LS-JEPA: LEARNING LONG SHORT-TERM DYNAMICS FOR FAST LATENT PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Joint-Embedding Predictive Architecture (JEPA) offers a promising route to model-based control by predicting future latent dynamics. Current approaches mainly adopt one-step prediction objectives, which fail to capture the long-term latent transitions for action recovery. In this work, we introduce LS-JEPA, an end-to-end latent world model that jointly learns forward and inverse dynamics across Long Short-term temporal scales. Specifically, LS-JEPA combines one-step prediction with fixed-horizon autoregressive rollout, while an endpoint-conditioned action decoder reconstructs both individual actions and multi-step action sequences from latent displacements. These complementary objectives encourage predictive, action-relevant representations and directly connect latent transitions to control. Furthermore, the learned one-step and multi-step inverse mappings can be naturally used for fast latent planning during the inference time. We propose a Test-Time Replanning (TTR) strategy in which a goal-directed action sequence is decoded and then a one-step correction from a fresh observation is applied to refine the action. Extensive experiments are conducted to validate our proposed approach. The results show that LS-JEPA achieves the state-of-the-art results with 96.00%, 100.00%, 99.67%, and 100.00% on Push-T, OGBench-Cube, Reacher, and Two-Room, respectively, yielding a macro-average of 98.92% with only 11 ms of planning time per decision. These results suggest that jointly learning Long Short-Term forward and inverse dynamics with our proposed TTR strategy enjoys both the accuracy and efficiency for latent planning. Code is available at https://anonymous.4open.science/r/LS-JEPA-E034/.

1 INTRODUCTION

World models provide a framework for planning and control by learning how an environment evolves under actions. Early approaches coupled latent dynamics with observation reconstruction (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2020), whereas more recent methods increasingly model dynamics directly in representation space, thereby avoiding the need to reproduce task-irrelevant visual detail (Hansen et al., 2024; Zhou et al., 2025). World models of Joint-Embedding Predictive Architecture (JEPA) (LeCun, 2022) style extend this principle by learning predictive representations without reconstructing observations: an encoder maps observations to latent states, and an action-conditioned predictor models their evolution by predicting future representations rather than future pixels. This shifts the objective from reproducing appearance towards capturing the aspects of the environment that are most relevant to dynamics and control. Within this paradigm, DINO-WM (Zhou et al., 2025) models dynamics over pretrained visual features, LeWorldModel (LeWM) (Maes et al., 2026) enables stable end-to-end learning of representations and latent dynamics, and Delta-JEPA (Zhang et al., 2026b) introduces inverse-dynamics supervision to retain action-relevant information in latent transitions.

Yet three limitations remain. First, learning control-relevant representations from dynamics alone is challenging to optimize. Since the learned latent space is ultimately used for planning, it is desirable to shape the representation through action-conditioned transitions rather than through generic statistical regularizers that are not explicitly tied to controllable dynamics. However, local dynamics supervision alone leaves substantial freedom in how the latent space is organized. Second, planning requires repeated prediction beyond local transitions, allowing small errors to accumulate and distort imagined futures. Third, forward prediction need not expose the inverse mapping from tran-

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.

sitions to actions. Action selection is then left to test-time optimization, where planners such as the cross-entropy method (CEM) repeatedly sample and evaluate candidate sequences.

To tackle these problems, we introduce LS-JEPA, an end-to-end latent world model designed to learn forward and inverse dynamics across Long and Short-Term temporal scales. LS-JEPA constrains forward dynamics through both one-step prediction and fixed-horizon autoregressive rollout, requiring the same predictor to remain consistent under recursive application. In parallel, a long short-term action decoder (LSAD) recovers both one-step and multi-step action sequences from latent endpoint transitions. Forward and inverse supervision therefore act on complementary aspects of the same latent dynamics: the former promotes temporally consistent prediction, whereas the latter encourages transitions to retain action information associated with state change. The resulting representation is intended not only to predict future latent states, but also to make the actions underlying those transitions directly recoverable.

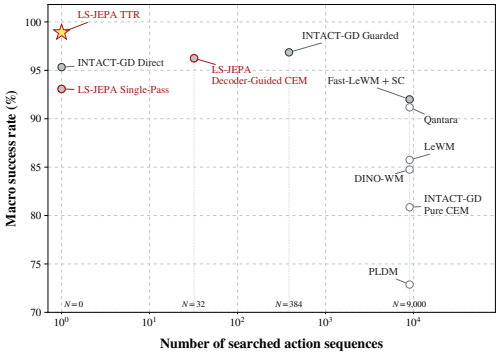


Figure 1: Comparison of performance and the number of searched sequences between LS-JEPA and other methods. Macro-average success is plotted against worst-task success across the four benchmark environments. Numbers in brackets denote the deployment search budget, measured as the number of sampled action sequences. The upper-right region therefore corresponds to methods with high average and worst-task success on the evaluated tasks. LS-JEPA variants expose the effect of different deployment strategies, with TTR attaining the strongest overall trade-off while using zero candidate-sequence sampling.

This joint formulation changes the role of the world model at test time. Rather than relying on increasingly expensive search to compensate for imperfect latent dynamics, LS-JEPA shifts more of the computational burden into representation learning itself. Across Two-Room, Push-T, Reacher and OGBench-Cube, LS-JEPA exhibits markedly smaller raw model-space rollout drift and slower one-step-normalized relative error accumulation than LeWM, while supporting accurate control with markedly reduced search. Combined with lightweight Test-Time Replanning (TTR), it achieves a 98.92% macro-average success rate with zero candidate-sequence sampling and only 11 ms of planning time per decision in the no-rendering setting. Under the matched 25-step goal-offset protocol, these results demonstrate effective control without candidate-sequence sampling as shown in Fig. 1. The rollout diagnostics show smaller native latent drift and slower one-step-normalized relative error growth, not representation-independent gains in physical prediction accuracy.

Our contributions are threefold: (1) We propose LS-JEPA, which jointly learns long- and short-term forward prediction and inverse action recovery for better latent representation learning and planning. (2) We propose a strategy of TTR, combining goal-directed sequence decoding with one observation-based correction, which improves the efficiency of inference by avoiding candidate-sequence sampling. (3) We achieve state-of-the-art performences on four environments with accuracy of 96.00%, 100.00%, 99.67%, and 100.00% on Push-T, OGBench-Cube, Reacher, and Two-Room, and a 98.92% macro success at 11 ms per decision with TTR.

2 Related Work

2.1 World Models

World Models (Ha & Schmidhuber, 2018), PlaNet (Hafner et al., 2019), and Dreamer (Hafner et al., 2020) learn latent dynamics with an observation model; DreamerV3 (Hafner et al., 2023) and DayDreamer (Wu et al., 2022) extend this approach across control tasks. TD-MPC (Hansen et al., 2022) and TD-MPC2 (Hansen et al., 2024) remove observation reconstruction but retain reward and value supervision. Among reconstruction-free approaches, DINO-WM (Zhou et al., 2025) predicts frozen visual features, PLDM (Sobal et al., 2025) studies reward-free offline planning, and V-JEPA 2-AC (Assran et al., 2025) adapts video representations for action-conditioned prediction. LeWM (Maes et al., 2026) jointly learns representations and dynamics, while Sub-JEPA (Zhao et al.,

2026) adds subspace regularization. LS-JEPA couples this latent modeling approach with forward and inverse supervision at two temporal scales.

