LATENT PROJECTION FOR LATENT PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models let an agent plan in latent space. The planner searches for an action sequence that minimizes a cost, and that cost is often the ℓ_2 distance between the predicted terminal latent and the goal latent. The latent, however, is partly noise: the regularizer that prevents collapse during world model training promotes variance in the directions carrying no signal. We show that the same regularizer also supplies the structure to prune that noise in the latent. SIGReg drives the latent marginal to a standard Gaussian at every step, and assuming further that the latent sequence is a Gauss–Markov process, the transition is linear with its covariance pinned by the map itself, $z' \mid z \sim \mathcal{N}(Az,\, I - AA^\top)$. Fitting the state dynamics A therefore yields the covariance at no additional cost, and in its eigenbasis each latent direction carries its own uncertainty. Furthermore, constraining A to be a normal matrix fixes that basis across timesteps, so one eigenbasis measures the uncertainty over an arbitrary number of steps and serves as a stable basis to compute the planning cost on. We then restrict the ℓ_2 cost to the k-dimensional subspace the state dynamics carry forward with the least uncertainty. The correction is post hoc: the encoder, the predictor, the rollout, and the planner are all left untouched, and only a single constrained matrix, A, is fit. We evaluate across six environments and show large gains on the 2D navigation tasks, TwoRoom and PointMaze, raising success rate over the plain ℓ_2 cost by up to $+54.7$ points under a harder planning setting.

1 INTRODUCTION

Latent world models let an agent plan in representation space. An encoder compresses an observation into a latent representation, a predictor advances that latent under an action, and a planner searches for an action sequence that carries the latent close to the goal. Every part of the loop involves the latent, and learning this representation is a central design choice that varies across latent world models (Maes et al., 2026; Assran et al., 2025; Zhou et al., 2024; Hansen et al., 2024; Hafner et al., 2025).

Joint-Embedding Predictive Architectures (JEPAs) (LeCun et al., 2022), a recipe for general representation learning, are widely chosen for their simplicity: a JEPA predicts the target's representation from the context's representation, and that prediction objective is what shapes the representation. Left unconstrained, the representation collapses to a constant, so JEPAs rely on a regularizer to keep the embedding informative. Recent instances of JEPA use regularizers that encourage the embedding to lie in a structured distribution, most commonly an isotropic Gaussian or a variant of it (Balestriero & LeCun, 2025; Wu et al., 2026; Kuang et al., 2026b).

LeWorldModel (LeWM) (Maes et al., 2026) builds a latent world model on this recipe, training an encoder and predictor end to end from pixels with a two-term loss: next-latent prediction and SIGReg (Balestriero & LeCun, 2025), which pulls the embedding toward an isotropic Gaussian. Similar to many other latent world models (Zhou et al., 2024; Assran et al., 2025; Sobal et al., 2025), LeWM plans by ranking candidate action sequences on the ℓ_2 distance between the predicted terminal latent and the goal latent.

The simplicity of the cost, however, overlooks two sources of noise in the representation: (i) the regularizer promotes variance in which noise fills the directions carrying no signal; and (ii) the encoder, trained to represent observations, retains content irrelevant to planning (Zhao et al., 2026; Wang et al., 2026; Li et al., 2026a). The distance is then measured partly along directions that carry no information about reaching the goal.

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.

Ironically, the same regularizer that injects the noise also gives the structure to repair it. The embedding is driven to be stationary under the transition, and reading the latents as a Gaussian world model in the sense of Klindt et al. (2026), we find the noise is not free but pinned to the transition itself: fit the state dynamics and the covariance comes with it. By projecting onto a subspace determined by the covariance, we obtain a representation restricted to the directions that are certain with respect to the state dynamics. In general, the subspace moves under the transition, but we show that it is fixed under a normality constraint on the state dynamics. All of this is post hoc: the only thing we fit is a constrained linear function; the encoder, the predictor, and the planner are untouched.

Our main contributions are:

1. **Uncertainty measure in a time-invariant basis.** Under SIGReg's standard Gaussian marginal and a Gauss–Markov model, the noise is pinned to the dynamics, $z' \mid z \sim \mathcal{N}(Az,\, I - AA^\top)$, where each eigenvector carries its own uncertainty. Each step of the state dynamics generally moves that basis. We show that when A is a normal matrix, the retained subspace is closed under the dynamics and the same basis serves every timestep.
2. **A new planning cost.** We measure the cost on the subspace spanned by the k eigenvectors of $I - AA^\top$ with the smallest eigenvalues: the directions the state dynamics carry forward with certainty. Empirically, planning with this cost delivers large gains in success rate on the 2D navigation tasks, TwoRoom and PointMaze, of up to $+54.7$ points under a harder planning setting.

2 Preliminaries: LeWorldModel

LeWM (Maes et al., 2026) is a latent world model that learns the representation and the state-action dynamics together. An encoder f maps an observation x to a latent $z = f(x) \in \mathbb{R}^n$, and a predictor g outputs the future latent, $\hat{z}' = g(z, a)$, under an action a.

Training. The learning is driven by two terms, the predictor loss and the SIGReg loss (Balestriero & LeCun, 2025):

$$\min_{f,g}\ \mathbb{E}\big[\, \big\| g(f(x), a) - f(x') \big\|^2 \,\big] + w\, \mathcal{R}_{\mathrm{SIGReg}}. \tag{1}$$

The SIGReg term pulls the latent distribution toward $\mathcal{N}(0, I)$, which prevents representation collapse; it acts on the encoded latents at every step of the trajectory, so z and z' are driven toward the same marginal, providing stationarity in transition.

Planning. Given an initial observation x_0 and a goal observation x_g, a candidate action sequence $a_{1:T}$ is rolled out from $z_0 = f(x_0)$ through the predictor, and its terminal latent $\hat{z}_T$ is scored by the distance to the goal latent $z_g = f(x_g)$:

$$\mathcal{C}(a_{1:T}) = \big\| \hat{z}_T - z_g \big\|_2. \tag{2}$$

A sampling-based planner (Rubinstein & Kroese, 2004) ranks candidates by this cost and iteratively refines its sampling distribution toward better action sequences. The cost, however, falls short as a planning signal: distance in the latent space does not measure progress on the task, the regularizer injects noise into the embedding, and the representation also holds details irrelevant to it. This weakness is well recognized (Wang et al., 2026; Bai & Xiong, 2026; Li et al., 2026b; Thil et al., 2026; Zhao et al., 2026).

In this work, we take the encoder and predictor as given: both stay frozen throughout, and we modify only the cost (2), replacing the plain ℓ_2 distance with one computed after projecting latents onto a subspace that reflects task-relevant dynamics.

3 Transition under a Gaussian World Model

Our objective is to identify the latent subspace that retains clean *state dynamics*. In this section, we model the latent space such that the state dynamics take the form of a normal matrix, and show that its covariance is pinned by the transition itself. This covariance in turn identifies the subspace: the directions the state dynamics carry forward with certainty.

This follows from Gaussian world model assumptions similar to those of Klindt et al. (2026). By construction of LeWM training, the latent marginals are standard Gaussian and stationary: SIGReg

drives the encoded latent toward $\mathcal{N}(0, I)$ at every step of a trajectory. The rest is our modeling choice: we assume the latent sequence is a Gauss–Markov process, and further assume the state dynamics act independently in some orthonormal basis (on lines and planes).

