I)$ clipped to $[-1, 1]^2$ at σ=0.5, 0.37% of the 224×224 frame changes per predictor step on SLIM against 0.64% on PushT (16 start states $\times$ 20 sequences of 25 steps; all bodies counted). The objects account for most of the gap: PushT's T-block is displaced by 27 px in 25 steps (root-mean-square, 512-px simulator frame), a SLIM object by 1.1 px in a training scene (three to six objects) and 0.54 px in a single-object scene, because a 14 px pusher rarely hits 15 px bodies. The fraction of static-object training transitions does not separate the two benchmarks (Appendix F); how much of the frame a moving body changes does.

4 An inverse-dynamics auxiliary loss for JEPA world models

Background: LeWM. A latent JEPA world model (Maes et al., 2026b) consists of an encoder f_θ with $z_t = f_\theta(x_t)$, an action encoder g_ϕ, and an autoregressive transformer predictor p_ψ that maps a context window of H latents and actions to the next latent, $\hat{z}_{t+1} = p_\psi(z_{t-H+1:t}, g_\phi(a_{t-H+1:t}))$. Training minimises the latent prediction error plus SIGReg, a regulariser toward an isotropic Gaussian that prevents representational collapse (plain stays collapsed at a third or three times its default

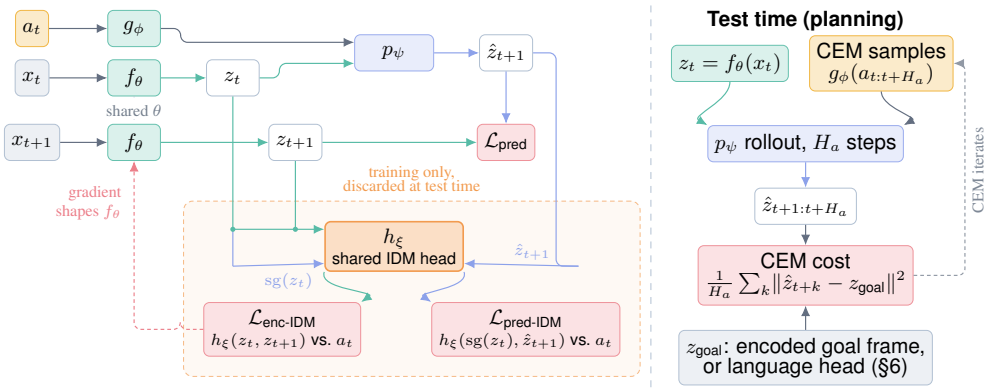


Figure 3: **LeWM+I.** *Left (training):* the LeWM backbone is unchanged — encoder f_θ, action encoder g_ϕ, predictor p_ψ, latent prediction loss $\mathcal{L}_{\text{pred}}$. We add one shared inverse-dynamics head h_ξ (orange), applied twice per batch with the same weights: to the encoder pair (z_t, z_{t+1}), whose gradient flows back into f_θ and forces consecutive latents to retain the action, and to the predictor pair $(\text{sg}(z_t), \hat{z}_{t+1})$, which forces the predicted latent to carry an action signature the same head can read. *Right (test):* the head is discarded; CEM rolls out encoder and predictor under sampled actions and minimises the latent goal cost averaged over the rollout, with z_{goal} from an encoded goal frame or, in Section 6, from the language head.

weight, Appendix H),

$$\mathcal{L}_{\text{vanilla}} = \underbrace{\tfrac{1}{H}\textstyle\sum_{h=1}^{H} \|\hat{z}_{t-H+h+1} - z_{t-H+h+1}\|^2}_{\mathcal{L}_{\text{pred}}} + \lambda_{\text{sigreg}}\, \mathcal{L}_{\text{sigreg}}(z). \tag{1}$$

At test time a cross-entropy-method (CEM) planner rolls sampled action sequences $a_{t:t+H_a-1}$ out in latent space and minimises the path cost $\frac{1}{H_a}\sum_{k=1}^{H_a} \|\hat{z}_{t+k} - z_{\text{goal}}\|^2$ against the encoded goal frame (the mean over the rollout; a terminal-only cost is far worse, Appendix A.1).

This suffices on PushT, where one large block moves whenever the pusher touches it. On SLIM the movable bodies contribute little to the frame-to-frame change that $\mathcal{L}_{\text{pred}}$ measures; we hypothesise that the encoder then drops the state of pusher and objects alike, so the predictor copies the current latent forward and the cost surface goes flat (Section 5.4).

The loss. We add one auxiliary objective: a shared inverse-dynamics head $h_\xi : \mathbb{R}^{2D} \to \mathbb{R}^{|a|}$, with D the latent width, is a 3-layer MLP (width 256, $\sim$ 167K parameters) that predicts the frameskip-packed action from a pair of latents; it is applied twice per batch with the same weights.

$$\mathcal{L}_{\text{enc-IDM}} = \|h_\xi(z_t, z_{t+1}) - a_t\|^2, \tag{2}$$

$$\mathcal{L}_{\text{pred-IDM}} = \|h_\xi(\text{sg}(z_t), \hat{z}_{t+1}) - a_t\|^2. \tag{3}$$

Encoder-IDM (Eq. (2)) back-propagates through f_θ and forces consecutive encoder latents to retain whatever content is needed to recover the action; it says nothing about the predictor. *Predictor-IDM* (Eq. (3)) requires the predicted latent to carry an action-decodable signature that the *same* head can read, so the predictor cannot write action information into a private subspace the encoder does not occupy. sg is a stop-gradient, so Eq. (3) updates p_ψ and h_ξ but not f_θ; the head is anchored by Eq. (2) on real latent pairs (Appendix A). The full objective is

$$\mathcal{L}_{\text{I}} = \mathcal{L}_{\text{pred}} + \lambda_{\text{sigreg}}\mathcal{L}_{\text{sigreg}} + \alpha_{\text{inv}}\mathcal{L}_{\text{enc-IDM}} + \alpha_{\text{pid}}\mathcal{L}_{\text{pred-IDM}}, \tag{4}$$

with $(\lambda_{\text{sigreg}}, \alpha_{\text{inv}}, \alpha_{\text{pid}}) = (0.09, 1, 1)$ held fixed for both benchmarks and every seed. The head is discarded at test time, so planning uses exactly the vanilla LeWM modules; the head adds under 1% of the parameters. We call the result LeWM+I, and the unrepaired baseline *plain*.

	SLIM (three training seeds, both horizons)					PushT	
model	easy	medium	hard	very hard	trial mean	H_a=5	H_a=10
LeWM (plain)	0.007	0.000	0.003	0.000	0.003 ± 0.001	0.98 ± 0.01	0.53 ± 0.03
LeWM+I (ours)	**0.990**	**0.243**	**0.157**	0.000	$\mathbf{0.348 \pm 0.016}$	0.97 ± 0.01	$\mathbf{0.83 \pm 0.01}$
random or zero-action policy	0.00	0.00	0.00	0.00	0.00	—	—
goal-agnostic, one direction (tuned in hindsight)	0.12	0.20	0.00	0.00	0.08	—	—
scripted controller, privileged state	1.00	1.00	1.00	0.28	0.82	—	—

Table 1: **Headline results.** Per-tier values pool three training seeds and both horizons (300 trials each); trial-mean and PushT columns are mean ± SEM over seeds (McNemar plain vs. LeWM+I, canonical seed: $p < 10^{-43}$; per-seed values in Appendix A). Reference rows use the same scenes, budgets and thresholds: random or zero actions, the best single direction and duration chosen per tier in hindsight (it never reads the goal), and a scripted controller with simulator state (Appendix E).