2.2 Action-Sensitive and Multi-Timescale Latent Dynamics

Learning to predict future representations does not ensure that those representations distinguish the effects of different actions. CLOUD (Wang et al., 2021) couples forward and inverse dynamics to learn representations that retain information relevant to interaction. In latent world models, Delta-JEPA (Zhang et al., 2026b) and Sensorimotor World Models (SMWM) (Ivashkov et al., 2026) use inverse-dynamics supervision to retain action information in latent transitions. World-Action Model (WAM) (Han & Yilmaz, 2026) also uses inverse dynamics to regularize representations. World Models as Group Actions (Wang et al., 2026b) takes a different approach, constraining how actions correspond to transformations in latent space. A related question is how to retain action information in both short-term and long-term predictions. PSG-JEPA (Yan et al., 2026) grounds latent states in proprioception and uses physical-state changes over different time intervals to supervise latent transitions. ActSWM (Gan et al., 2026) likewise encourages action sensitivity over multi-step roll-outs. These works are closely related to our interest in action-sensitive representations and long-term prediction. LS-JEPA jointly learns long and short-term forward and inverse dynamics. Forward prediction and inverse action recovery are supervised in both cases, with the aim of retaining action information during recursive rollout.

2.3 Search-Efficient Control with Latent World Models

Planning with a latent world model often requires evaluating many candidate action sequences at test time. A common choice is the Cross-Entropy Method (CEM) (Rubinstein, 1999), with iCEM (Pinneri et al., 2021) improving sampling efficiency. Other approaches use model structure to guide the search, including hierarchical latent planning (Zhang et al., 2026a) and inverse-dynamics-based trajectory evaluation (Seo et al., 2026). Several recent methods seek to reduce the need for online search. Latent Geometry Beyond Search (Nguyen et al., 2026) amortizes planning through goal-conditioned inverse dynamics. TD-JEPA (Bai & Xiong, 2026) learns a latent geometry informed by temporal distance, while INTACT (Sun et al., 2026) maps latent goal displacement directly to actions. In LS-JEPA, long and short-term inverse dynamics are learned alongside the corresponding forward dynamics, allowing actions to be decoded directly from latent transitions. We use this action decoding to reduce reliance on sampling-based planning at test time.

3 Method

3.1 Overview

Given offline trajectories of RGB observations o_t and continuous actions a_t, we propose LS-JEPA, which learns both long and short-term forward and inverse dynamics as shown in Fig. 2. Our goal is to learn latent transitions that support both prediction and action recovery. A shared encoder maps observations to latent states $z_t = E_\theta(o_t)$. An action-conditioned predictor P_ϕ predicts subsequent latent states, while a long short-term action decoder (LSAD) D_ψ recovers actions from the displacement between two encoded observations. We train both mappings over one step and a longer horizon, sharing their parameters across the two scales. At inference time, LSAD decodes a sequence from the current-to-goal latent displacement. Its one-step mapping can update the final action from a fresh observation.

3.2 Training Forward and Inverse Dynamics

Given latent states z_t extracted from Encoder E_θ, we then train forward and inverse dynamics, i.e., forward latent prediction and inverse action recovery at Long and Short-Term temporal scales, including a single transition and a horizon of H steps.

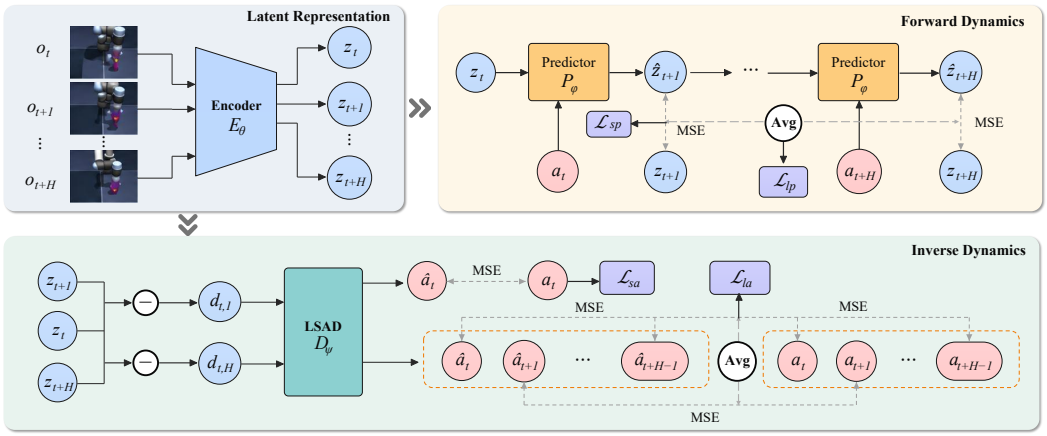


Figure 2: Overview of LS-JEPA. Latent representation (top left): A shared encoder maps observations to latent states. Forward dynamics (top right): An action-conditioned predictor estimates the next latent state and recursively rolls out H steps using recorded actions. The single-step loss $\mathcal{L}_{\text{sp}}$ supervises the first transition, while $\mathcal{L}_{\text{lp}}$ averages prediction errors across the rollout. Inverse dynamics (bottom): LSAD takes the endpoint displacement $d_{t,h} = z_{t+h} - z_t$ and the horizon $h \in \{1, H\}$ to recover a single action or an H-step action sequence without intermediate states. The losses $\mathcal{L}_{\text{sa}}$ and $\mathcal{L}_{\text{la}}$ supervise these outputs against the recorded actions. MSE denotes mean squared error; Avg denotes temporal averaging.

Forward dynamics. We train the action-conditioned predictor using both single-step and multi-step supervision. Starting from the encoded observation $\hat{z}_t = z_t$, the predictor recursively generates future latent states:

$$\hat{z}_{t+h} = P_\phi(\hat{z}_{t+h-1}, a_{t+h-1}), \qquad h = 1, \ldots, H. \tag{1}$$

The first prediction uses the encoded state z_t; subsequent predictions use the preceding model output. Recorded actions are provided throughout the rollout.

For the first transition, we minimize the mean squared error between the predicted state $\hat{z}_{t+1}$ and the encoded observation z_{t+1}:

$$\mathcal{L}_{\text{sp}} = \frac{1}{D} \left\| \hat{z}_{t+1} - z_{t+1} \right\|_2^2, \tag{2}$$

where D is the latent dimension. This objective directly supervises local transition accuracy.