Model (state dynamics under a Gaussian world model).

(i) **Markov.** At every step, $p(z' \mid z, \ldots) = p(z' \mid z)$.

(ii) **Joint Gaussianity.** The pair (z, z') is jointly Gaussian.

(iii) **Stationarity.** Both marginals are standard Gaussian: $z, z' \sim \mathcal{N}(0, I)$.

(iv) **Normality.** The state dynamics act independently on orthogonal lines and planes (see Section 3.2).

Klindt et al. (2026) prove that LeJEPA-style training (Balestriero & LeCun, 2025) recovers the true latent of a Gaussian world model up to a rotation; Zhang et al. (2026) extend this to an action-conditioned predictor as the learner. Every assumption above is invariant under rotation, so we read the model as a statement about the true latent, not about a particular embedding.

3.1 The Transition is Linear with Pinned Noise

By the Markov property (i) and joint Gaussianity (ii), the future latent is expressed as $z' = m(z) + \eta$ at every step, where the state dynamics $m(z)$ are affine and the noise η is Gaussian and independent of z. Stationarity (iii) centers both marginals, so $m(z) = Az$, and pins the noise covariance to keep the marginals at $\mathcal{N}(0, I)$. As in Klindt et al. (2026), we have an Ornstein–Uhlenbeck transition, now determined by a single matrix A:

$$\boxed{z' \mid z \;\sim\; \mathcal{N}\big(Az,\; I - AA^\top\big)} \tag{3}$$

Whatever variance is contracted under the transition is refilled with exactly that much noise, keeping the marginal standard Gaussian. Two consequences follow. First, the noise is pinned to the transition: fit A, and $\Sigma = I - AA^\top$ comes for free. Second, since $\Sigma \succeq 0$, every singular value of A lies in $[0, 1]$.

The covariance further tells us where the uncertainty lies. Since the covariance is symmetric, its eigenvectors form an orthonormal basis: $I - AA^\top = U \operatorname{diag}(\sigma_1^2, \ldots, \sigma_n^2) U^\top$. Rotating the latent into that basis, $\bar{z} = U^\top z$, leaves the transition in the same form,

$$\bar{z}' \mid \bar{z} \;\sim\; \mathcal{N}\big(\bar{A}\bar{z},\; I - \bar{A}\bar{A}^\top\big), \tag{4}$$

where $\bar{A} = U^\top A U$ and the covariance is diagonal, $I - \bar{A}\bar{A}^\top = \operatorname{diag}(\sigma_1^2, \ldots, \sigma_n^2)$. *Along this basis the noise is independent, and each direction u_i carries its own uncertainty σ_i^2.* Retaining those of smallest σ_i^2 isolates a subspace with clean state dynamics.

3.2 Normality of the State Dynamics

The remaining assumption (iv) states that the state dynamics are *normal*: the matrix A satisfies

$$\boxed{AA^\top = A^\top A} \tag{5}$$

In the spirit of the independence assumption in Klindt et al. (2026), we assume the state dynamics evolve in some basis in an independent manner. A real normal matrix decomposes as $A = QBQ^\top$ (Horn & Johnson, 2012), with Q orthogonal and B block diagonal: 1×1 blocks λ, each scaling a line, and 2×2 blocks $rR(\theta)$, each rotating a plane by θ and scaling it by r.[1] In the rotated coordinates, $\bar{z} = Q^\top z$, the transition becomes $\bar{z}' \mid \bar{z} \sim \mathcal{N}\big(B\bar{z},\; I - BB^\top\big)$, where the mean and the noise are both block diagonal. Unlike in Section 3.1, the transition is block-wise independent: each block is a transition of its own, carrying uncertainty $1 - \lambda^2$ on a line and $1 - r^2$ on a plane. Note that we may take $Q = U$.

Beyond the independence itself, normality buys us two things.

[1] $rR(\theta) = r \left(\begin{smallmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{smallmatrix}\right)$, a rotation by θ scaled by $r \geq 0$.

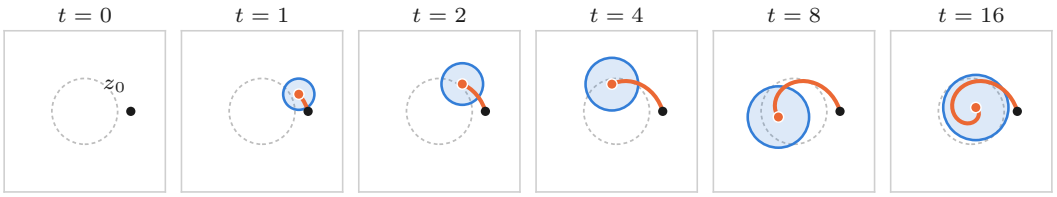


Figure 1: How uncertainty grows across transitions in a 2×2 block — one independent plane, given by normality. Starting from z_0 (black), the state dynamics rotate by θ per step while contracting by r (orange). The blue circle marks the uncertainty the transition leaves, drawn at one standard deviation; it widens toward the unit circle of the stationary law $\mathcal{N}(0, I)$ (grey, dashed).

Shared basis across every horizon. For any A, iterating (3) gives the t-step transition,

$$z^{(t)} \mid z \sim \mathcal{N}\big(A^t z,\ I - A^t (A^t)^\top\big), \tag{6}$$

but for a non-normal A the eigenbasis of that covariance generally changes with t, so the predictable subspace depends on the horizon.[2] Normality provides the equality $A^t(A^t)^\top = (AA^\top)^t$, under which the basis U of (4) is shared across every horizon. A transition fitted at a different frame skip should then share the basis and differ only in spectrum. Along the basis, the uncertainty grows as $1 - \lambda^{2t}$ on a line and $1 - r^{2t}$ on a plane. What motivates a horizon-independent basis is discussed further in Appendix A.

Static and motion content, separated. The blocks distinguish two kinds of predictable content. A 1×1 block predicts *static* content: λ fixes the uncertainty $1 - \lambda^2$ and nothing more, so all it captures is how much of the feature survives the transition. A 2×2 block predicts *motion* content: r fixes the uncertainty $1 - r^2$ the same way, and moreover θ captures how the feature moves in the plane under the transition; at $\theta = 0$ the feature stays put, so the block reduces to predicting static content. Figure 1 illustrates a motion block across transitions.

How much of the latent is motion versus static content is set by a hyperparameter: P planes and $L = n - 2P$ lines, where n is the latent dimension. We use the all-planar layout, $P = \lfloor n/2 \rfloor$, unless otherwise stated.

4 Latent Projection for Latent Planning

We now turn (3) into a planning cost. The method has two stages: fit the state dynamics A to trajectories encoded by a pretrained encoder, then use the covariance it determines to decide which latent directions the planner should compare. LeWM's encoder and predictor stay frozen; planning then runs on them as before, with A shaping the cost.