5 EXPERIMENTS

5.1 SETUP

Unless stated otherwise, encoder, predictor, optimiser and planner are identical across every cell and both benchmarks; only the I switch, the dataset and the seed vary. The encoder is a ViT-Tiny (5.5M parameters) trained *from scratch*, the predictor a 6-layer transformer with context $H = 3$, ~ 18M trainable parameters in total; full hyper-parameters are in Appendix A. We report the final training checkpoint (epoch 30). PushT uses the standard 10K-episode release (Chi et al., 2023); actions are frameskip-packed (one predictor step = 5 environment steps). The planner is CEM with action horizon $H_a \in \{5, 10\}$ predictor steps (H_a=5 is the training horizon), replanning after each frameskip block on SLIM and after the full plan on PushT, following PushT's published protocol and, on SLIM, the cadence under which LeWM+I plans best (Appendix A.1 swaps them). A SLIM number is a *trial mean*: 50 evaluation seeds × 4 tiers × 2 planner horizons, $N = 400$ paired trials per model (200 scenes at two horizons), compared by exact McNemar tests between two trained models.

5.2 PLANNING RESULTS

On SLIM (Table 1) plain fails uniformly: 3 of 1200 trials across three training seeds. LeWM+I reaches 0.348 ± 0.016; navigation saturates at 99% and very hard floors at zero for both, so the pushing tiers carry the comparison. As evidence of goal-directed pushing we use the hard tier, where success rises from 0 to 0.16 and a policy that ignores the goal scores 0; on medium LeWM+I reaches 0.24, but a single movement direction chosen in hindsight already reaches 0.20 (Section 3), so medium says little about goal-directed pushing. A scripted controller with simulator state solves easy, medium and hard in full and very hard in 0.28 of trials within the budget (1.00 with 60 steps; Appendix E), so the pushing ceiling lies in the learned model and its cost, and the very-hard zero partly in the budget. On PushT both models saturate at the trained horizon, but at twice that horizon plain drops to 0.53 while LeWM+I holds 0.83 (no seed overlap).

5.3 ABLATIONS: WHICH GRADIENT DOES THE WORK?

Table 2 isolates which component carries the effect with ablation cells that share data and training budget.

Encoder-IDM is load-bearing. Predictor-IDM alone yields $4/400$ successes, indistinguishable from plain (McNemar $p = 0.13$). Encoder-IDM alone reaches 0.298 ± 0.005 over three training seeds, about 85% of the full gain over plain (both three-seed); predictor-IDM adds the remaining 5 pp on top of it (canonical seed $+7.8$ pp, McNemar $p < 10^{-6}$, on medium and hard), so it is a complement that needs the encoder term, not a carrier on its own. *The gradient, not the head, matters.* In the shuffled-action null the IDM targets are permuted across the minibatch (consistent latent pairs, wrong actions): both IDM losses stay at the action variance and the model stays at plain's level (0.002 ± 0.001 over three seeds) at fixed architecture and compute. *A stronger frozen encoder is not a substitute.* With a frozen DINOv2-S encoder (Oquab et al., 2024; Zhou et al., 2025), I lifts success

cell	α_{inv}	α_{pid}	encoder	trial mean	vs. plain (McNemar)
plain	0	0	ViT-Tiny, trained	0.000	—
predictor-IDM only	0	1	ViT-Tiny, trained	0.010	$+1.0$ pp ($p = 0.13$)
encoder-IDM only	1	0	ViT-Tiny, trained	0.290	$+29.0$ pp ($p < 10^{-34}$)
shuffled-action null	1	1	ViT-Tiny, trained	0.003	$+0.3$ pp ($p = 1$)
frozen DINOv2-S, no I	0	0	DINOv2-S, frozen	0.048	$+4.8$ pp
frozen DINOv2-S + I	1	1	DINOv2-S, frozen	0.163	$+16.3$ pp †
LeWM+I (full)	1	1	ViT-Tiny, trained	**0.368**	$+36.8$ pp ($p < 10^{-43}$)

Table 2: **Disentangle.** Canonical training seed (the highest of three for LeWM+I), on which McNemar pairs trials; three-seed means are 0.003 (plain), 0.348 (LeWM+I), 0.298 (encoder-IDM) and 0.002 (shuffled null), per-seed values in Appendix A. †Against frozen DINOv2-S without I: $+11.5$ pp, $p < 10^{-9}$.

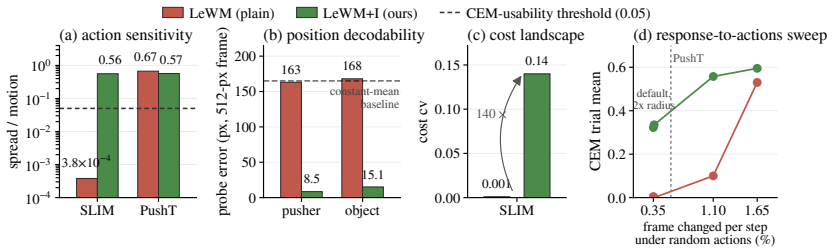


Figure 4: **Mechanism probes** on SLIM (and PushT for reference), LeWM vs. LeWM+I. (a) Plain's latent is almost action-insensitive, two orders of magnitude below the usability threshold and three below LeWM+I; PushT-plain is action-sensitive without I. (b) Pusher and object positions cannot be decoded from plain's latent (dashed line: constant mean-position predictor) and are recovered to within one patch by LeWM+I; errors in the 512-px simulator frame. (c) Plain's CEM cost surface is flat; LeWM+I restores a usable descent direction. (d) Planning success against the fraction of the frame that changes per step under random actions, over four environment variants (Appendix G): plain stays near zero until enough of the frame responds.

from 0.048 to 0.163 but stays below half of LeWM+I (easy 0.50 vs. 1.00): the gradient shapes the thin $384 \to 192$ projector, not the frozen backbone.

5.4 Mechanism analysis

Four model-internal probes need no environment interaction. **Action sensitivity**: push $K = 32$ random actions through the predictor from each held-out z_t and report spread/motion $= \sqrt{\text{Var}_k[\hat{z}^{(k)}_{t+1}]} \,/\, \sqrt{\|\hat{z}^{(k)}_{t+1} - z_t\|^2}$, the variance of the predicted latent across the K actions over its mean squared displacement from z_t (both per latent dimension, averaged over held-out contexts), a scale-free ratio that drift cannot fake. **Rollout error**: prediction error over a 10-step rollout divided by the true per-step latent motion; copying the latent forward scores on the order of the rollout length. **Position decodability**: pixel error of an MLP probe from z_t to pusher and object positions. **Cost landscape**: coefficient of variation (cv) of the CEM cost over 1000 random action sequences from a start state, averaged over 200 start states (the 50 evaluation seeds $\times$ 4 tiers).

Figure 4 (numbers in Table 4, appendix) summarises the probes. On SLIM-plain the latent is action-insensitive (spread/motion 3.8×10^{-4}), the 10-step rollout error is $12.4\times$ the per-step true motion, on the order of copying the current latent forward (about the rollout length, 10), the pusher position is undecodable (163 px, at the $\approx$165 px of a constant mean-position predictor), and the cost surface is flat (cv $= 0.001$). LeWM+I restores all four: spread/motion 0.56–0.61 across two seeds, rollout-error ratio 1.6–1.7, pusher decodable to 8.5 px (within one patch), cv $= 0.14$, and its latent cost ranks sampled actions like the simulator (Spearman 0.84, Appendix I). PushT-plain has none of the failure modes (spread/motion 0.67; its predicted latent moves as far as the true one), consistent with plain planning well on PushT at the trained horizon.