To also supervise predictions made from the model's own outputs, we average the latent prediction error over the full rollout:

$$\mathcal{L}_{\text{lp}} = \frac{1}{HD} \sum_{h=1}^{H} \left\| \hat{z}_{t+h} - z_{t+h} \right\|_2^2. \tag{3}$$

The two objectives therefore emphasize local accuracy and accuracy across recursive predictions, respectively. Although $\mathcal{L}_{\text{lp}}$ includes the first step, the separate $\mathcal{L}_{\text{sp}}$ term provides additional control over its weight. Cross-encoder comparisons of these representation-dependent errors are discussed in Appendix A.14.

Inverse dynamics. We complement forward prediction with action recovery, encouraging latent transitions to preserve information about the actions connecting their endpoints. Given two encoded observations separated by h steps, LSAD predicts the intervening actions from their displacement:

$$d_{t,h} = z_{t+h} - z_t, \qquad \hat{\mathbf{a}}^{(h)}_{t:t+h-1} = D_\psi(d_{t,h}, h), \quad h \in \{1, H\}. \tag{4}$$

The decoder uses shared action queries to attend to the transition representation, with a horizon embedding distinguishing the two scales. It receives no intermediate states, consistent with the current–goal input available at deployment.

At the single-step scale, LSAD predicts the action connecting consecutive states, supervised by

$$\mathcal{L}_{\text{sa}} = \|\hat{a}_t^{(1)} - a_t\|_2^2. \tag{5}$$

At the long-term scale, the same decoder predicts H actions from the endpoint displacement $d_{t,H}$. We compare each prediction with the recorded action at the corresponding position:

$$\mathcal{L}_{\text{la}} = \frac{1}{H} \sum_{k=0}^{H-1} \|\hat{a}_{t+k}^{(H)} - a_{t+k}\|_2^2. \tag{6}$$

Together, these losses supervise action recovery from both local and long-horizon displacements. Since different action sequences can connect the same endpoints, the recorded sequence provides a training target rather than a unique inverse solution.

We jointly train the encoder, predictor and LSAD with the weighted objective as follows:

$$\mathcal{L} = \underbrace{\lambda_{\text{sp}}\mathcal{L}_{\text{sp}} + \lambda_{\text{lp}}\mathcal{L}_{\text{lp}}}_{\substack{\text{Long Short-Term} \\ \text{Forward Dynamics}}} + \underbrace{\lambda_{\text{sa}}\mathcal{L}_{\text{sa}} + \lambda_{\text{la}}\mathcal{L}_{\text{la}}}_{\substack{\text{Long Short-Term} \\ \text{Inverse Dynamics}}}. \tag{7}$$

Training begins with the two local objectives, followed by the long-term losses. When long-term action recovery is first introduced, we detach its input displacement from the encoder, then enable that gradient path in the next stage. The local objectives remain active throughout this schedule. Appendix A.1 gives the architecture, loss weights and complete training procedure.

3.3 Inference Strategies

At inference time, we consider two search-based strategies, Pure CEM (Maes et al., 2026) and Decoder-Guided CEM, and three strategies without candidate sampling, Single-Pass, One-Step Refinement, and Test-Time Replanning (TTR). We describe the latter three below and provide the search-based alternatives in Appendix A.2. Given an initial observation o_1 and a goal observation o_g, we encode them as $z_1 = E_\theta(o_1)$ and $z_g = E_\theta(o_g)$. The planning horizon comprises H action blocks, indexed from 1 to H.

Single-Pass. We use the goal representation as the target endpoint and decode an initial action sequence from the current-to-goal displacement:

$$\hat{\mathbf{a}}_{1:H} = D_\psi(z_g - z_1, H). \tag{8}$$

Single-Pass executes this sequence open-loop. It requires one encoding pass for each endpoint and one decoding pass, without candidate sampling or forward-model rollout. The predictor is used during training but is not called during Single-Pass inference.

One-Step Refinement. This strategy uses observation feedback to refine each action block in the initial sequence. For $h = 1, \ldots, H-1$, we encode the current observation as $z_h = E_\theta(o_h)$ and predict the next latent state using the corresponding initial action $\hat{a}_h$. LSAD then maps the predicted displacement back to an action:

$$\hat{z}_{h+1} = P_\phi(z_h, \hat{a}_h), \qquad a_h^{\text{exec}} = D_\psi(\hat{z}_{h+1} - z_h, 1). \tag{9}$$

After executing a_h^{exec}, we obtain a fresh observation for the next block. For the final block, we decode directly towards the goal, $a_H^{\text{exec}} = D_\psi(z_g - z_H, 1)$. Thus, the initial sequence supplies reference actions, while the predictor supplies local latent targets. The procedure is summarized in Algorithm 2.

Test-Time Replanning (TTR). TTR executes the first $H-1$ blocks of the initial sequence unchanged and uses observation feedback only for the final block. After observing o_H, we replace the final planned action with

$$a_H^{\text{exec}} = D_\psi(z_g - E_\theta(o_H), 1)\,. \tag{10}$$

This correction preserves the H-block execution horizon and adds one encoding and decoding pass relative to Single-Pass. It requires no forward-model rollout, candidate sampling, or parameter updates in Appendix A.3.

Table 1: Comparison with other methods on Push-T, Cube, Reacher and Two-Room environments (Maes et al., 2026). Prior results are taken from INTACT and Delta-JEPA. LS-JEPA follows the matched task protocol. We report mean ± sample standard deviation over three training seeds $\{0, 1, 42\}$, with 100 episodes per seed. Seq. denotes the number of sampled candidate sequences, while Macro is the mean success rate across the four tasks.