4.1 Fitting the State Dynamics

We encode the trajectory data once with the frozen encoder f, and fit A to the latent transitions by least squares,

$$\min_A \ \mathbb{E}\big[\, \|Az - z'\|^2 \,\big]. \tag{7}$$

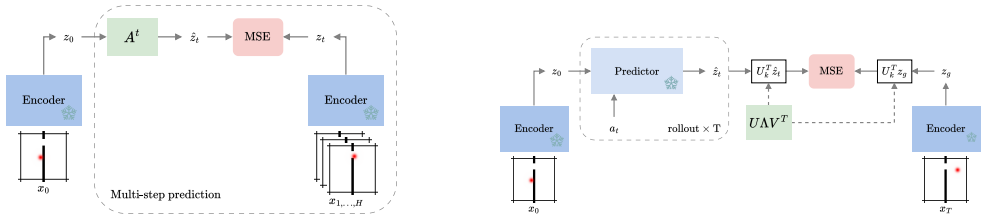
Note that the fit is action-free, so A describes the state dynamics under the policy that generated the data. The fit must respect two constraints: $\Sigma = I - AA^\top$ must be a valid covariance, $\Sigma \succeq 0$, and A must be a normal matrix, as discussed in Section 3. We parameterize A so that both hold by construction; Appendix B gives the full parameterization.

Multi-step fitting. The t-step transition is just A^t (6), and trajectory datasets supply H future frames. We therefore fit multiple timesteps jointly,

$$\min_A \ \sum_{t=1}^{H} \mathbb{E}\big[\, \|A^t z - z^{(t)}\|^2 \,\big], \tag{8}$$

so that a single A has to account for the t-step transition at every $t \le H$ (Figure 2a).

[2] With $A = USV^\top$, $A^2 = US(V^\top U)SV^\top$, and that middle $V^\top U$ leaves $I - A^2(A^2)^\top$ no longer diagonal in U.



(a) Multi-step fitting.

(b) Latent projection for latent planning.

Figure 2: (a) The frozen encoder maps the initial observation and each of the H future observations to latents, and A^t is fit to the t-step transition by squared error. A is the only trained component. (b) A candidate action sequence is rolled out through the frozen predictor, and its terminal latent is compared to the goal after both are projected onto the retained directions U_k.

4.2 Planning in the Projected Latent

Ranking action sequences by latent distance is a statement about reachability, and reachability is governed by the state dynamics. Fitting A recovers those dynamics and their uncertainty measure. Restricting the cost to the certain directions can then sharpen the ranking.

Fitting A sets the covariance as $\Sigma = I - AA^\top$ at no additional cost. Its eigendecomposition

$$\Sigma = U \operatorname{diag}(\sigma_1^2, \ldots, \sigma_n^2)\, U^\top, \qquad \sigma_1 \le \cdots \le \sigma_n, \tag{9}$$

gives the basis that the cost will use: the i-th column u_i of U carries noise variance σ_i^2, and the columns are ordered from most to least certain.

We keep the k most certain directions. Writing $U_k \in \mathbb{R}^{n\times k}$ for the corresponding columns, we replace the plain ℓ_2 cost (2) with the distance computed after projecting both latents onto that subspace,

$$\mathcal{C}_k(a_{1:T}) = \left\| U_k^\top \hat{z}_T - U_k^\top z_g \right\|_2, \tag{10}$$

where $\hat{z}_T$ is obtained by rolling the candidate action sequence forward through the frozen predictor g exactly as before. Only the cost changes; the encoder, the predictor, the rollout, and the planner are all untouched. Taking $k = n$ recovers the plain ℓ_2 cost, since the projection is then a rotation, which leaves distances unchanged.

The subspace does not depend on the horizon. The t-step transition (6) carries noise $I - A^t(A^t)^\top$. By the normality assumption this equals $I - (AA^\top)^t$, which shares the eigenvectors of Σ at every t (Section 3.2). Only the eigenvalues change, growing monotonically with t. One basis therefore serves every horizon. For a non-normal A the basis generally changes with t, and no single choice serves every horizon. This is what motivates building normality into the parameterization of A.

5 Experiments

The experiments answer three questions: whether the cost under the projected latent representation improves planning and how it compares to other baseline methods (Sections 5.2 and 5.3); whether the effect relies on the structure the regularizer imposes on the embedding (Section 5.4); and whether the structure imposed on the state dynamics matters (Section 5.5).

5.1 Environments and Planning Setup

We evaluate on six environments: TwoRoom, PushT, OGBench-Cube, Reacher, PointMaze-UMaze, and PointMaze-Medium. For each we take a pretrained checkpoint and keep the encoder and predictor frozen, with per-task details deferred to Appendix C. The state dynamics A are fit under the parameterization of Appendix B, on the same trajectory data that trained the world model, at frame skip 5 and prediction horizon $H = 4$, for 50 epochs with AdamW at learning rate 10^{-3} and batch size 1024. The latent dimension is $n = 192$ throughout, and the rank of the projection is swept over $k \in \{2, 4, 8, \ldots, 128, 192\}$ on validation seed 0 to pick the best rank $k^\star$. Taking $k = n$ recovers the plain ℓ_2 cost that LeWM plans with. We fit A with 3 seeds $\{0, 1, 2\}$ and report the mean and standard

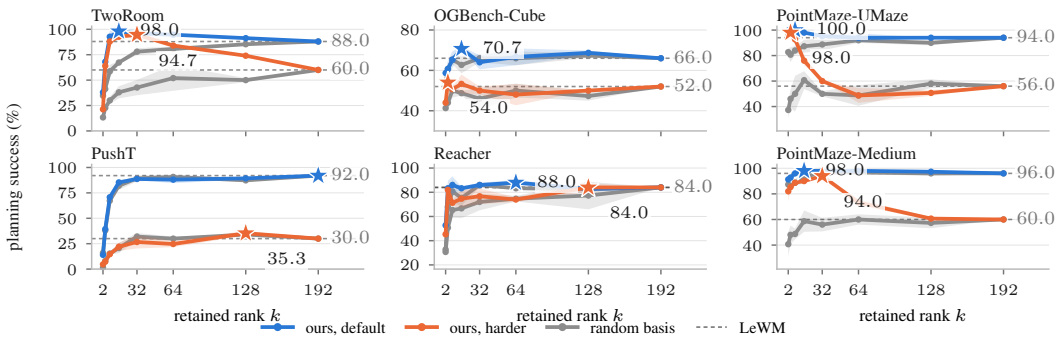


Figure 3: Planning success rate (%) by retained rank k on validation seed, mean $\pm$ std over 3 seeds of 50 episodes. Every task is run in both the default and harder settings. A random basis at the same retained rank is included as a control, drawn in gray. At $k = 192$ every basis recovers LeWM, drawn as the dashed rule; a star marks the selected rank $k^\star$.

deviation of planning success rate. Evaluation is held fixed at test seed 1, so every entry is scored on the same 50 initial states and goals, matching the evaluation protocol of prior work (Maes et al., 2026; Zhou et al., 2024).

We adopt the planning configuration of LeWM throughout: a cross-entropy method (CEM) solver (Rubinstein & Kroese, 2004) with a planning horizon of 5 steps, our *default* setting; the exact configuration is deferred to Appendix D. We also evaluate a *harder* setting with planning horizon 10, where the choice of cost has a larger effect on success rate.

5.2 Planning under Projection

To answer whether the cost under the projected latent representation improves planning, we plan with $\mathcal{C}_k$ (10) at every rank of the sweep. As a control, we project the latent onto a random orthonormal basis of rank k instead, resampled for each seed. Any gap between the two comes from keeping the informative latent directions. At $k = n$ both recover the plain ℓ_2 cost (2) exactly, since projection onto a full orthonormal basis preserves the distance.