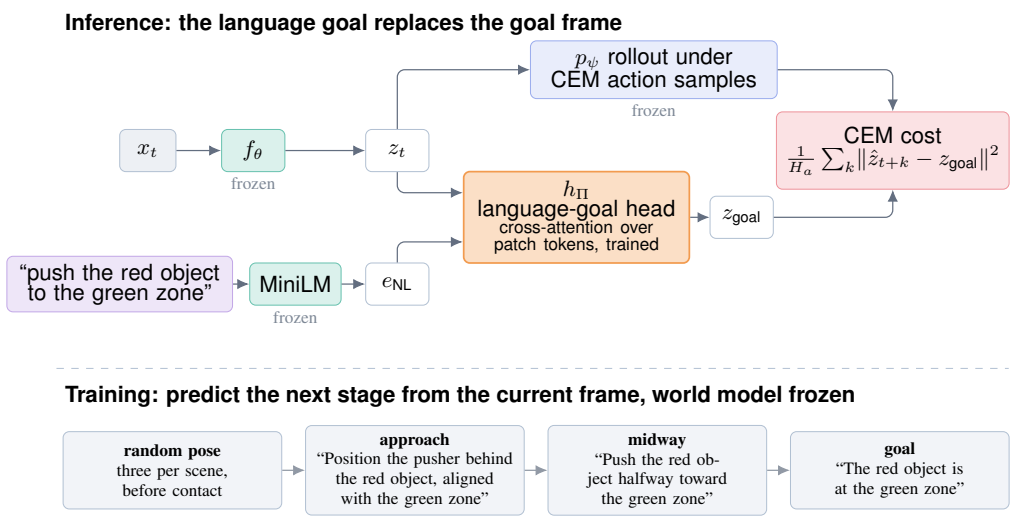


Figure 5: **Language-goal head.** *Top (inference):* a small trained head maps the frozen encoder's z_t and a frozen sentence embedding of the goal description to a goal latent z_{goal}, which replaces the encoded goal frame in the CEM cost; encoder, predictor and planner are unchanged. *Bottom (training):* from each frame the head predicts the encoded latent of the next stage of a push, given that stage's sentence; at test time it is always asked for the final goal.

Action sensitivity is necessary, not sufficient. Across the twenty-eight configurations we trained (Fig. 6, appendix), every one below spread/motion ≈ 0.05 fails, but above it success ranges from 0.01 to 0.83, and varying only the loss weight reproduces this inside one recipe (Appendix H). The probe is a gate on plannability, not a ranking of planners.

The dose is how much of the frame responds to actions. We vary it at fixed data volume and training budget (Fig. 4d; Appendix G) and measure it model-free, as the fraction of the frame that changes per predictor step under the environment's random actions. Doubling the object radius quadruples the footprint, but the objects still rarely move and the changing fraction stays at 0.36% (default 0.35%): plain stays collapsed (0 of 400 trials, spread/motion 3×10^{-4}) and LeWM+I plans as before (0.335 vs. 0.323). Making the scene respond, with lighter, less damped objects and a larger, faster pusher, raises the fraction to 1.10% and 1.65%: plain's encoder no longer collapses (spread/motion 0.88 on both arms) and it starts to plan (0.100 and 0.530), while LeWM+I stays ahead (0.558 and 0.595). PushT, where plain needs no help, lies at 0.57%. Within SLIM, the vanilla encoder collapses only where too little of the frame responds; larger or faster-moving bodies can both raise that fraction, enlarging the objects alone did not, so what matters is the fraction rather than object size; the auxiliary loss supplies the action signal the frames lack. Distractor-free training, walls, ViT-Small and OGBench: Appendices B to D and J.

6 Language-conditioned goals

6.1 Language-goal head

Head. A small head $h_\Pi(z_t, e_{\text{NL}}) \mapsto z_{\text{goal}}$ (Fig. 5) produces the goal latent for the planner's cost; e_{NL} is the embedding of the goal sentence from a frozen `all-MiniLM-L6-v2` encoder. Two cross-attention layers let a query from z_t and one from e_{NL} attend over the encoder's 256 patch tokens (1.1M parameters, Appendix L). The head is trained on $40\,000$ simulator scenes, disjoint from the evaluation seeds, each rendered as six frames along a push with one sentence per stage (Fig. 5); from each frame it predicts the encoded latent of the *next* stage. At test time it always receives the final goal sentence ("The red object is at the green zone"), a pairing with the start frame it never saw in training. It is queried twice per 20-step episode with the same sentence, at the start

goal channel	easy	medium	hard	very hard	trial mean	Δ vs. oracle (McNemar)
goal-agnostic, one direction (hindsight)	0.12	0.20	0.00	0.00	0.08	—
visual-goal oracle	1.00	0.30	0.17	0.00	0.368	—
goal-frame-conditioned head (MLP)	0.99	0.29	0.17	0.00	0.363	-0.5 pp ($p = 0.80$)
language head over the global latent (MLP)	0.53	0.16	0.09	0.00	0.195	-17.3 pp ($p < 10^{-11}$)
language head, patch attention, 3 seeds	**0.84**	0.06	0.04	0.00	$\mathbf{0.235 \pm 0.026}$	-13.3 pp ($p < 10^{-9}$ each seed)
zero sentence embedding	0.00	0.00	0.01	0.00	0.003	vs. head: $p < 10^{-27}$
wrong colour word	0.10	0.00	0.00	0.00	0.025	vs. head: $p < 10^{-24}$
target fixed at t=0	0.74	0.00	0.04	0.00	0.195	vs. head: $p < 10^{-3}$
paraphrases, train pool	0.84	0.04	0.02	0.00	0.225	vs. head: $p = 0.15$
paraphrases, unseen nouns	0.78	0.01	0.03	0.00	0.205	vs. head: $p = 0.005$

Table 3: **Language goals** on the LeWM+I world model of the canonical seed, whose visual-goal planner is the oracle (per-tier values average both horizons; $\pm$: SD over head seeds). Indented rows ablate head seed 42 (0.245), against which their p-values are computed; per-seed values in Appendix A.

and again after 10 planner steps from the then-current latent, so the goal latent is refreshed from the new state; the planner uses the latest output in between. Evaluation withholds the goal frame; the visual-goal planner of Table 1 on the same world model is the *oracle*.

6.2 Results

The head grounds the named zone. Attending over the encoder's patch tokens, the head reaches 0.84 on navigation against the oracle's 1.00 and 0.235 ± 0.026 over all tiers (Table 3). It depends on the sentence (zeroing its embedding leaves $1/400$), and naming a colour that appears nowhere in the scene drops it to 0.025. To test whether it reads the colour or has learned where targets usually lie relative to the pusher, we swap the named zone with the other zone on the navigation tier, so that a zone of the wrong colour sits at the typical target position. The head never ends at this decoy, ends closer to the named zone in all 300 episodes (50 scenes, 3 seeds, 2 horizons), and succeeds as often as without the swap (252 vs. 253): it goes where the sentence says, not where targets usually are. An MLP head conditioned on the goal frame instead of the sentence matches the oracle (0.363 vs. 0.368, $p = 0.80$), so the training pipeline and planner are not the bottleneck; the gap lies in mapping a sentence, which specifies a set of goal states, to one goal latent. An MLP language head over z_t trades navigation (0.53) for pushing (medium 0.16, hard 0.09) and is less reliable under the swap.

Language degrades gracefully under paraphrase. On seed 42 the head keeps 92% of its success on 107 paraphrases ($p = 0.15$) and 84% on 31 held-out templates with unseen object and zone nouns ($p = 0.005$), mostly on navigation. One query at t=0 already reaches 0.195; a mid-episode refresh adds 5 pp ($p < 10^{-3}$).

Pushing from language is the open part. On the pushing tiers the head reaches about a fifth of the oracle (medium 0.06, hard 0.04 vs. 0.30 and 0.17) and stays below the goal-agnostic policy on medium. Choosing which object to push toward which zone from a sentence, and binding colours to object and zone, remain open on SLIM.

7 Conclusion

When little of the frame responds to actions, a JEPA world model collapses before language enters: an encoder that discards pusher and objects still meets the prediction objective. One inverse-dynamics auxiliary loss on encoder and predicted latents restores a plannable latent at no test-time cost, our controls attribute the repair to the gradient into the encoder, and the action-sensitivity probe offers a cheap check before planning. On the repaired latent a small head plans from sentences, approaches the goal-frame oracle on navigation, and goes to the zone the sentence names even when a decoy occupies the usual target position.

SLIM makes the next steps measurable: pushing from language, where the head must choose which object to move toward which zone, and carrying the repair to larger encoders, whose probes recover but whose planning did not yet transfer.