Method / objective	Deployment	Seq.	Push-T	Cube	Reacher	Two-Room	Macro
DINO-WM (Zhou et al., 2025)	CEM (published)	9k	74.0±4.5	86.0±4.7	79.0±5.1	100.0±.0	84.75
LeWM (Maes et al., 2026)	CEM 300×(30/10)	9k/3k	96.0±4.0	74.0±3.0	86.0±5.0	87.0±2.5	85.75
Fast-LeWM (Gao & Xu, 2026)	matched CEM	9k	96.0	80.0	88.0	98.0	90.50
Fast-LeWM + SC (Gao & Xu, 2026)	CEM + score	9k	98.0	82.0	90.0	98.0	92.00
Qantara (Rakhimov et al., 2026)	CEM 300×30, $K = 4$	9k	90.1±1.1	93.7±.7	80.9±1.8	100.0±.0	91.18
PLDM (Sobal et al., 2025)	Pure CEM 300×30	9k	76.13±1.70	57.27±1.53	64.33±2.14	93.73±1.03	72.87
Sub-JEPA (Zhao et al., 2026)	Pure CEM 300×30	9k	63.73±.12	62.67±1.45	81.00±2.40	90.60±.53	74.50
Delta-JEPA (Zhang et al., 2026b)	Pure CEM 300×30	9k	89.07±1.90	79.27±1.81	81.33±.50	100.00±.00	87.42
GC-IDM (Nguyen et al., 2026)	Direct	0	84.7±5.0	99.3±1.2	100.0±.0	100.0±.0	96.00
PRISM (Wang et al., 2026a)	MPPI 128×30	3,840	89±4	79±6	–	–	–
C-JEPA (Nam et al., 2026)	CEM 300×30	9k	88.67	–	–	–	–
GC-IDM (Nguyen et al., 2026)	Direct	0	9.33±2.31	69.33±7.02	84.33±2.08	99.33±.58	65.58
PRISM (Wang et al., 2026a)	PoG-MPPI 128×30	3,840	18.67±5.51	86.67±1.15	66.33±1.15	88.33±1.53	65.00
INTACT, goal displacement (Sun et al., 2026)	Direct	0	85.78±1.54	100.00±.00	97.67±.00	97.89±1.26	95.33±.58
	Pure CEM 300×30	9k	88.44±1.17	68.44±.77	83.67±.67	82.89±.84	80.86±.51
	Actor-on CEM 300×30	9k	93.56±.96	96.89±.19	86.67±.88	98.00±1.15	93.78±.77
	Guarded A 128×3	384	92.22±.69	99.78±.19	97.44±.77	98.00±1.15	96.86±.38
LS-JEPA (Ours)	Single-Pass	0	92.33±3.21	100.00±.00	80.00±5.29	100.00±.00	93.08±1.88
	Pure CEM 300×30	9k	94.00±2.00	80.67±3.06	86.00±6.00	100.00±.00	90.17±.58
	One-Step Refinement	0	69.67±3.21	99.67±.58	92.33±2.52	99.67±.58	90.34±1.04
	Decoder-Guided CEM 16×2	32	94.33±2.31	100.00±.00	90.67±3.06	100.00±.00	96.25±.50
	TTR	**0**	**96.00±3.00**	**100.00±.00**	**99.67±.58**	**100.00±.00**	**98.92±.76**

4 Experiments

In this section, we first compare our proposed LS-JEPA with existing methods on four visual control tasks: Push-T, Cube, Reacher, and Two-Room. We then conduct experiments to validate the effectiveness of long-term training objectives and different inference strategies. We finally analyze the robustness of LeWM and our method from different perspectives in details.

4.1 Experimental Setup

We evaluate offline RGB-to-action control on Push-T, OGBench-Cube, Reacher, and Two-Room using the LeWorldModel benchmark. Each goal is an observation sampled 25 simulator steps after the initial state, corresponding to $H = 5$ with five simulator steps per action block. We use the benchmark success criterion for each task. Models are trained separately from scratch for 10 epochs, without rewards, goal labels, or online interaction. For LS-JEPA, we use three random seeds $\{0, 1, 42\}$ and evaluate 100 episodes per seed. Encoder, optimizer, compute, and objective-schedule details are given in Appendix A.1.

4.2 Comparison with State-of-the-Art Results

We compare our method with other methods as shown in Table 1. It can be seen that LS-JEPA with TTR achieves the state-of-the-art results on four individual environments. It achieves 96.00% on Push-T, 100.00% on Cube, 99.67% on Reacher, and 100.00% on Two-Room. Averaged over three training seeds, its macro success is $98.92 \pm 0.76\%$, exceeding the strongest listed INTACT variant ($96.86 \pm 0.38\%$) by 2.06 percentage points. TTR uses no sampled candidate action sequences, whereas that INTACT variant uses 384. Within LS-JEPA, TTR improves macro success over single-pass decoding (93.08%) and decoder-guided CEM (96.25%) by 5.84 and 2.67 percentage points, respectively. The matched-start example in Fig. 3 illustrates the corresponding behavioral difference: LS-JEPA with TTR reaches the goal, whereas LeWM with CEM diverges from the matched starting state. This example complements the aggregate success rates; further trajectories are provided in Appendix A.6.

4.3 Inference Strategies and Planning Efficiency

We compare inference strategies using the same trained LS-JEPA checkpoints to separate the effects of action decoding, search, and feedback on control success and planning time.

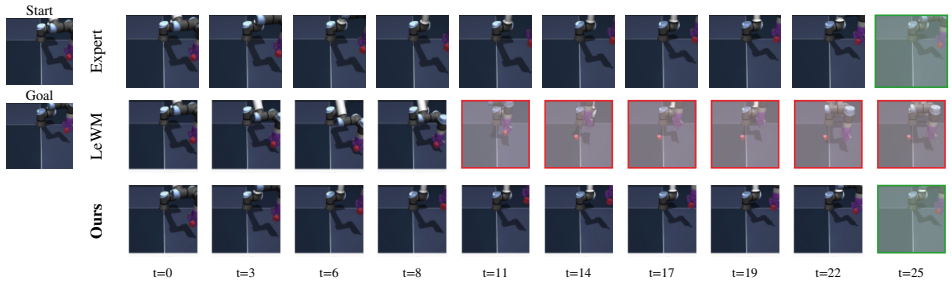


Figure 3: Visualization of OGBench-Cube trajectories of LeWM and our LS-JEPA. Green frames mark the successful endpoints and red frames mark the failure cases.

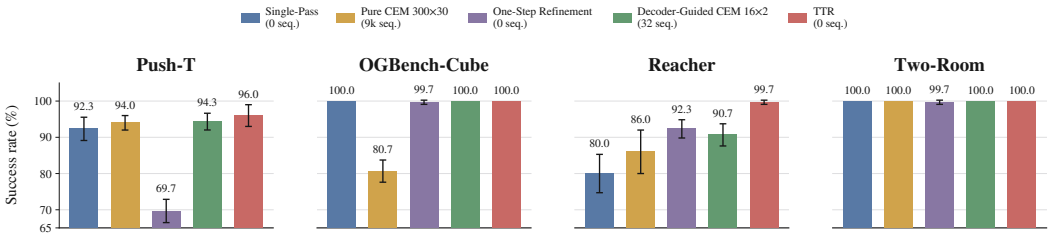


Figure 4: Comparison of different inference strategies on Push-T, OGBench-Cube, Reacher, and Two-Room environments.

Control success. As shown in Fig. 4, Single-Pass reaches 93.08% macro success without sampling, exceeding Pure CEM's 90.17% with 9k sequences. Decoder-Guided CEM reaches 96.25% with 32 samples; TTR reaches 98.92% without sampling. Single-Pass already saturates on Cube and Two-Room. TTR improves Reacher from 80.00% to 99.67% and Push-T from 92.33% to 96.00%, where open-loop control remains imperfect.

Planning cost. Table 2 shows that LS-JEPA supports efficient planning. Single-Pass takes only 6 ms per decision, compared with 22.5 ms for INTACT Direct and 148.5 ms for INTACT Guarded A. TTR takes 11 ms while improving macro success from 93.08% to 98.92%, and remains cheaper than Decoder-Guided CEM. Thus, feedback improves success with modest additional planning cost and no extensive candidate sampling.

Table 2: Latent planning latency (ms) of different inference methods.