Figure 3 reports the sweep on the validation seed. On most tasks, a small number of ranks already matches what the cost computed on the full basis achieves, and projection onto a random basis behaves similarly. This indicates the latent representation carries a great deal of information irrelevant to planning. Appendix E supports this, reporting the eigenvalues of the transition noise, where the state dynamics are typically uncertain along a large part of the basis. More subtly, our cost matches the full-basis success rate at a lower rank than the random control. The same ordering appears in state recovery (Appendix F), where a linear probe recovers the ground-truth state from the retained directions well before a random subspace of the same size does.

We now turn to where the projection helps. It wins clearly on TwoRoom, PointMaze-UMaze and PointMaze-Medium. The gain is most significant under the harder setting. Projection onto a random basis shows no comparable gain at any rank. On the remaining tasks the gain is marginal: PushT needs the full basis, and Reacher and OGBench-Cube show no such gain over the random control. Planning can fail in three places: the information the task requires may be absent from the representation, the predictor may not advance the latent accurately over the horizon, or the cost may not capture closeness to the goal latent. Our method repairs only the cost, cannot compensate for the other two, and rests on the model of Section 3, whose assumptions may not hold under contact-rich dynamics.

5.3 Comparison to Other Planning Costs

Our method projects the latent onto the subspace the state dynamics carry forward with certainty. We compare it against other planning cost baselines that run on the same frozen world model:

- **TD-JEPA (frozen)** (Bai & Xiong, 2026) replaces ℓ_2 with a learned temporal distance d_ψ: a quasimetric head predicts how many steps separate two latents, and the planner ranks a candidate

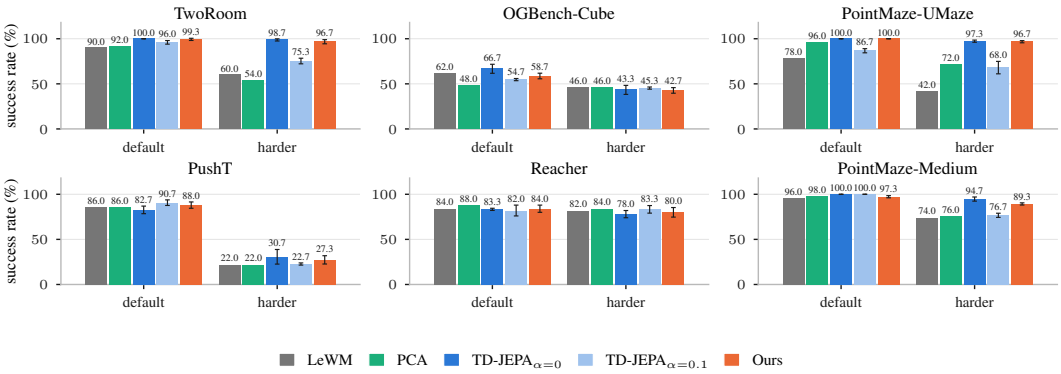


Figure 4: Planning success rate (%) under five planning costs, evaluated on 50 episodes. TD-JEPA blends its distance with the plain ℓ_2 cost as $(1-\alpha)\, d_\psi + \alpha\, \ell_2$, and both $\alpha = 0$ and $\alpha = 0.1$ are reported. PCA takes the covariance of the latents and keeps its top eigenvectors. PCA and our cost are reported at the retained rank $k^\star$ selected on the validation seed, swept over $k \in \{2, 4, 8, \ldots, 128, 192\}$. Both the state dynamics A and d_ψ are fit with 3 seeds; bars are the mean over those seeds, and error bars one standard deviation.

by the predicted temporal distance from its terminal latent to the goal. TD-JEPA blends that distance with ℓ_2 as $(1 - \alpha)\, d_\psi + \alpha\, \ell_2$ and compares $\alpha \in \{0, 0.1, 1\}$, where $\alpha = 1$ is the plain ℓ_2 cost. As published the head trains jointly with the encoder and predictor, shaping the representation itself. Here, to make the baseline comparable, we fit only the head, leaving LeWM's encoder and predictor frozen.

- **PCA** is a standard linear dimensionality reduction (Jolliffe, 2002). We take the covariance of the latents, project onto its top k eigenvectors, and measure the ℓ_2 distance between the predicted terminal latent and the goal latent. The rank is swept over the same grid as our cost, $k \in \{2, 4, 8, \ldots, 128, 192\}$, and selected the same way. Note that SIGReg drives the marginal toward $\mathcal{N}(0, I)$, whose covariance is the identity; at the optimum PCA reduces to projecting onto a random orthonormal basis.

Both run under the planning setup of Section 5.1, on all six tasks and in both the default and harder settings. Figure 4 reports the comparison on the test seed. The observation from the validation seed (Section 5.2) holds, with large gains under the harder setting: $+36.7$ on TwoRoom, $+54.7$ on PointMaze-UMaze and $+15.3$ on PointMaze-Medium over LeWM. Among the baselines, TD-JEPA trains its cost with additional supervision, the number of steps separating two latents, while our cost comes purely from the frozen representation. Despite that, the two are comparable: averaged over tasks we are -0.9 default, -1.7 harder, both against $\alpha = 0$. PCA also measures the cost on a subspace, but chooses it by variance in the embedding rather than variance in the dynamics, and we outperform it by a greater margin, $+3.2$ default and $+13.1$ harder.

5.4 Varying the Latent Regularizer

The derivation in Section 3 starts from the latent marginals being standard Gaussian, which SIGReg supplies in training. To test whether the latent projection still works when that structure is weakened or removed, we train three further world models on TwoRoom with the LeWM recipe, replacing only the SIGReg term in (1):

- **VISReg** (Wu et al., 2026) targets the same isotropic Gaussian, enforced with a sliced-Wasserstein term rather than the Epps–Pulley statistic.
- **RDMReg** (Kuang et al., 2026b;a) matches the latent to a rectified Laplace — the coordinate-wise ReLU of a Laplace — yielding a sparse, non-negative code.
- **Stop-grad** (Chen & He, 2021) uses no explicit regularizer at all: collapse is prevented by detaching the prediction target, and the marginals are unconstrained. With no regularizer weight, we tune the encoder learning rate instead.

Table 1: Comparison across anti-collapse methods on the harder TwoRoom setting. Each variant is a retrained LeWM recipe with only the regularizer swapped, at a strength chosen to bring their success rate under the plain ℓ_2 cost close together. *Gaussian* marks whether the method targets a Gaussian marginal. *Epps–Pulley* measures the Gaussianity of the latent. *Gap closed* is the gap that our cost closes, $(\mathrm{SR}_{\mathrm{ours}} - \mathrm{SR}_{\mathrm{baseline}})/(100 - \mathrm{SR}_{\mathrm{baseline}})$, where SR is the planning success rate, measured under the plain ℓ_2 cost ($\mathrm{SR}_{\mathrm{baseline}}$) and under our cost at the best retained rank $k^\star$ ($\mathrm{SR}_{\mathrm{ours}}$); both appear in parentheses. Latent regularizers that encourage Gaussian structure close the largest share of the gap.