AI USE STATEMENT

In this work, we used generative AI tools (LLM-based coding assistants) to assist in implementing methods, probes and evaluation code, to clean and reformat datasets (memory-mapped conversion of the rendered episodes) and to provide feedback on research methodology and experiments (including suggesting control experiments and the response-to-actions sweep). We have not used generative AI tools to generate datasets: all training and evaluation data are produced by the seeded physics simulator of Section 3. Developing theoretical models, formulating or proving mathematical claims, translation, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for drafting and editing parts of the paper, creating and modifying figures and plotting scripts, searching and summarising related literature, formatting references, and proposing titles. We have reviewed all AI-assisted work: code was checked by the authors and validated throughout, every reported number was traced to an evaluation log, and AI-suggested related work was verified against the original papers. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses only synthetic 2D simulation data and public benchmark releases; it involves no human subjects, personal data or deployment. We see no direct negative societal impact beyond those generic to robot learning.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix A specify the benchmark (simulator constants, tier generator, success thresholds, the seeded data-collection policy and the train/validation split), the model and optimiser, and the planner protocol (path cost, 20 planner steps, 50 fixed evaluation seeds per tier shared by every cell); Appendix F characterises the training data, Appendix E the scripted reference controller, Appendix G the physics overrides of the response sweep, and Appendices K and L the language head, its curriculum, its paraphrase pools and the grounding controls. Per-seed values, the statistical tests and their pairing are in Appendix A. The code (environment, seeded dataset generator, training and evaluation code, probes, language templates and paraphrase pools) will be released with the camera-ready version; an anonymised copy is provided as supplementary material.

REFERENCES

Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, 2016.

Guo An, Zijing Wu, Honghua Dong, Yuhao Yan, Zixuan Gui, Haochong Chen, Shanzhao Ruan, Xiang Wang, Yurong Ling, and Qi Tian. Diagnosing JEPA world models with action-conditioned predictive consistency. *arXiv preprint arXiv:2608.12939*, 2026.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *CVPR*, 2023.

Mahmoud Assran et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Jiaxin Bai and Jiaxuan Xiong. Temporal-distance JEPA: Plan-aware representation learning for latent world model predictive control. *arXiv preprint arXiv:2607.25337*, 2026.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

Samuel Barbeau, Simon Roy, Giovanni Beltrame, Christian Desrosiers, and Nicolas Thome. Latent goal prediction from language for model-based planning. *arXiv preprint arXiv:2606.20627*, 2026.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. In *International Conference on Learning Representations*, 2024.

Jack Boylan and Chris Hokamp. No gaussian required: Contrastive inverse dynamics for JEPA world models. *arXiv preprint arXiv:2608.17542*, 2026.

Amirhosein Chahe and Lifeng Zhou. Policy-guided world model planning for language-conditioned visual navigation. *arXiv preprint arXiv:2603.25981*, 2026.

Jiaheng Chen. ATM: Action-consistency transfer matrix for diagnosing and improving latent world models. *arXiv preprint arXiv:2606.09028*, 2026.

Letian Cheng, Qi Zhang, and Yisen Wang. SAGE: Subgoal-conditioned action generation for latent world model planning. *arXiv preprint arXiv:2607.17973*, 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

Tal Daniel, Carl Qi, Dan Haramati, Amir Zadeh, Chuan Li, Aviv Tamar, Deepak Pathak, and David Held. Latent particle world models: Self-supervised object-centric stochastic dynamics modeling. In *ICLR*, 2026. URL https://arxiv.org/abs/2603.04553.

Yonathan Efroni, Dipendra Misra, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Provably filtering exogenous distractors using multistep inverse dynamics. In *ICLR*, 2022.

Zhenfeng Gan, ZiTong Zeng, Jiajun Cheng, Yeke Song, Yongyi Tang, and Xueqian Wang. ActSWM: Action-sensitive world models for long-horizon planning in open-world games. *arXiv preprint arXiv:2607.26712*, 2026.

David Ha and Jürgen Schmidhuber. World models. In *NeurIPS*, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Riashat Islam, Manan Tomar, Alex Lamb, Yonathan Efroni, Hongyu Zang, Aniket Didolkar, Dipendra Misra, Xin Li, Harm van Seijen, Remi Tachet des Combes, and John Langford. Agent-controller representations: Principled offline RL with rich exogenous information. In *International Conference on Machine Learning*, 2023.

Petr Ivashkov, Randall Balestriero, and Bernhard Schölkopf. Sensorimotor world models: Perception for action via inverse dynamics. *arXiv preprint arXiv:2606.20104*, 2026.

Alex Lamb, Riashat Islam, Yonathan Efroni, Aniket Didolkar, Dipendra Misra, Dylan Foster, Lekan Molu, Rajan Chari, Akshay Krishnamurthy, and John Langford. Guaranteed discovery of control-endogenous latent states with multi-step inverse models. *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2207.08229.

Yann LeCun. A path towards autonomous machine intelligence. *OpenReview*, 2022.

Alexander Levine, Peter Stone, and Amy Zhang. Multistep inverse is not all you need. *arXiv preprint arXiv:2403.11940*, 2024.

Quanyi Li, Lan Feng, Haonan Zhang, Wuyang Li, Letian Wang, Alexandre Alahi, and Harold Soh. Grounded world model for semantically generalizable planning. *arXiv preprint arXiv:2604.11751*, 2026a.

Wenyuan Li, Guang Li, Keisuke Maeda, Takahiro Ogawa, and Miki Haseyama. Predictive but not plannable: RC-aux for latent world models. *arXiv preprint arXiv:2605.07278*, 2026b.

Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. Learning to model the world with language. In *ICML*, 2024.

Muyuan Liu, Yue Huang, Zheng Liang, and Xiang Gao. Toward physically grounded JEPA world models for goal-conditioned robotic planning. In *IROS Workshop on Physical World Models for Scaling Embodied AI (PWMS)*, 2026. URL https://arxiv.org/abs/2609.03565.

Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. In *Robotics: Science and Systems*, 2021.

Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023. URL https://arxiv.org/abs/2210.06407.

Lucas Maes, Quentin Le Lidec, Luiz Facury, Nassim Massaudi, Ayush Chaurasia, Francesco Capuano, Richard Gao, Taj Gillin, Dan Haramati, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel: A platform for reproducible world modeling research and evaluation. *arXiv preprint arXiv:2605.21800*, 2026a.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026b.

Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 2022a.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 2022b. URL https://arxiv.org/abs/2112.03227.

Suraj Nair, Eric Mitchell, Kevin Chen, Brian Ichter, Silvio Savarese, and Chelsea Finn. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, 2021.

Heejeong Nam, Quentin Le Lidec, Lucas Maes, Yann LeCun, and Randall Balestriero. Causal-JEPA: Learning world models through object-level latent masking. In *ICML*, 2026. URL https://arxiv.org/abs/2602.11389.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024.

Seohong Park et al. Ogbench: Benchmarking offline goal-conditioned rl. *arXiv preprint arXiv:2410.20092*, 2024.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. *arXiv preprint arXiv:2502.14819*, 2025.

Jonathan Spieler, Angel Villar-Corrales, and Sven Behnke. Slot-MPC: Goal-conditioned model predictive control with object-centric representations. *arXiv preprint arXiv:2605.14937*, 2026.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *Transactions on Machine Learning Research*, 2026. URL https://arxiv.org/abs/2512.24497.

Leonardo F. Toso, Davit Shadunts, Yunyang Lu, Nihal Sharma, Donglin Zhan, Nam H. Nguyen, and James Anderson. Learning invariant visual representations for planning with joint-embedding predictive world models. *arXiv preprint arXiv:2602.18639*, 2026.

Jiawei Wang, Ke Rui, Yushen Zuo, Yichun Feng, and Minglei Li. Decision-metric alignment in latent world models: Diagnostics and action-conditioned objectives for MPC planning. *arXiv preprint arXiv:2608.18746*, 2026a.