Method	Time
INTACT	
Guarded A (128 × 3)	148.5
Direct	22.5
LS-JEPA (Ours)	
Decoder-Guided CEM (16 × 2)	25
TTR	11
Single-Pass	**6**

4.4 Ablation Studies

Effectiveness of long-term objectives. Table 3 shows the effectiveness of long-term forward and inverse objectives. Under Pure CEM, the short-term-only baseline reaches 83.25% macro success. Adding long-term forward supervision raises this to 86.83%, and adding long-term inverse supervision gives 86.92%. Using both reaches 90.17%, a gain of 6.92 percentage points under the same inference strategy. The benefit depends on how the model is deployed. Adding the long-term forward objective to inverse supervision lowers Single-Pass macro success from 95.25% to 93.08%, but raises TTR success from 96.50% to 98.92%. Additional representation diagnostics are provided in Appendix A.13.

Effectiveness of test-time replanning. As reported in Table 1, TTR raises success from 80.00% to 99.67% with the same Reacher model. Fig. 5 illustrates how replanning at step 20 improves alignment and reduces final joint errors without candidate sampling.

Table 3: Effectiveness of long-term forward and inverse objectives. Checkmarks and crosses indicate whether each long-term objective is used during training. The short-term objectives $\mathcal{L}_{sp}$ and $\mathcal{L}_{sa}$ are active in every configuration and are omitted from the table. The first row is the baseline. Other rows report success rates (%) and differences from this baseline in percentage points. Without $\mathcal{L}_{la}$, the decoder conditioned on the sequence endpoints is not trained, so these models are evaluated only with Pure CEM.

Long-term objectives		Deployment	PushT	Cube	Reacher	TwoRoom	Macro
$\mathcal{L}_{lp}$	$\mathcal{L}_{la}$						
×	×	Pure CEM 300×30	76.00	77.33	79.67	100.00	83.25
✓	×	Pure CEM 300×30	87.00 (+11.00)	77.67 (+0.34)	82.67 (+3.00)	100.00 (0.00)	86.83 (+3.58)
×	✓	Single-Pass	89.67 (+13.67)	100.00 (+22.67)	92.33 (+12.66)	99.00 (-1.00)	95.25 (+12.00)
		Pure CEM 300×30	83.00 (+7.00)	76.70 (-0.63)	88.00 (+8.33)	100.00 (0.00)	86.92 (+3.67)
		One-Step Refinement	66.00 (-10.00)	100.00 (+22.67)	93.33 (+13.66)	99.67 (-0.33)	89.75 (+6.50)
		Decoder-Guided CEM 16×2	92.00 (+16.00)	100.00 (+22.67)	93.33 (+13.66)	99.00 (-1.00)	96.08 (+12.83)
		TTR	92.67 (+16.67)	100.00 (+22.67)	94.33 (+14.66)	99.00 (-1.00)	96.50 (+13.25)
✓	✓	Single-Pass	92.33 (+16.33)	100.00 (+22.67)	80.00 (+0.33)	100.00 (0.00)	93.08 (+9.83)
		Pure CEM 300×30	94.00 (+18.00)	80.67 (+3.34)	86.00 (+6.33)	100.00 (0.00)	90.17 (+6.92)
		One-Step Refinement	69.67 (-6.33)	99.67 (+22.34)	92.33 (+12.66)	99.67 (-0.33)	90.34 (+7.09)
		Decoder-Guided CEM 16×2	94.33 (+18.33)	100.00 (+22.67)	90.67 (+11.00)	100.00 (0.00)	96.25 (+13.00)
		TTR	**96.00 (+20.00)**	**100.00 (+22.67)**	**99.67 (+20.00)**	**100.00 (0.00)**	**98.92 (+15.67)**

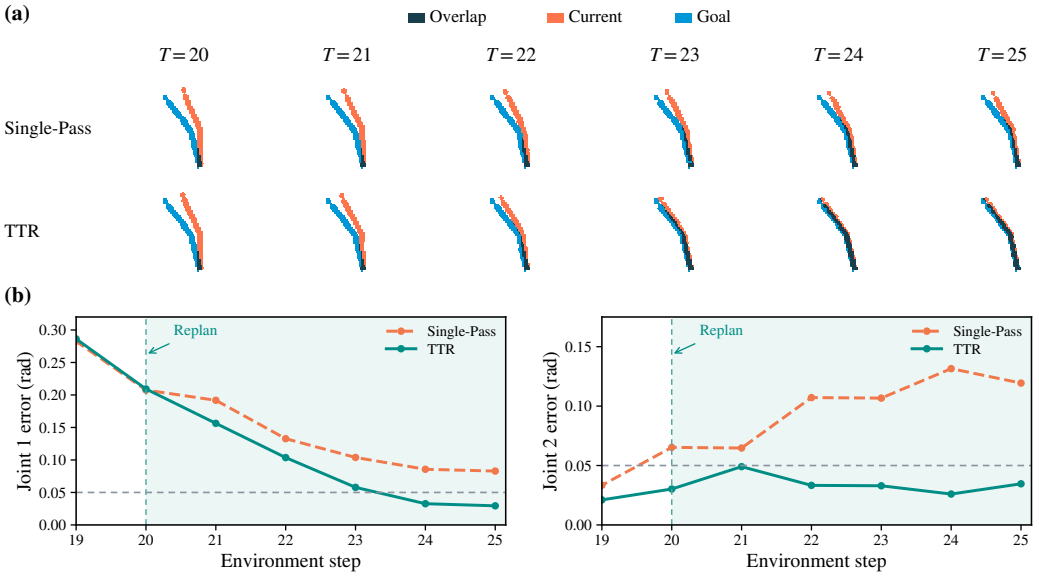


Figure 5: Effectiveness of TTR on the Reacher environment. (a) Goal alignment of Single-Pass and TTR from environment steps 20 to 25. (b) Joint errors during execution. The vertical dashed line marks replanning at step 20, and the horizontal dashed lines indicate the 0.05 rad threshold.

Native latent drift. We compare LS-JEPA and LeWM in open-loop rollouts from matched contexts using identical recorded actions. Fig. 6 shows lower native latent error and slower relative error growth for LS-JEPA. Its one-step-normalized error is lower by 21.14%, 42.59%, and 54.08% at 10, 25, and 50 steps, respectively. This comparison is representation-dependent, however; variance-normalized MSE favors LeWM on Reacher, as detailed in Appendix A.14.