Regularizer	Strength	Gaussian	Epps–Pulley ↓	Gap closed ↑
SIGReg (LeWM)	$w = 0.09$	✓	2.83	92% (60.0 → 96.7)
VISReg	$w = 20$	✓	1.52	98% (40.0 → 98.7)
RDMReg	$w = 20$	×	23.80	87% (58.0 → 94.7)
Stop-grad	lr = 3e−6	×	42.63	0% (52.0 → 52.0)

To compare the gain across variants, we bring planning success under the plain ℓ_2 cost close together by modulating the regularization strength. Each variant then reruns the sweep of Section 5.2 under the harder TwoRoom setting and is reported at the test seed, everything else held fixed. To measure how well the Gaussian assumption actually holds, we evaluate the Epps–Pulley statistic (Epps & Pulley, 1983) of SIGReg (Balestriero & LeCun, 2025) on held-out latents.

Table 1 reports the outcome. The plain-ℓ_2 scores land close but not equal, so we report the gain as a fraction of the headroom each variant has left (*gap closed*). Every variant with an explicit regularizer term benefits from the projection. The recovery weakens as the latent moves away from the Gaussian: the non-Gaussian target of RDMReg gives up part of the gain (87%), and stop-grad, with no regularizer at all, leaves the latent with no structure for the projection to exploit (0%). The gain reflects how well the embedding carries the Gaussian structure assumed in Section 3.

5.5 Ablating the Structural Constraints

The parameterization imposes structural constraints on A, and we ablate each component. (i) *Positive semidefinite* (PSD): the spectrum is kept below 1 in magnitude so that $\Sigma = I - AA^\top$ remains a valid covariance. (ii) *Normal*: $AA^\top = A^\top A$, so one projection basis serves every horizon (Section 3.2). (iii) *Motion*: the blocks are 2×2 rotation–scaling planes (Section 3.2); with 1×1 blocks only, A reduces to a symmetric matrix, in which case $L = n$. (iv) *Multi-step*: A fits $H = 4$ steps jointly (8) rather than the 1-step transition alone.

Each variant reruns the sweep of Section 5.2 on the harder setting of TwoRoom and PointMaze-UMaze, the two tasks we ablate on, and is reported at the test seed, everything else held fixed; Table 2 reports the result. The full parameterization attains the best success rate, and dropping the normality constraint costs the most on both tasks. The structure drives the gain: an unstructured linear fit improves on LeWM by only $+22.0$ on TwoRoom and not at all on PointMaze-UMaze, while the full parameterization of Section 3 gains $+36.7$ and $+54.7$.

Table 2: Ablating the structure of A on the harder setting of TwoRoom and PointMaze-UMaze. Each row keeps (✓) or removes (×) components of the full parameterization (shaded). Success rate is measured at the best retained rank $k^\star$, mean ± std over 3 seeds. The last row is LeWM, which plans with the plain ℓ_2 cost. *PSD*: Σ remains a valid covariance. *Normal*: $AA^\top = A^\top A$. *Motion*: the 2×2 rotating planes; without them A is a symmetric matrix. *Multi-step*: fitting multi-step transitions instead of the single step. Every component contributes.

	PSD	Normal	Motion	Multi-step	TwoRoom	PointMaze-UMaze
Ours	✓	✓	✓	✓	**96.7**±2.3	**96.7**±1.2
	×	✓	✓	✓	96.0±2.0	94.7±1.2
	✓	×	—	✓	88.0±2.0	90.7±3.1
	✓	✓	×	✓	95.3±1.2	96.0±2.0
	✓	✓	✓	×	94.7±5.0	92.0±5.3
	×	×	—	✓	82.0±2.0	42.0±0.0
LeWM	—	—	—	—	60.0	42.0

6 Related Work

Planning in latent space. Model-based planning on a learned latent has a long lineage (Ha & Schmidhuber, 2018; Schrittwieser et al., 2020; Hansen et al., 2022). We study model predictive control (MPC) (Richalet et al., 1978; Richards & How, 2005) in latent space: roll the model forward over candidate action sequences, rank them, execute, replan. PlaNet (Hafner et al., 2019) does this on a recurrent state-space model, ranking rollouts by a learned reward. Nearer to us are models that define the planning cost in representation space: DINO-WM (Zhou et al., 2024) fits a transition on frozen DINO features; V-JEPA 2-AC (Assran et al., 2025) post-trains an action-conditioned predictor on a frozen video encoder; LeWM (Maes et al., 2026) trains one end to end from pixels. They rank candidate sequences by the distance between the predicted terminal latent and the goal latent, ℓ_2 in LeWM and DINO-WM and ℓ_1 in V-JEPA 2-AC, so every latent direction counts the same. In this work, we show that the latent holds content irrelevant to planning, that there exists a clean subspace that carries what matters, and that the distance restricted to that subspace plans better.

Beyond the plain distance. A recent line of work on latent MPC shares one premise: the latent distance alone is a poor proxy for the reachability the planner needs. The approaches split in two. One line trains the representation so that the plain distance is aligned with what planning needs (Zhang et al., 2020; Park et al., 2024; Wang et al., 2026; Thil et al., 2026; Li et al., 2026b). The other redefines the cost, commonly regressing a distance on trajectory step gaps as a surrogate for reachability (Li et al., 2026a; Bai & Xiong, 2026). We keep the latent frozen and train no distance function; instead we fit the state dynamics, a linear function under normality constraints, and compute the cost on the subspace it determines.

Structure in the latent. Self-supervised learning (SSL) yields representations with emergent properties (Chen et al., 2020; Caron et al., 2021; Radford et al., 2021; Zhai et al., 2023; He et al., 2022; Oquab et al., 2023; Ranasinghe et al., 2022). Among these, a JEPA learns by latent prediction (LeCun et al., 2022; Assran et al., 2023; Bardes et al., 2024; Maes et al., 2026). Without an anti-collapse term the representation collapses (Tian et al., 2021; Jing et al., 2021). To remedy this, a stop-gradient with an EMA target was used (Grill et al., 2020; Chen & He, 2021). Other methods enforce structure on the embedding itself through variance and covariance constraints (Bardes et al., 2021; Zbontar et al., 2021; Ermolov et al., 2021) and, more recently, through an isotropic Gaussian target: SIGReg (Balestriero & LeCun, 2025) and subsequent work (Zhao et al., 2026; Wu et al., 2026; Kuhn et al., 2026; Zimmermann et al., 2026). LeWM adopts SIGReg, making its latent stationary under the transition (Maes et al., 2026). But the regularizer can harm the representation: the latent is intrinsically low-dimensional, and an isotropic prior on the latent space is an overly strong bias (Zhao et al., 2026). We instead use the very structure the regularizer imposes to identify the subspace with clean state dynamics.

7 Conclusion

In this work we show that the state dynamics of the latent select the subspace a planning cost should measure on. Under the standard Gaussian marginal SIGReg supplies and a Gauss–Markov model, they are linear and their covariance is pinned by the map itself, $z' \mid z \sim \mathcal{N}(Az,\, I - AA^\top)$. Constraining A to be a normal matrix fixes one eigenbasis across every horizon, and the eigenvalues of $I - AA^\top$ order that basis by the uncertainty the transition leaves along each direction. Measuring the planning cost on the subspace spanned by the k most certain directions delivers large gains on the 2D navigation tasks. The broader point is that the regularizer preventing collapse both injects the noise and supplies the structure to remove it, and that exploiting that structure yields a clean planning signal without supervision.