Tongzhou Wang, Simon S. Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised MDPs: Learning world models better than the world itself. In *ICML*, 2022.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim G. J. Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning. In *ICML*, 2026b. URL `https://arxiv.org/abs/2603.12231`.

Jewon Yeom, Hanseul Kim, Jeongjae Park, Sungmok Jung, Jaejin Lee, and Taesup Kim. What makes video world model latents action-relevant: Prediction over reconstruction. *arXiv preprint arXiv:2606.07687*, 2026.

Xi Zeng, Haojie Ren, and Ziying Song. PhyLatent: Learning dynamics-relevant representations for JEPA world models. *arXiv preprint arXiv:2608.05720*, 2026.

Tianqiu Zhang, Muyang Lyu, Yufan Zhang, Fang Fang, and Si Wu. DiLA: Disentangled latent action world models. *arXiv preprint arXiv:2605.15725*, 2026a.

Zhenghao Zhang, Yuanxiang Wang, Zhenyu Guan, Yujia Yang, Bingkang Shi, Tianyu Zong, Hongzhu Yi, Guoqing Chao, Xingchen Chen, Tiankun Yang, Chenxi Bao, Tao Yu, Jingjing Zhou, and Jungang Xu. Delta-JEPA: Learning action-sensitive world models via latent difference decoding. *arXiv preprint arXiv:2606.31232*, 2026b.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. In *ICML*, 2025.

A Additional experimental details

Environment constants. SLIM is simulated at 512×512 internal resolution and rendered at 224×224. Pusher: radius 14 px, mass 5, PD gains K_p=100, K_v=20, action scale 40 (calibrated to PushT's per-step pixel displacement), velocity cap 100 px/s; the cap binds for action norms above about 0.45, where the pusher moves 6–6.6 px per environment step and the action mainly sets the direction of motion. Objects (1–3 per scene, 0–2 on the navigation tier; two zones per scene): radius 15 px, mass 8, global damping 0.5, per-substep object damping 0.85. One environment step is ten physics substeps at 10 Hz; one control (planner) step is five environment steps (frameskip). Tier configurations place pusher, target object and target zone by the tier's template, translate the template uniformly within the arena (all three at least 80 px from the border), draw each distractor object and zone as the corresponding point of an independent, translated and rotated template of the same tier (rejecting overlaps), and then map the whole scene through one of the eight symmetries of the square arena, drawn per (seed, tier) from a separate random stream. On medium and hard the target object is then the most central object about as often as chance (0.44 and 0.43 vs. 0.42 and 0.41 over 500 seeds); on very hard, whose template spans the arena diagonal and leaves little room for translation, it is the most central in 0.60 of scenes (chance 0.42), a residual cue on a tier that no planner solves. Walls, when enabled, are chains on a 40×40 grid with extension probability 0.9 and decay 0.1, traversability guaranteed. Zero action produces exactly zero pusher motion, which the action-sensitivity probe relies on. The environment's own termination threshold is 25 px; evaluation success uses 30 px (pusher) and 40 px (object) to account for residual end-state distance after a closed-loop rollout, identically for every model.

Planner. All PushT training and evaluation uses the stable-worldmodel platform (Maes et al., 2026a). CEM with 30 iterations $\times$ 300 samples and top-30 elites on PushT, 20 iterations $\times$ 200 samples and top-20 elites on SLIM, variance scaling 1.0, no momentum; the cost is the squared distance to the goal latent averaged over the H_a predicted latents of the rollout (path cost); 20 planner steps per episode, each executed as five environment steps.

Success semantics. Success is scored on the final step of a fixed-length episode. Scoring "ever within threshold" (checked after each executed block) instead raises LeWM+I by $+1.3$ pp on the easy/medium/hard cells ($147 \rightarrow 151$ of 300 paired trials, discordant pairs $0/4$, McNemar $p = 0.125$; three on medium, one on hard); per-episode minimum distances are stored, so either reading is reproducible.

Data. 10 000 episodes; training clips are split 90/10 at random into training and validation, identically for every variant (clips of one episode can fall on both sides, so the validation loss is a training diagnostic rather than a held-out measure); frames are resized to 224 px and normalised as in LeWM, without augmentation, and no data-side choice is modified for LeWM+I. Training episodes contain 3–6 objects and 1–3 zones (evaluation scenes: 1–3 objects, two zones). Inter-object pushes, in which the pusher moves one object that displaces another, occur in about 8% of frames.

Model. Encoder: ViT-Tiny, 192 hidden, 12 layers, 3 heads, patch 14 (16×16 tokens at 224 px), random initialisation. Predictor: 6 layers, hidden 192, 16 heads, MLP 2048, dropout 0.1, context $H{=}3$, supervised at all context positions. Action encoder: linear $\mathbb{R}^2 \rightarrow 192$. IDM head: MLP $384 \rightarrow 256 \rightarrow 256 \rightarrow 10$ (frameskip 5 $\times$ 2-dim actions). AdamW lr 5×10^{-5}, weight decay 10^{-3}, gradient clipping 1.0, batch 128, bf16. Training 30 epochs; the final checkpoint (epoch 30) is evaluated, without peak-epoch selection; the one exception is a plain seed that diverged (below). A SLIM evaluation of 20 episodes takes 3–4 minutes on one A100; a 50-episode PushT evaluation about 8 minutes.

Inverse-dynamics head. The encoder-side term is applied to every consecutive latent pair of a training window (three pairs per window at context 3), the predictor-side term to the last context position only (one pair per window); both use unit weight, and the head is updated by both (the stop-gradient in Eq. (3) is on z_t only). In the shuffled-action null the latent pairs are kept and the action targets are permuted across the minibatch (for the encoder term, across minibatch and time steps), so the head receives inconsistent targets; the null therefore bounds what head capacity and compute contribute, and a permuted target can at most push the encoder away from action information. The training seed fixes the train/validation split; parameter initialisation and dropout are not tied to it, so repeated runs with the same seed value differ in initialisation.

Language head training. AdamW, MSE $+0.1\times$ cosine loss, 90 epochs for the cross-attention head and 30 for the MLP head; the world model is frozen and the training scenes are encoded once.

Per-seed headline values. SLIM trial mean: LeWM+I 0.368 / 0.360 / 0.315 (training seeds 3072, 9999, 42), plain 0 / 2 / 1 of 400 (seeds 3072, 1234, 5678). Plain seed 5678 diverged to non-finite weights in its 28th training epoch; we report its last finite checkpoint (epoch 27). McNemar tests pair the 400 evaluation trials (identical scenes and seeds) of two models; training seeds need not coincide; PushT $H_a{=}10$ (seeds 3072, 1234, 5678 for both models): plain 0.58 / 0.52 / 0.48, LeWM+I 0.82 / 0.84 / 0.82. Encoder-IDM only: 0.290 / 0.308 / 0.298 (seeds 3072, 9999, 42); shuffled-action null: 0.003 / 0.000 / 0.003 (seeds 3072, 9999, 42). Language head (Table 3): trial means 0.245 / 0.205 / 0.255 (head seeds 42, 7, 11; easy 0.88 / 0.78 / 0.87), McNemar against the visual-goal oracle $p = 1.5 \times 10^{-12}$, 5.1×10^{-17} and 2.1×10^{-10}.

Probe values. Table 4 lists the probe values behind Fig. 4.

Other auxiliary objectives. We also trained two other auxiliaries on SLIM: a residual predictor (R) and a patch decoder (P), alone and combined. Every cell without I kept plain-level probes (spread/motion 5.8×10^{-3}, 1×10^{-4} and 6×10^{-4} for R, P and R+P; pusher decodability 166 px), whereas every cell with I restored them (I 0.56, I+P 0.54, R+I+P 0.70 spread/motion; pusher decodability 8.5, 7.4 and 5.3 px). P is a small refinement on top of I and R is redundant, so the paper uses I alone.