Robustness to temporal intervals. We compare LS-JEPA and LeWM under the same Pure CEM 300×30 planner at 10, 25, and 50 step goal intervals, with execution budgets twice the interval as shown in Table 4. The 25-step setting matches training, while 10 and 50 steps test shorter and longer intervals. At 50 steps, LS-JEPA achieves 77.5% macro success versus 62.0% for LeWM and performs better on all four tasks. From 25 to 50 steps, macro success drops by 13.0 points for LS-

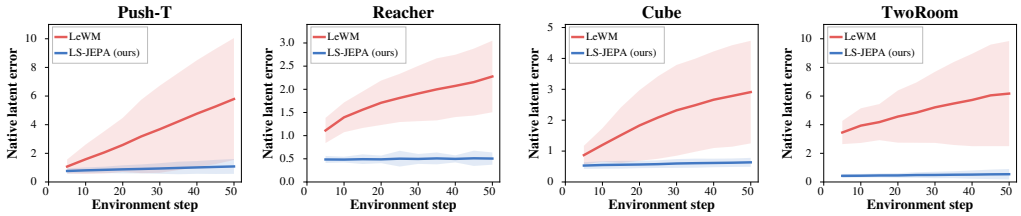


Figure 6: Latent prediction errors of LS-JEPA and LeWM. Both models roll out from matched contexts using identical recorded actions, without observation updates. Native errors compare predicted and encoded future states in each model's latent space. One-step-normalized errors divide these errors by the corresponding one-step error.

Table 4: Goal-reaching success rates (%) of LS-JEPA and LeWM at temporal intervals of 10, 25, and 50 steps.

Interval	Method	Push-T	Cube	Reacher	Two-Room	Macro
10 steps	LeWM	98	92	84	98	93
	LS-JEPA	98 (0.00)	98 (+6.00)	80 (-4.00)	100 (+2.00)	94 (+1.00)
25 steps	LeWM	88	72	84	80	81
	LS-JEPA	92 (+4.00)	82 (+10.00)	88 (+4.00)	100 (+20.00)	90.5 (+9.50)
50 steps	LeWM	36	58	90	64	62
	LS-JEPA	54 (+18.00)	62 (+4.00)	94 (+4.00)	100 (+36.00)	77.5 (+15.50)

JEPA versus 19.0 for LeWM, showing better retention at the longest tested interval. These control results use a single seed.

Prediction horizon. On Push-T, we vary the prediction length H and evaluate all models with the same Pure CEM strategy. As shown in Fig. 7, success increases from $69.33 \pm 6.35\%$ at $H = 1$ to $94.00 \pm 2.00\%$ at $H = 5$. It then falls to $87.00 \pm 4.00\%$ at $H = 7$ and $77.67 \pm 2.52\%$ at $H = 10$. The best result is at $H = 5$; extending long-term supervision further does not improve control under this protocol.

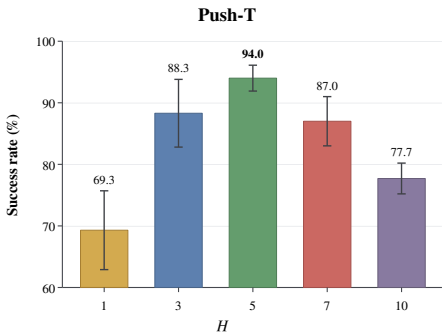


Figure 7: Effectiveness of different prediction length H on Push-T. All models are evaluated using the same Pure CEM planning strategy.

5 Conclusion

In this paper, we proposed LS-JEPA, which jointly learns forward and inverse dynamics at Long and Short-term temporal scales. We further proposed a TTR strategy for fast latent planning during inference time. The proposed approach achieved the state-of-the-art results of 96.00%, 100.00%, 99.67%, and 100.00% on Push-T, OGBench-Cube, Reacher, and Two-Room and a macro success of 98.92% with 11 ms per decision without candidate-sequence sampling.

Limitation. LS-JEPA relies on forward and inverse dynamics objectives to organize the latent space, without an explicit distributional regularizer or direct supervision of task-relevant state variables. Consequently, z_t is not explicitly constrained to preserve information such as the robot end-effector or gripper pose, and may underrepresent such factors when they are not required by the training objectives. Incorporating latent regularization or auxiliary state-aware objectives could improve representation coverage while preserving action sensitivity.

REFERENCES

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. 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.

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:2607.26712*, 2026.

Yuntian Gao and Xiangyu Xu. Fast LeWorldModel. *arXiv preprint arXiv:2606.26217*, 2026.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *ICML*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020.

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

Yuci Han and Alper Yilmaz. Enhancing policy learning with world-action model. *arXiv preprint arXiv:2603.28955*, 2026.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *ICML*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *ICLR*, 2024.

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

Yann LeCun. A path towards autonomous machine intelligence, 2022. URL https://openreview.net/forum?id=BZ5a1r-kVsf. Position paper.

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

Heejeong Nam, Quentin Le Lidec, Lucas Maes, Yann LeCun, and Randall Balestriero. Causal-JEPA: Learning world models through object-level latent interventions. *arXiv preprint arXiv:2602.11389*, 2026.

Hoang Nguyen, Xiaohao Xu, and Xiaonan Huang. Latent geometry beyond search: Amortizing planning in world models. *arXiv:2605.08732*, 2026.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *CoRL*, 2021.

Ruslan Rakhimov, George Bredis, Yuriy Maksyuta, and Daniil Gavrilov. Qantara: Bridge-flow training for multi-paradigm JEPA control. *arXiv preprint arXiv:2607.04978*, 2026.

Reuven Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

Gawon Seo, Dongwon Kim, and Suha Kwak. ACID: Action consistency via inverse dynamics for planning with world models. *arXiv:2607.02403*, 2026.

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. In *NeurIPS*, 2025.

Junhan Sun, Hao Zhao, and Guofeng Zhang. INTACT: Isomorphic intent-to-action learning for search-free world models. *arXiv preprint arXiv:2607.26056*, 2026.

Jianren Wang, Yujie Lu, and Hang Zhao. CLOUD: Contrastive learning of unsupervised dynamics. In *Conference on Robot Learning*, 2021.

Yuhai Wang, Jiawei Xia, Rongxuan Zhou, Xiao Hu, Yongliang Shi, Jing Du, and Yang Ye. PRISM: Prior-guided imagination sampling in world models. *arXiv preprint arXiv:2606.07974*, 2026a.

Zijie Wang, Wei Zhang, Weiming Zhang, Fanqi Zhang, Xiao Tan, Yipeng Qin, and Guanbin Li. World models as group actions. *arXiv preprint arXiv:2605.24578*, 2026b.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. Daydreamer: World models for physical robot learning. *arXiv preprint arXiv:2206.14176*, 2022.

Haodong Yan, Jiaguan Zhu, Mingyuan Jia, Ruiqing Yin, Junjie He, Zhide Zhong, Junfeng Li, Jinxuan Lu, Hengtao Li, Tianran Zhang, Jiayi Chen, Wenxuan Song, Wen Chen, Yuxiang Gao, and Haoang Li. Is forward prediction enough? physical state grounding for JEPA world models. *arXiv:2608.06799*, 2026.

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, and Nicolas Ballas. Hierarchical planning with latent world models. *arXiv:2604.03208*, 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:2606.31232*, 2026b.

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: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. In *ICML*, 2025.

A Supplementary Materials

A.1 Implementation and Reproducibility Details

Training procedure. Algorithm 1 gives the training procedure, including the temporary stop-gradient on the long-horizon inverse objective. The schedule and numerical settings are specified below for all experiments.