Limitations and future work. We address the cost, but planning can fail in the representation and in the rollout as well, and restricting the cost to a subspace repairs neither. The derivation also rests on two modeling choices: a Gauss–Markov latent sequence, and a normal matrix A. Future work will explore training a world model that satisfies these constraints. Reachability is also a quasimetric, an asymmetric distance function, while our cost is an ℓ_2 distance on a subspace and so symmetric. Our method does fit the state dynamics, however, and they can be used to compute the state occupancy measure (Sutton et al., 1998), a surrogate for reachability. Learning the state dynamics can therefore be a building block for approximating reachability, which we see as an interesting research direction.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we describe the full experimental setup in the paper: the fitting hyperparameters, seeds and rank grid in Section 5.1, the parameterization of A in Appendix B, the environments, datasets and checkpoints in Appendix C, and the planner configuration in Appendix D. Every reported number is a mean over three fitting seeds with its standard deviation, scored on the same 50 evaluation episodes at a held-out test seed. Code will be released upon publication.

AI USE STATEMENT

In this work, we used generative AI tools to provide critical ingredients for proving mathematical claims, implement methods, and clean and reformat datasets. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, or interpret results, and supporting qualitative and thematic data analysis is not applicable to this work. Additionally, we used generative AI tools to brainstorm, to identify, source and summarize existing literature, to create and modify scientific figures, to create and edit software code, to draft parts of this paper, and to edit it for readability. We have reviewed all AI-assisted work. The theoretical framework is established in prior work that we cite, and we checked our formulation against those sources. Research ideas were checked for potential plagiarism through a manual literature survey, every cited work was retrieved and read by the authors, and all AI-drafted text was reviewed and edited by the authors. AI-generated code was verified and tested for correctness by the authors, and the code will be released so that the implementation is inspectable. The numbers in every generated figure are checked against the stored result files they are drawn from. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

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 *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, 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.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

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.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9630–9640. IEEE, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PmLR, 2020.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 15745–15753. IEEE, 2021.

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. *The International Journal of Robotics Research*, 2025.

Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2024. URL `http://github.com/Farama-Foundation/Gymnasium-Robotics`.

Thomas W Epps and Lawrence B Pulley. A test for normality based on the empirical characteristic function. *Biometrika*, 70(3):723–726, 1983.

Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *International conference on machine learning*, pp. 3015–3024. PMLR, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 15979–15988. IEEE, 2022.

Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint arXiv:2110.09348*, 2021.

Ian T Jolliffe. *Principal component analysis*. Springer, 2002.

David Klindt, Yann LeCun, and Randall Balestriero. When does lejepa learn a world model? *arXiv preprint arXiv:2605.26379*, 2026.

Yilun Kuang, Yash Dagade, Quentin Le Lidec, Lucas Maes, Randall Balestriero, and Yann LeCun. LpWM: A case for sparse representations in world models. *arXiv preprint arXiv:2608.22764*, 2026a.

Yilun Kuang, Yash Dagade, Tim G. J. Rudner, Randall Balestriero, and Yann LeCun. Rectified Lp-JEPA: Joint-embedding predictive architectures with sparse and maximum-entropy representations. *arXiv preprint arXiv:2602.01456*, 2026b.

Lukas Kuhn, Lucas Maes, Giuseppe Serra, Quentin Le Lidec, Yann LeCun, Randall Balestriero, and Florian Buettner. LeVJEPA: Efficient & scalable video pretraining without the heuristics. *arXiv preprint arXiv:2608.27395*, 2026.

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Mario Lezcano-Casado and David Martınez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *International conference on machine learning*, pp. 3794–3803. PMLR, 2019.

Liangyu Li, Shengzhi Wang, Libin Qiu, Mingliang Xiong, and Qingwen Liu. World model control by trajectory reachability metrics. *arXiv preprint arXiv:2605.22164*, 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.

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

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. *arXiv preprint arXiv:2402.15567*, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Kanchana Ranasinghe, Muzammal Naseer, Salman Khan, Fahad Shahbaz Khan, and Michael S Ryoo. Self-supervised video transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2864–2874. IEEE, 2022.

Jacques Richalet, André Rault, Jean-Louis Testud, and Jean Papon. Model predictive heuristic control. *Automatica (journal of IFAC)*, 14(5):413–428, 1978.

Arthur Richards and Jonathan How. Robust model predictive control with imperfect information. In *Proceedings of the 2005, American Control Conference, 2005.*, pp. 268–273. IEEE, 2005.

Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Stress-testing offline reward-free reinforcement learning: A case for planning with latent dynamics models. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025.

Richard S Sutton, Andrew G Barto, and Andrew Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Lucas Thil, Jesse Read, Rim Kaddah, and Guillaume Doquet. Subspace-decomposed jepas: Disentangling progression and content in latent world models. *arXiv preprint arXiv:2605.31111*, 2026.

Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, pp. 10268–10278. PMLR, 2021.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim GJ Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning. *arXiv preprint arXiv:2603.12231*, 2026.

Haiyu Wu, Randall Balestriero, and Morgan Levine. Visreg: Variance-invariance-sketching regularization for jepa training. *arXiv preprint arXiv:2606.02572*, 2026.

Omar G. Younis, Rodrigo Perez-Vicente, John U. Balis, Will Dudley, Alex Davey, and Jordan K Terry. Minari, September 2024. URL `https://doi.org/10.5281/zenodo.13767625`.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pp. 12310–12320. PMLR, 2021.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11941–11952. IEEE, 2023.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

Xiangteng Zhang, Yang Guan, Bo Zhang, Ya-Qin Zhang, and Shengbo Eben Li. On the identifiability of controlled world models. *arXiv preprint arXiv:2607.22430*, 2026.

Kai Zhao, Dongliang Nie, Yuchen Lin, Zhehan Luo, Yixiao Gu, Deng-Ping Fan, and Dan Zeng. Sub-jepa: Subspace gaussian regularization for stable end-to-end world models. *arXiv preprint arXiv:2605.09241*, 2026.

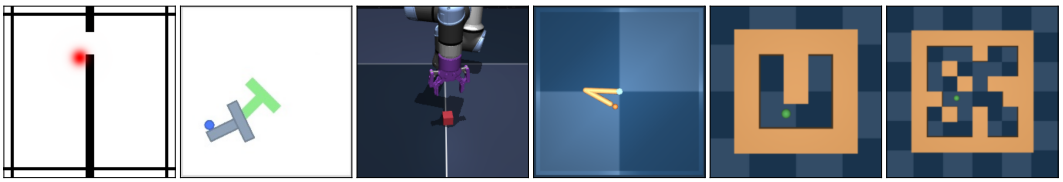
Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

Eric Zimmermann, Harley Wiltzer, Justin Szeto, David Alvarez-Melis, and Lester Mackey. KerJEPA: Kernel discrepancies for Euclidean self-supervised learning. *arXiv preprint arXiv:2512.19605*, 2026.

A Why Does Normality Matter?

The cost of Section 4.2 keeps the k most certain directions of the one-step covariance $\Sigma = I - AA^\top$ and measures the distance there. The subspace is chosen once, before planning begins, and applied to every candidate the planner scores. The choice hides a dependence on the horizon. Over t steps the transition carries noise $I - A^t(A^t)^\top$ (6), and for a general A the subspace moves with t. No candidate settles the choice of t — its remaining gap to the goal is neither given nor fixed. Normality removes the dependence. Every horizon shares one basis and one ordering (Section 3.2), so the subspace from the one-step transition serves them all.