A.1 Planner cadence control

The benchmarks inherit different receding-horizon cadences from their upstream pipelines. Running SLIM with PushT's full-plan cadence lowers LeWM+I from 0.490 to 0.390 trial mean on the six

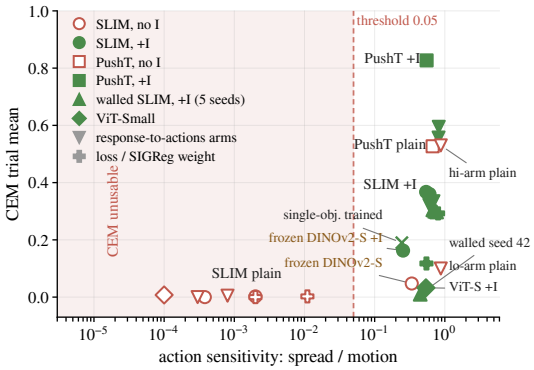


Figure 6: Action sensitivity vs. CEM success for 28 trained configurations (circles SLIM, squares PushT, triangles walled SLIM, diamonds ViT-Small, downward triangles the response-to-actions arms, plus signs the loss-weight and SIGReg cells; filled with I, hollow without; the cross is LeWM+I trained on the single-object variant and evaluated cross-distribution). Below 0.05 every planner fails; above it success ranges from 0.01 to 0.83: the frozen-DINOv2 cells, the non-converged walled seed and ViT-Small clear the threshold yet plan poorly.

probe	SLIM plain	SLIM LeWM+I	PushT plain	PushT LeWM+I
action sensitivity, spread/motion	3.8×10^{-4}	0.56 / 0.61	0.67	0.57
rollout error / true motion (h=10)	12.4	1.69 / 1.62	1.02†	0.95†
pusher position MLP error (px)	163 ± 2	8.5 ± 0.1	—	—
object position MLP error (px)	168 ± 0.4	15.1 ± 0.2	—	—
cost-landscape cv	0.001	0.14	—	—

Table 4: Mechanism probes. †For PushT only the predicted-over-true latent displacement was measured, not the error ratio, so these two entries are not comparable with the SLIM column. Two values for SLIM LeWM+I are two training seeds (within 10% on spread/motion, 5% on the rollout-error ratio). Position errors (mean $\pm$ SEM over three probe seeds, 80/10/10 split) are in the 512-px simulator frame, where a constant mean-position predictor scores ≈ 165 px.

easy/medium/hard cells ($147/300 \rightarrow 117/300$, McNemar $p = 1.4 \times 10^{-6}$; losses concentrated on medium and hard); running PushT with SLIM's replan-every-block cadence crashes both models (H_a=10: LeWM+I $0.83 \rightarrow 0.20$, plain $0.53 \rightarrow 0.18$). Each benchmark is most usable under its native cadence, the mismatch is symmetric, and the LeWM+I-vs-plain gap keeps its sign under both.

Horizon and cost mode. On PushT the I advantage vanishes at $H_a = 15$ (both 0.04), three times the trained horizon. Path cost (the mean over the H_a predicted latents) beats terminal cost: the r=30 checkpoint of Appendix G scores 0.335 under path cost and 0.083 under terminal cost over 400 paired trials (McNemar $p < 10^{-21}$). Compounding predictor error limits the very-hard tier; within LeWM+I the cost-landscape cv is 0.16–0.17 on easy, medium and hard but 0.07 on very hard.

B Robustness to obstacle walls

We regenerated the dataset with procedural walls (10K episodes; the wall generator turns chains toward increasing coordinates in 82% of turns) and retrained plain (one seed) and LeWM+I (five seeds) for 30 epochs. Walled plain (seed 3072): $2/400$. Walled LeWM+I: $0.310, 0.298, 0.300, 0.305$ (seeds 3072, 9999, 7, 11; McNemar vs. plain $p < 10^{-30}$ each) and 0.010 (seed 42). Conditional on convergence the four seeds cluster at 0.303 ± 0.006, close to the open arena's pooled 0.348; unconditionally 0.245 ± 0.131 with a 1-in-5 failure (this and the other $\pm$ values in this appendix are SDs across seeds) (Jeffreys 95% interval on the failure rate $[1.3\%, 62\%]$). Convergence is defined by a training-side criterion fixed before evaluation: validation encoder-IDM loss below 0.5 at the last epoch (converged seeds 0.19–0.28, seed 42: 0.67). Two of five walled runs show a validation encoder-IDM spike, at epochs 14 and 21 (seed 42) and at epoch 17 (seed 9999, peak 3.9); one

recovers, one does not (Fig. 7). Per tier over the four converged seeds (mean $\pm$ SD): easy 0.90 ± 0.02, medium 0.27 ± 0.03 (open arena 0.24), hard 0.04 ± 0.02, very hard 0; hard drops because wall chains obstruct the steered pushes. Each McNemar test contrasts one LeWM+I training instance with one plain instance on the same 400 trials, so it tests that pair of models, not the recipe distribution across training seeds.

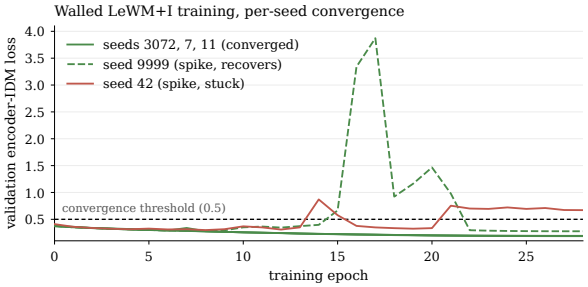


Figure 7: Validation encoder-IDM loss per epoch for the five walled LeWM+I seeds; seed 9999's spike (peak 3.9 at epoch 17) exceeds the axis range.

C ENCODER SCALE: VIT-SMALL

Retraining both models at ViT-Small (22M encoder parameters) with otherwise identical settings and the same 30-epoch budget preserves the probe asymmetry: plain spread/motion 2.0×10^{-4} and rollout-error ratio 28.4; LeWM+I 0.525 and 2.08. The planner result, however, degrades: LeWM+I reaches only 0.033 trial mean (plain 0.008), and object-position decodability worsens from 15 px to 72 px, while the cost-landscape cv stays in the same range (0.11 vs. 0.14 at ViT-Tiny, averaged over tiers), the sharpest instance of the probe's insufficiency. The comparison is at matched epochs, not matched convergence; we claim only that the mechanism is not a ViT-Tiny artefact at the representation level, and that the absolute success ceiling at our budget is ViT-Tiny-specific.

D OGBENCH CUBE: AN EXPECTED NULL

On OGBench visual cube-single (Park et al., 2024) (64×64 frames bilinearly upsampled to 224, 1M frames, 30 epochs, $N{=}50$), plain and LeWM+I are indistinguishable ($4/50$ vs. $3/50$, disjoint episodes, McNemar $p = 1.0$), consistent with the single-object null on PushT at the trained horizon. On cube-double both are at $0/50$; at ViT-Tiny scale on upsampled 64^2 renders neither model reaches the capability floor, so this cell is uninformative rather than negative.

E A SCRIPTED REFERENCE CONTROLLER

To calibrate the tier numbers we run a hand-coded controller that reads the simulator state, under exactly the evaluation protocol of Section 5.1: the same 50 scene seeds per tier, the same scene setup and zero-action warm-up, the same episode budgets (20 or 60 planner steps of five environment steps) and the same success thresholds. On the navigation tier it drives the pusher to the target zone. On the push tiers it orbits the target object at a fixed radius until it is behind the object relative to the target zone, closes in on the contact point, pushes along the object-to-zone direction, and re-plans every environment step from the current positions; it stops once the object is within 12 px of the zone centre. Results (success rate over 50 seeds): easy 1.00 / 1.00, medium 1.00 / 1.00, hard 1.00 / 1.00, very hard 0.28 / 1.00 at 20 / 60 steps. Every push-tier episode moves the target object by more than 5 px. The very-hard tier is therefore budget-limited even for a controller with state access: its long pushes, often past distractors, rarely fit into 20 planner steps. The same controller solves every tier within 20 steps on both response arms of Appendix G. The controller is not an optimum, only a demonstration that the scenes are solvable within the budget by a policy with state access, which the learned world models are not.