Algorithm 1 Progressive Training of LS-JEPA

Require: Trajectory window $(o_{t:t+H}, a_{t:t+H-1})$, horizon H
Encode observations: $z_{t:t+H} \leftarrow E_\theta(o_{t:t+H})$
Predict $\hat{z}_{t+1} \leftarrow P_\phi(z_t, a_t)$ and compute $\mathcal{L}_{sp}$
Decode $\hat{a}_t \leftarrow D_\psi(z_{t+1} - z_t, 1)$ and compute $\mathcal{L}_{sa}$
$\mathcal{L} \leftarrow \lambda_{sp}\mathcal{L}_{sp} + \lambda_{sa}\mathcal{L}_{sa}$
if long-horizon objectives are active **then**
    Recursively roll out P_ϕ for H steps and compute $\mathcal{L}_{lp}$
    $d_{t,H} \leftarrow z_{t+H} - z_t$
    if before full joint optimization **then**
        $\tilde{d}_{t,H} \leftarrow \mathrm{sg}(d_{t,H})$
    else
        $\tilde{d}_{t,H} \leftarrow d_{t,H}$
    end if
    Decode $\hat{\mathbf{a}}_{t:t+H-1} \leftarrow D_\psi(\tilde{d}_{t,H}, H)$ and compute $\mathcal{L}_{la}$
    $\mathcal{L} \leftarrow \mathcal{L} + \lambda_{lp}\mathcal{L}_{lp} + \lambda_{la}\mathcal{L}_{la}$
end if
Update E_θ, P_ϕ, and D_ψ by minimizing $\mathcal{L}$

LSAD architecture. LSAD is a three-layer non-causal cross-attention Transformer shared across temporal scales. Each MultiScaleDecoderBlock contains self-attention, cross-attention and a feed-forward module. An MLP projects the 192-dimensional endpoint displacement $d_{t,k}$ into a 512-dimensional transition memory, which is combined with an embedding of horizon k. Learnable action queries attend to this memory through eight heads, each with 64 dimensions. The feed-forward width is 512, dropout is zero, and the maximum action horizon is five. With action dimension 10, LSAD has 8,268,170 parameters ($\sim$8.27M); this count changes slightly with the task-specific action head.

We use the same decoder for $k = 1$ and $k = H$, so both objectives train the shared inverse-dynamics model. For fixed-horizon supervision, sliding windows provide endpoint transitions at different positions within each trajectory.

Progressive objective schedule. We start with local prediction and action recovery, then introduce the long-horizon objectives. The loss weights $(\lambda_{\mathrm{sp}}, \lambda_{\mathrm{sa}}, \lambda_{\mathrm{lp}}, \lambda_{\mathrm{la}})$ are $(1.0, 10.0, 0, 0)$ in epoch 1 and $(1.0, 10.0, 0.5, 1.0)$ in epoch 2. During epoch 2, we detach the endpoint displacement passed to the long-horizon action decoder, blocking gradients from that loss to the encoder. In epoch 3, we restore this gradient path and use weights $(1.0, 10.0, 1.0, 1.0)$. Epochs 4–10 use $(1.0, 10.0, 1.0, 2.0)$, with the local objectives active throughout training.

Optimization and compute. We train a separate model from scratch for each environment. The encoder is a ViT-Tiny/14 with a 192-dimensional representation, three context observations ($C = 3$), and a default horizon of $H = 5$. Images are resized to 224×224 and normalized with ImageNet statistics. All environments use frame skip five, so $H = 5$ spans 25 simulator steps and matches the downstream goal interval.

The complete model has approximately 26.3M trainable parameters and is trained on a single NVIDIA RTX 4090 GPU. We train for 10 epochs using AdamW, a learning rate of 10^{-4}, weight decay of 10^{-3}, and batch size 98 on every task. We use BF16 mixed precision, gradient clipping at 1.0, and linear learning-rate warmup followed by cosine decay. An epoch takes approximately 1.8 hours on Push-T, OGBench-Cube and Reacher, and 0.7 hours on Two-Room.

Algorithm 2 One-Step Refinement

Require: Initial observation o_1, goal observation o_g, horizon H
$z_1 \leftarrow E_\theta(o_1)$
$z_g \leftarrow E_\theta(o_g)$
$\hat{\mathbf{a}}_{1:H} \leftarrow D_\psi(z_g - z_1, H)$
for $h = 1, \ldots, H-1$ **do**
  $\hat{z}_{h+1} \leftarrow P_\phi(z_h, \hat{a}_h)$
  $a_h^{\text{exec}} \leftarrow D_\psi(\hat{z}_{h+1} - z_h, 1)$
  Execute a_h^{exec} and observe o_{h+1}
  $z_{h+1} \leftarrow E_\theta(o_{h+1})$
end for
$a_H^{\text{exec}} \leftarrow D_\psi(z_g - z_H, 1)$
Execute a_H^{exec}

A.2 Alternative Inference Strategies

We describe three alternatives to Single-Pass decoding and TTR: Pure CEM, One-Step Refinement, and Decoder-Guided CEM. Given an initial observation o_1 and a goal observation o_g, we define $z_1 = E_\theta(o_1)$ and $z_g = E_\theta(o_g)$. At the default horizon, a plan contains $H = 5$ action blocks, each spanning five environment steps. Indices below refer to action blocks rather than individual environment steps.

Pure CEM planning. CEM samples candidate action sequences $\mathbf{a}_{1:H}$ and evaluates them through recursive forward prediction, starting from $\hat{z}_1 = z_1$:

$$\hat{z}_{h+1} = P_\phi(\hat{z}_h, a_h), \qquad h = 1, \ldots, H. \tag{11}$$

Each sequence is scored by the discrepancy between its predicted terminal state and the goal,

$$J(\mathbf{a}_{1:H}) = d_z(\hat{z}_{H+1}(\mathbf{a}_{1:H}), z_g)\,, \tag{12}$$

where lower scores indicate closer predicted endpoints. At each iteration, the lowest-scoring candidates are used to update the sampling distribution. We use 300 candidates per iteration and 30 iterations, for a total of 9,000 candidate sequence evaluations. Pure CEM uses the encoder and forward predictor without calling LSAD.

One-Step Refinement. We first decode an initial plan $\hat{\mathbf{a}}_{1:H} = D_\psi(z_g - z_1, H)$. Before each of the first $H-1$ blocks, we encode the current observation and predict the next latent state under the corresponding initial action. LSAD then decodes the predicted displacement with horizon one, and the resulting action block is executed. For the final block, we decode directly from the observed current-to-goal displacement. Algorithm 2 summarizes the procedure.

Decoder-Guided CEM. We decode $\hat{\mathbf{a}}_{1:H} = D_\psi(z_g - z_1, H)$ and execute its first $H-1$ blocks open-loop. We then encode the new observation as $z_H = E_\theta(o_H)$ and use CEM to optimize only the final action block:

$$J(a) = d_z(P_\phi(z_H, a), z_g)\,. \tag{13}$$

The search uses 16 candidates per iteration over two iterations, with optional initialization from the original final action $\hat{a}_H$. This requires 32 candidate block evaluations, each covering the final five environment steps. Algorithm 3 gives the procedure; deployment results and planning costs are reported in Sec. 4.3.