From distance to reachability. The cost scores the predicted terminal latent against the goal, as a distance. A reachability cost — the state occupancy measure — accumulates likelihood over horizons instead. Under normality every horizon is read in one basis, so the sum over t factorizes and amortizes, which could be helpful in approximating reachability. We leave that to future work.



TwoRoom PushT OGBench-Cube Reacher PointMaze-UMaze PointMaze-Medium

Figure 5: The six evaluation tasks.

B Parameterization of the State Dynamics

The fit of Section 4.1 must respect two constraints: A must be a normal matrix and $\Sigma = I - AA^\top$ must be a valid covariance, $\Sigma \succeq 0$ (Section 3). We parameterize A so that both hold by construction during the fit.

We take the form from the decomposition $A = QBQ^\top$ of Section 3.2. Splitting Q and B between the $L = n - 2P$ lines and the P planes,

$$Q = [Q_L \quad Q_P], \qquad B = \begin{bmatrix} B_L & 0 \\ 0 & B_P \end{bmatrix}, \tag{11}$$

we parameterize each type of block separately,

$$\begin{aligned} B_L &= \operatorname{diag}(\lambda_\ell)_{\ell=1}^{L}, \qquad \lambda_\ell = \frac{v_\ell}{\sqrt{1+v_\ell^2}} \in (-1,1), \\ B_P &= \operatorname{blkdiag}\big(r_p R(\theta_p)\big)_{p=1}^{P}, \qquad r_p = \frac{|u_p|}{\sqrt{1+u_p^2}} \in [0,1), \\ A &= Q_L B_L Q_L^\top + Q_P B_P Q_P^\top, \end{aligned} \tag{12}$$

where Q is the matrix exponential of a free skew-symmetric matrix (Lezcano-Casado & Martınez-Rubio, 2019), orthogonal by construction. The free parameters are that skew-symmetric matrix, the L scalars v_ℓ, and the P scalars u_p and angles θ_p.

Normality holds by construction. Each 2×2 block is a scaled rotation, and $R(\theta)$ is orthogonal, so $(rR)(rR)^\top = (rR)^\top(rR) = r^2 I_2$. Each 1×1 block is trivially normal. B is block diagonal with these blocks, so $BB^\top = B^\top B$, and

$$AA^\top = QBB^\top Q^\top = QB^\top BQ^\top = A^\top A. \tag{13}$$

The covariance is positive definite. The same computation gives $BB^\top = \operatorname{diag}\big(\lambda_1^2, \ldots, \lambda_L^2, r_1^2, r_1^2, \ldots, r_P^2, r_P^2\big)$, so

$$\Sigma = I - AA^\top = Q\big(I - BB^\top\big)Q^\top \tag{14}$$

has eigenvalues $1 - \lambda_\ell^2$ and $1 - r_p^2$, the latter appearing twice. The squashing in (12) keeps $r_p < 1$ and $|\lambda_\ell| < 1$, so every eigenvalue is strictly positive and $\Sigma \succ 0$.

C Environments and Datasets

The evaluation covers six tasks. Four of them — TwoRoom, PushT, OGBench-Cube and Reacher — come from LeWM (Maes et al., 2026): the same environments, LeWM's trajectory data, and the released checkpoints, so the encoder and predictor are frozen throughout and are not retrained here. For PointMaze-UMaze and PointMaze-Medium, we collect the training data, following DINO-WM's setting (Zhou et al., 2024), and train the world model with the LeWM recipe. Every world model is trained for 10 epochs. Dataset sizes are given in Table 3. Planning succeeds when the state reached comes within a tolerance of the goal; what that is measured on differs by task.

TwoRoom. A simple continuous navigation task in 2D (Sobal et al., 2025). The environment has two rooms joined by a single opening, and wall (black) blocks the agent everywhere else. The agent (red) navigates to a randomly selected location in the other room, so it must pass through the opening. The trajectories come from a noisy heuristic that steers to the opening first and to the target once through it.

PushT. A contact-rich 2D manipulation task introduced by Chi et al. (2025). The agent (blue) pushes a T-shaped block (gray) onto a T-shaped goal (green). The agent and block positions are scored together, along with the block's angle up to its rotational symmetry. The data is the PushT set released with DINO-WM (Zhou et al., 2024), which LeWM adopts.

OGBench-Cube. A 3D manipulation task from the OGBench suite (Park et al., 2025). A robot arm moves a cube to the goal position. We use the single-cube variant. An episode succeeds when the cube ends near the goal, wherever the arm is. The trajectories come from the scripted policy the benchmark provides for data collection.

Reacher. A continuous control task from the DeepMind Control Suite (Tassa et al., 2018). A two-joint arm reaches for a target. Every joint angle is scored, so the whole arm has to match and not just the fingertip. The trajectories come from an expert SAC policy (Haarnoja et al., 2018).

PointMaze (UMaze and Medium). A 2D navigation task with physics (Fu et al., 2020), in two of its maze layouts, run through the Gymnasium-Robotics implementation (de Lazcano et al., 2024). A point mass moves through a walled maze, and the action is a force along x and y. Only the position of the mass is scored. We collect the training data ourselves, rolling out random action sequences as DINO-WM does (Zhou et al., 2024).

The evaluation start–goal pairs do not come from that random data. On random rollouts the mass often gets stuck at a corner of the maze, so the two states of a pair sit close together; a success rate measured that way comes out inflated. We take the pairs from the D4RL expert trajectories instead, distributed through Minari (Younis et al., 2024), where the point mass does cross the maze.

Table 3: Training data for each world model. Lengths marked "avg." are averages over variable-length episodes.

Task	Episodes	Episode length
TwoRoom	10,000	92 (avg.)
PushT	18,685	125 (avg.)
OGBench-Cube	10,000	200
Reacher	10,000	200
PointMaze-UMaze	10,000	91 (avg.)
PointMaze-Medium	10,000	98 (avg.)

D Planning Configuration

We adopt the planning configuration of LeWM. At each planning step, the CEM solver draws N candidate action sequences of horizon T, scores each by the planning cost, and retains the best E to update the sampling distribution; this repeats for M iterations, after which the whole optimized sequence is executed before replanning. One predictor step covers c environment timesteps (the frame skip). At evaluation the goal is drawn Δ timesteps ahead of the initial state within the same trajectory, and the agent is allowed a budget of b actions in the environment. Table 4 gives the values for the default and harder settings of Section 5.1.

E The Spectrum of the Transition Noise

Figure 6 plots the eigenvalues of $\Sigma = I - AA^\top$ for the A fitted to each task, sorted ascending. The eigenvalue σ_i^2 is the variance the one-step transition leaves along direction u_i: at $\sigma_i^2 = 0$ the next value is determined, and at $\sigma_i^2 = 1$ the transition law along u_i is the marginal itself, so the dynamics carry nothing forward there.

Table 4: Planning configuration for the default and harder settings.