F DISPLACEMENT STATISTICS OF THE TRAINING DISTRIBUTIONS

Section 3 characterises the environments under random actions; here we characterise the actual training data. For SLIM we replay the dataset generator (same seeds, same scripted primitive policy, 200-step episodes) for 150 episodes without storing frames and record every body position after every environment step; for PushT we read the block and agent positions from the state channel of 1500 random episodes of the standard training release. Figure 8 shows the displacement over one predictor step (frameskip 5) in each simulator's 512-px frame. On SLIM (29,400 transitions) no object moves in 33% of transitions, the most-moved object shifts by under 5 px in 48% and under 25 px in 99% (mean 7.4 px, median 5.7 px); the pusher moves 22 px in the median. On PushT (179,890 transitions) the T-block is static in 54% of transitions (median 0.2 px) but moves further when it moves (mean 7.8 px, 90th percentile 26 px), and the agent moves 38 px in the median. The fraction of static-object transitions therefore does *not* separate the two benchmarks: a copy-forward prediction of the object is right about as often on PushT as on SLIM. What separates them is how much of the frame a moving body changes (the T-block covers far more pixels than a SLIM object, so its share of $\mathcal{L}_{\text{pred}}$ is much larger) and how readily the random actions a sampling-based planner draws move it (Section 3: 27 vs. 1.1 px root-mean-square displacement over 25 steps). The response-to-actions sweep of Appendix G tests this account directly by raising the response to random actions at a fixed data policy. Pusher and object positions in the training data are uniform over the arena; push directions are slightly anisotropic (largest deviation of an eight-sector histogram 3.7σ over 2 500 episodes) and in single-zone episodes the zone lies in the left half in 82% of cases.

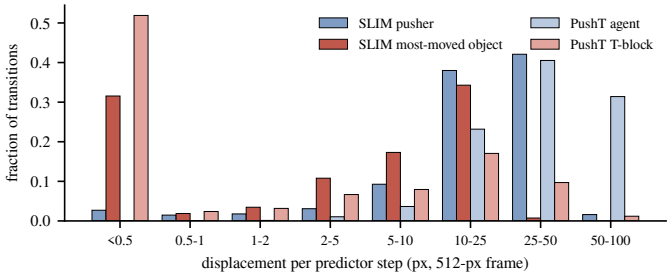


Figure 8: Per-predictor-step displacement of pusher/agent and of the most-moved object in the SLIM and PushT training data.

G RESPONSE-TO-ACTIONS SWEEP

We vary how much of the frame responds to sampled actions (Fig. 9 and Table 5) while holding data volume (10 000 episodes of 200 steps, base seed 42, the same primitive policy as the main dataset), training (canonical seed 3072, 30 epochs, final checkpoint) and evaluation (the canonical protocol of Section 5, tier generator run under the same physics) fixed. The dose is measured model-free under the environment's native action distribution (under the shared σ=0.5 distribution of Section 3 the default scene changes 0.37% and PushT 0.64%): the fraction of the 224×224 frame whose pixels change per predictor step (16 start states $\times$ 20 sequences of 25 steps; zero actions change 0%), and, for reference, the root-mean-square displacement of an object after 25 steps in a single-object scene. Two levers. *Size*: object radius 30 px ($4\times$ footprint) leaves the changing fraction at 0.36% against 0.35% (object motion 1.1 vs. 0.56 px), because larger objects are still rarely hit. Larger radii cannot help further: 45 px raises object motion only to 1.7 px, and from 50 px the physics breaks (objects overlap at initialisation; zero-action motion 2.7 px at 50 px, 6.9 px at 60 px, exactly zero below). *Response*: at the default 15-px footprint, lighter and less damped objects with a larger and faster pusher, calibrated so that zero-action motion stays zero and the scripted controller of Appendix E solves every tier within 20 steps: *lo* (pusher 28 px, mass 0.5, damping 0.95, action scale 80; 1.10% of the frame, object motion 6.6 px) and *hi* (pusher 40 px, mass 1, damping 0.95, action scale 80; 1.65%, 9.9 px). Both the objects and the enlarged pusher contribute to the changing fraction, and we make no claim about which one matters; the claim is about how much of the frame responds. Mass and damping alone do not raise the response much (object motion 3.3 px at mass 0.5, damping

0.99): in 24 to 28 of 32 random starts the pusher never touches an object in 25 steps, so contact geometry, not object inertia, is the bottleneck. The response arms are slightly easier for a policy that does not plan: zero actions still score 0.00, a uniform-random policy 0.045 (hi) and 0.040 (lo) trial mean (medium 0.12 and 0.10, hard 0.06 on both; 0.00 at the default physics), far below the repaired model. At radius 30 the controller solves very hard in 0.30 of scenes within 20 steps (default 0.28). At the lo arm plain is already action-sensitive (0.88) yet plans poorly (0.100), one more latent that clears the gate without being plannable. The LeWM+I cell of the hi arm diverged at the canonical learning rate (validation loss oscillating fourfold from epoch 7) and was retrained at half the rate; all other cells use the canonical setting.

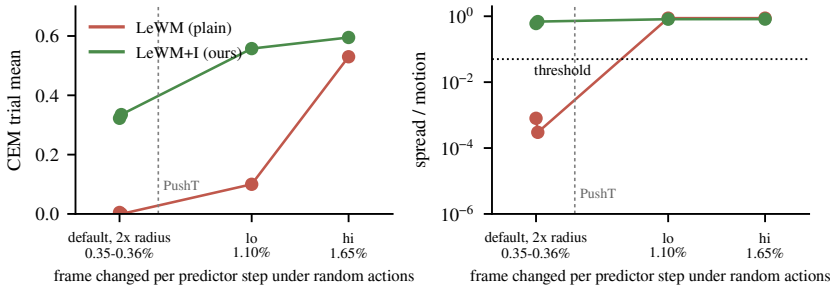


Figure 9: Response-to-actions sweep. Left: CEM trial mean; right: action-sensitivity probe (dotted line: usability threshold). Arms ordered by the fraction of the frame that changes per predictor step under native random actions (0.35, 0.36, 1.10, 1.65%).

arm	radius	pusher	mass / damping	scale	frame changed	model	easy	medium	hard	very hard	trial mean	spread/motion
r15	15	14	8 / 0.85	40	0.35%	plain	0.01	0.01	0.00	0.00	0.005	8×10^{-4}
						LeWM+I	0.96	0.20	0.13	0.00	0.323	0.60
r30	30	14	8 / 0.85	40	0.36%	plain	0.00	0.00	0.00	0.00	0.000	3×10^{-4}
						LeWM+I	0.99	0.22	0.12	0.01	0.335	0.69
lo	15	28	0.5 / 0.95	80	1.10%	plain	0.09	0.19	0.12	0.00	0.100	0.88
						LeWM+I	0.96	0.82	0.44	0.01	0.558	0.81
hi	15	40	1 / 0.95	80	1.65%	plain	0.83	0.80	0.40	0.09	0.530	0.88
						LeWM+I	0.98	0.93	0.40	0.07	0.595	0.82

Table 5: Response-to-actions sweep. One training seed per cell; per-tier values average the two planner horizons; "frame changed" is the model-free fraction of the frame that changes per predictor step under native random actions; the r15 rows are the canonical cell of Table 1 retrained.