Symbol	Quantity	Default	Harder
N	candidate sequences per iteration	300	300
M	CEM iterations	10 (30 on PushT)	10 (30 on PushT)
E	retained elites	30	30
T	planning horizon (predictor steps)	5	10
c	frame skip (env. timesteps per predictor step)	5	5
Δ	goal offset (timesteps)	25	50
b	evaluation budget (actions)	50	100

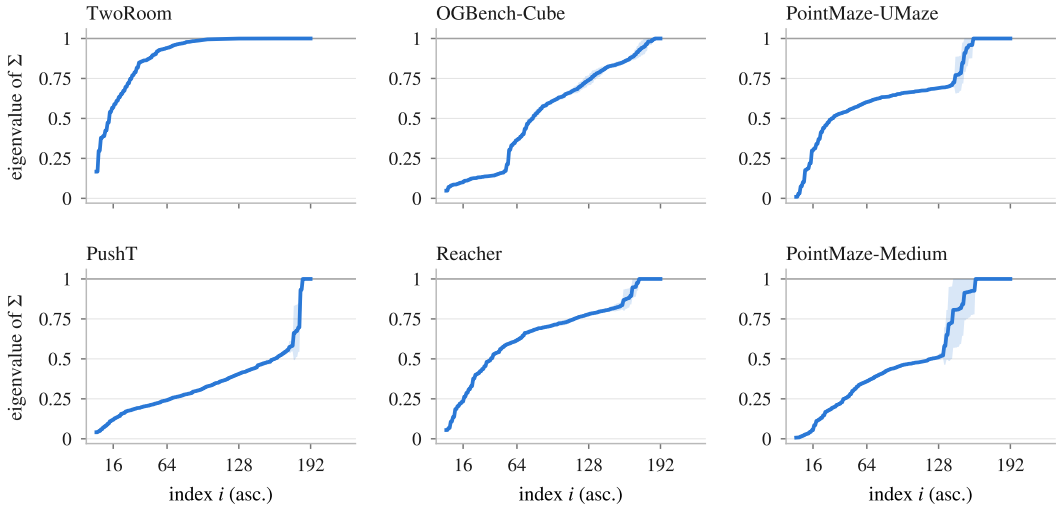


Figure 6: Eigenvalues of $\Sigma = I - AA^\top$ per task, sorted in increasing order. Mean $\pm$ std over the 3 seeds A is fit with.

Every spectrum reaches that ceiling. On TwoRoom, PointMaze-UMaze, Reacher and OGBench-Cube the transition variance is nearer the ceiling than zero over most of the basis, so the state dynamics carry little forward there. PushT stands apart: its spectrum stays low across the greater part of the range.

F State Recovery in the Retained Subspace

We assume the latent holds the information planning needs. The ground-truth state of the environment can serve as a proxy for that information, so a linear probe from the retained directions to the state tests whether they carry it. We pair cached latents $z \in \mathbb{R}^{192}$ with the ground-truth state s. We keep the k most certain directions U_k of the fitted eigenbasis — the same basis the cost projects onto — and fit ordinary least squares $\hat{s} = W(U_k^\top z)$ on 10k frames, then report held-out R^2 on 10k frames from disjoint episodes. As a control, we run the same probe on a random orthonormal subspace of rank k.

Table 5 reports the probe. As the rank grows the retained directions recover the state first, while a random subspace of the same rank lags well behind. What carries the state is the ordering, then, and not the number of dimensions. How quickly the state arrives also tracks the rank planning needs (Section 5.2). PushT and OGBench-Cube are the two tasks slowest to recover, in different ways. PushT climbs gradually, every coordinate improving with k and none of them arriving early. OGBench-Cube recovers unevenly instead: the planar coordinates arrive at the very lowest ranks, while the two heights and the gripper stay absent until well into the sweep.

Table 5: Held-out R^2 of a linear probe reading the ground-truth state from a k-dimensional subspace of the frozen 192-dimensional latent, fit by ordinary least squares on 10k frames and scored on 10k held-out frames from disjoint episodes. *Ours* keeps the k least-noisy eigenvectors of $\Sigma = I - AA^\top$, the directions the cost (10) projects onto; *random* is a random orthonormal k-subspace. Rank increases left to right within each group. Each *mean* averages the rows of its task. Only positions are shown; velocities are omitted.

		ours				random				
Task	State	2	8	32	128	2	8	32	128	192
TwoRoom	x	0.71	0.94	0.97	0.99	0.03	0.09	0.35	0.99	1.00
	y	0.23	0.72	0.90	0.98	0.01	0.06	0.29	0.94	0.99
	mean	0.47	0.83	0.94	0.98	0.02	0.07	0.32	0.96	0.99
PushT	agent x	0.02	0.20	0.47	0.93	0.03	0.09	0.32	0.93	0.96
	agent y	0.03	0.19	0.41	0.92	0.02	0.07	0.33	0.93	0.96
	block x	0.02	0.36	0.86	0.97	0.03	0.09	0.41	0.97	0.98
	block y	0.06	0.41	0.71	0.96	0.02	0.07	0.33	0.95	0.97
	$\sin\theta_b$	0.01	0.17	0.56	0.84	0.01	0.06	0.30	0.84	0.91
	$\cos\theta_b$	0.05	0.23	0.59	0.89	0.01	0.07	0.33	0.88	0.93
	mean	0.03	0.26	0.60	0.92	0.02	0.07	0.34	0.92	0.95
OGBench-Cube	block x	0.92	0.94	0.96	0.99	0.01	0.06	0.31	0.98	0.99
	block y	0.95	0.96	0.97	0.99	0.01	0.07	0.23	0.98	0.99
	block z	0.00	0.00	0.00	0.95	0.02	0.06	0.24	0.97	0.99
	effector x	0.81	0.83	0.84	0.96	0.01	0.06	0.29	0.97	0.99
	effector y	0.90	0.91	0.92	0.96	0.01	0.07	0.23	0.95	0.97
	effector z	0.00	0.00	0.00	0.95	0.02	0.06	0.24	0.95	0.98
	gripper	0.00	0.00	0.00	0.91	0.02	0.08	0.27	0.90	0.92
	mean	0.51	0.52	0.53	0.96	0.01	0.07	0.26	0.96	0.98
Reacher	$\sin\theta_0$	0.46	0.99	0.99	1.00	0.07	0.10	0.48	1.00	1.00
	$\cos\theta_0$	0.67	0.99	0.99	1.00	0.04	0.13	0.48	1.00	1.00
	$\sin\theta_1$	0.83	0.95	0.99	1.00	0.03	0.12	0.47	1.00	1.00
	$\cos\theta_1$	0.00	0.99	0.99	1.00	0.03	0.11	0.43	1.00	1.00
	mean	0.49	0.98	0.99	1.00	0.04	0.11	0.46	1.00	1.00
PointMaze-UMaze	x	0.96	0.99	1.00	1.00	0.05	0.18	0.69	1.00	1.00
	y	0.78	0.99	1.00	1.00	0.04	0.22	0.70	1.00	1.00
	mean	0.87	0.99	1.00	1.00	0.04	0.20	0.69	1.00	1.00
PointMaze-Medium	x	0.25	0.91	0.99	1.00	0.03	0.18	0.47	0.99	1.00
	y	0.31	0.88	0.98	1.00	0.02	0.10	0.49	1.00	1.00
	mean	0.28	0.90	0.98	1.00	0.03	0.14	0.48	0.99	1.00