H Loss-weight sensitivity and the SIGReg weight

We retrain the canonical cell at $\alpha_{\text{inv}} = \alpha_{\text{pid}} \in \{0.1, 0.3\}$ (seed 3072, 30 epochs, canonical protocol) alongside the existing $\alpha = 0$ and $\alpha = 1$ cells. The repair is graded and monotone in the weight: trial mean $0.000 \to 0.118 \to 0.293 \to 0.368$ for $\alpha = 0/0.1/0.3/1$ (per tier easy/medium/hard: $0.41/0.05/0.01$, $1.00/0.10/0.07$, $1.00/0.30/0.17$; very hard 0 throughout; McNemar α=0.3 vs. 0.1 and 1 vs. 0.3 both $p < 10^{-5}$). Easy saturates at $\alpha = 0.3$; the pushing tiers, including hard where the goal-agnostic baseline scores 0, keep rising until the weight reaches 1 ($0.01 \to 0.07 \to 0.17$). Low weights buy navigation, the full weight buys pushing. At 0.1, the weight used by DA-LeWM (Wang et al., 2026a) in a setting where LeWM already plans, the recipe reaches 32% of the canonical result here.

Action sensitivity (environment-path variant of the probe, which gives 0.54 for the canonical cell against 0.56 in Table 4) does not track this ordering. $\alpha = 0.1$ already lifts spread/motion from 4×10^{-4} to 0.55, and $\alpha = 0.3$ has the highest value of the four cells (0.80) at a lower trial mean (0.293) than $\alpha = 1$ (spread/motion 0.54, trial mean 0.368). Since these cells differ only in one scalar, they are the cleanest evidence that the probe is a gate below which every configuration failed rather than a predictor of planning quality above it (Section 5.4).

The SIGReg weight does not cause the collapse. Retraining plain with $\lambda_{\text{sigreg}} \in \{0.03, 0.27\}$, a third and three times the default 0.09 (seed 3072, canonical protocol), leaves it at the floor: $1/400$,

$0/400$ and $0/400$ successes for $0.03/0.09/0.27$, with spread/motion 0.011, 4×10^{-4} and 0.002, all well below the usability gate. The multi-object collapse is therefore not a mistuned regulariser that the IDM gradient happens to compensate for.

I DOES THE LATENT COST RANK ACTIONS LIKE THE SIMULATOR?

The probes of Section 5.4 are model-internal. This one compares the planner's own cost against the environment. For each of 30 held-out scenes (seeds disjoint from the 50 evaluation seeds) we draw $N = 64$ action sequences of $H_a = 5$ from the CEM iteration-0 proposal, score each by the latent path cost the planner would use, execute the same sequences in the simulator, and rank-correlate the two. On the easy tier, where the real cost spreads by 145 px across the proposal sequences so there is something to rank, LeWM+I reaches Spearman $\rho = 0.84 \pm 0.01$, significantly positive on 30 of 30 scenes, while plain is uncorrelated (-0.02 ± 0.05). On the medium tier the same 25 actions move the object by a median of 2.6 px, so the diagnostic is signal-starved rather than uninformative: LeWM+I is $+0.12$ against plain's -0.02. This is the random-action gap of Section 3 seen from the planner's side.

Run across the disentangle cells (easy tier), the diagnostic reproduces Table 2 with tighter error bars than 400 Bernoulli trials allow: encoder-IDM only $+0.88 \pm 0.01$ and the full model $+0.84 \pm 0.01$ (both 30 of 30 scenes significant), against predictor-IDM only $+0.03 \pm 0.03$ and the shuffled-action null $+0.00 \pm 0.03$ (at most 4 of 30), both at plain's level. One tension is worth stating: encoder-IDM only *beats* the full recipe on this metric while scoring 7.8 pp lower planner success on the same canonical checkpoints, so the two measurements disagree about that one ordering. Predictor-IDM may trade a little cost-surface fidelity for a rollout CEM can exploit over 20 steps, but these data do not establish that.

J CROSS-DISTRIBUTION GENERALISATION

This control separates how little of the frame responds to actions from the presence of distractors. Models trained on a single-object, fixed-appearance SLIM variant (one red circle, radius 15 px) and evaluated on the default multi-object tiers: plain $1/400$; LeWM+I 0.190 trial mean (easy 0.55, medium 0.16, hard 0.05, very hard 0.00, pooled over both horizons), about half of the in-distribution 0.368. Without distractors in training, plain still collapses (spread/motion 6×10^{-4}, action-induced object motion ≈ 0), while LeWM+I recovers spread/motion 0.24, about $400\times$ plain on the same data.

K GROUNDING CONTROLS FOR THE LANGUAGE HEAD

A goal-agnostic policy. Moving in one fixed direction for k planner steps and then stopping, with direction and k chosen per tier in hindsight, never reads the goal. It solves 0.12 of easy and 0.20 of medium scenes and none on hard and very hard (random actions: 0 on every tier); medium is the tier where the four directions of its axis-aligned template leave the most to a blind policy, so we use the hard tier as evidence of goal-directed pushing.

Where the difficulty lies. The same MLP design conditioned on the encoded goal frame instead of the sentence, trained on the same $40\,000$ scenes, matches the oracle (0.363 vs. 0.368; per tier $0.99/0.29/0.17/0$; McNemar $p = 0.80$), so the training pipeline and planner are sound and the difficulty lies in mapping the sentence to a goal latent. A 5-layer MLP language head over the global latent reaches 0.195 ($0.53/0.16/0.09/0$): lower than the cross-attention head on navigation, higher on medium and hard, and less reliable under the zone swap below (success 0.42 and 0.48, two decoy hits, closer to the named zone in 95 of 100 episodes; easy success 0.45 with the swap vs. 0.53 without).

Zone decoy. To test whether the head reads the colour in the sentence rather than a typical spatial relation between pusher and target, we swap the target zone with the other zone on the easy tier, so that a zone of the wrong colour occupies the target's position and the named zone sits where the distractor was. A head that reads the colour still succeeds; one that has learned where targets lie goes

to the decoy. The visual-goal oracle scores 0.94 and 0.88 (H_a=5 and 10) on the swapped scenes. Over three seeds the cross-attention head succeeds on 0.84 of them, exactly as often as without the swap (252 vs. 253 of 300), never ends within the threshold of the decoy, and ends closer to the named zone than to the decoy in all 300 episodes (50 scenes, 3 seeds, 2 horizons); per-seed values are in Appendix L.

L LANGUAGE HEAD: ADDITIONAL RESULTS

Architecture and training. The cross-attention head has two pre-LN transformer layers (4 heads, hidden 192, FFN 768) over a cls-query initialised from z_t, a text-query from a 2-layer refinement of e_{NL}, and the encoder's 256 patch tokens (16×16 at patch 14) with learned positions; the cls output is z_{goal} (1.1M parameters). It is trained for 90 epochs with the mean-squared error to the encoded next-stage latent. Scenes come from the tier generator (seeds from $100\,000$, 2 500 per tier per $10\,000$ scenes); each of the six frames carries one sentence drawn from 41 stage templates. The MLP head of Table 3 has 5 layers (hidden 1024, $\sim$ 5M parameters) over z_t and e_{NL} and is trained for 30 epochs.

Stages and sentences. Training tuples are (latent of the current stage, sentence of the next stage) $\rightarrow$ latent of the next stage: the three random pre-contact poses map to the approach stage, approach to midway, midway to goal, and the goal stage to itself. The sentences change with the stage ("Position the pusher behind the red object, aligned with the green zone", "Push the red object halfway toward the green zone", "The red object is at the green zone"). At test time it receives the final-goal sentence at the start of the episode and again after 10 planner steps, each time from the current frame.

Paraphrase pools. The train pool has 107 templates, a superset of the 41 training templates with 66 rephrasings in the same vocabulary style; the test pool has 31 held-out templates whose object and zone nouns never occur in training, with verbs overlapping partly, mostly on the navigation templates. Per tier on seed 42 (easy/medium/hard): canonical $0.88/0.05/0.05$, train pool $0.84/0.04/0.02$, test pool $0.78/0.01/0.03$; very hard is 0 throughout.

Zone decoy on three seeds. Easy tier, H_a=$5/10$, 50 seeds each: success $0.92/0.86$, $0.84/0.68$, $0.92/0.82$ for head seeds 42, 7, 11 (oracle $0.94/0.88$); decoy hits 0 in all 300 episodes; ended closer to the named zone in 100 of 100 for every seed. Without the swap the same heads score 0.88, 0.78 and 0.87 on easy.