A.3 Test-Time Replanning Procedure

TTR uses a fresh observation to replace the last block of the decoded plan. With $H = 5$, we execute the first four blocks, re-encode the observed state, and decode one final block towards the goal. The procedure uses neither the forward predictor nor candidate search at inference (Algorithm 4).

Algorithm 3 Decoder-Guided CEM

Require: Initial observation o_1, goal observation o_g, horizon H
$z_1 \leftarrow E_\theta(o_1)$
$z_g \leftarrow E_\theta(o_g)$
$\hat{\mathbf{a}}_{1:H} \leftarrow D_\psi(z_g - z_1, H)$
Execute $\hat{a}_1, \ldots, \hat{a}_{H-1}$ open-loop
Observe o_H and encode $z_H \leftarrow E_\theta(o_H)$
Initialize the CEM distribution, optionally using $\hat{a}_H$
for $j = 1, \ldots, 2$ **do**
    Sample 16 candidate action blocks
    Evaluate each candidate using $J(a) = d_z(P_\phi(z_H, a), z_g)$
    Update the sampling distribution using the lowest-scoring candidates
end for
Select the best candidate a^*
Execute the final action block a^*

Algorithm 4 Test-Time Replanning

Require: Initial observation o_1, goal observation o_g, horizon H
$z_1 \leftarrow E_\theta(o_1)$
$z_g \leftarrow E_\theta(o_g)$
$\hat{\mathbf{a}}_{1:H} \leftarrow D_\psi(z_g - z_1, H)$
Execute $\hat{a}_1, \ldots, \hat{a}_{H-1}$ open-loop
Observe o_H and encode $z_H \leftarrow E_\theta(o_H)$
$a_H^{\text{exec}} \leftarrow D_\psi(z_g - z_H, 1)$
Execute the corrected final action block a_H^{exec}

A.4 Additional Rollout Diagnostics

Fig. 6 includes Two-Room alongside the other three environments. Under the same recorded actions, LS-JEPA has smaller prediction errors in its own latent space than LeWM has in its latent space. Sec. 4.4 discusses these results alongside the separate goal-offset planning experiment. Appendix A.14 provides absolute statistics and examines how representation scale affects the comparison.

A.5 Attention Rollout on Matched Observations

We use residual attention rollout to inspect which image regions receive attention in the learned encoders. Both models receive identical frames, and we aggregate CLS-to-patch attention across all ViT layers. Each map is normalized independently, so we compare spatial patterns rather than absolute attention values.

On OGBench-Cube, LS-JEPA attention extends along the gripper–object interaction, while LeWM highlights more localized salient regions. On Push-T, LeWM attends to much of the T-shaped object; LS-JEPA focuses more on the pusher and nearby contact regions. Both encoders attend to the arm on Reacher, with LS-JEPA responses more concentrated along the links and near the end effector. These examples suggest that LS-JEPA's attention is more closely associated with the parts involved in motion and contact.

The difference is less clear on Two-Room, where LS-JEPA attends to the agent and several separated regions of the scene. Those responses alone do not reveal a route or a planned trajectory. More generally, attention rollout cannot establish which visual features cause a control decision. The patterns are compatible with learning to predict transitions and recover actions, but they do not explain the performance difference between the models.

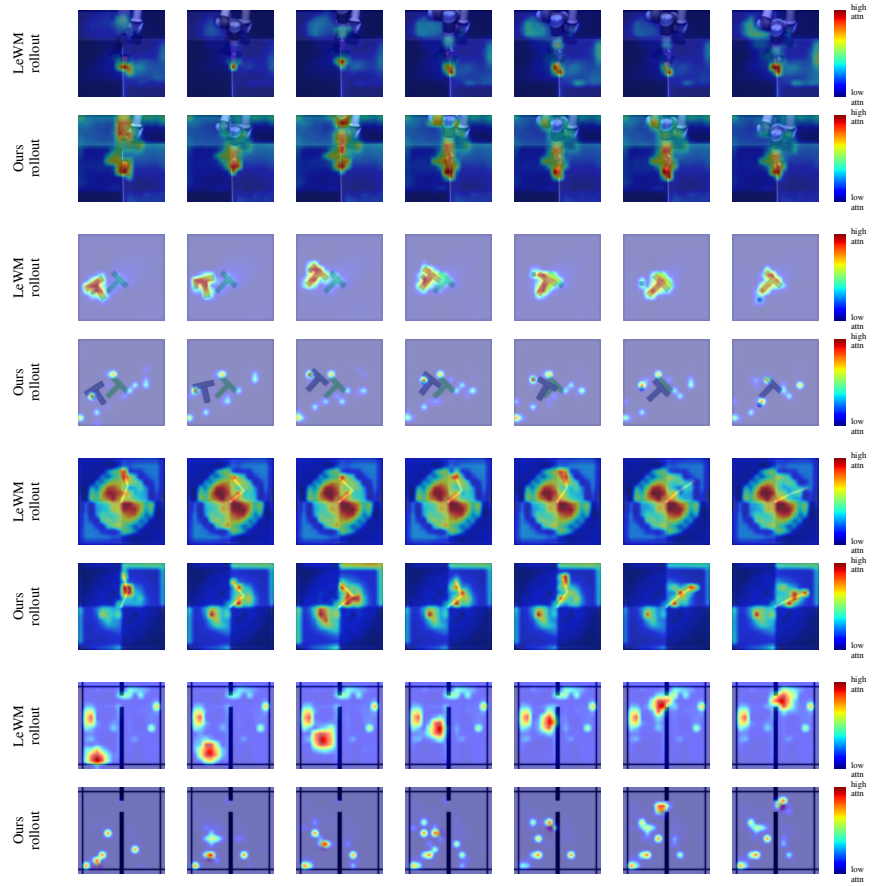


Figure 8: **Attention rollout on matched frames.** Rows show OGBench-Cube, Push-T, Reacher and Two-Room, with residual CLS-to-patch attention aggregated over all ViT layers for each model. In the manipulation and articulated-control examples, LS-JEPA attends to contact regions and moving structures, while LeWM often highlights larger salient regions. Each map is normalized independently, so colours can be compared spatially within a map but not in magnitude across maps.

A.6 Matched Qualitative Control Trajectories

We compare trajectories from identical start states to identical goals on Push-T, Reacher and Two-Room. OGBench-Cube is shown in Fig. 3. The three trajectories use expert actions, LeWM with Pure CEM 300×30, and LS-JEPA with TTR, respectively. Red boxes mark LeWM frames after visible divergence; green boxes mark successful terminal states.

In these examples, the methods remain close early in execution and separate later. On Push-T, LeWM begins to deviate around $t = 14$, after which the pusher and object no longer follow the expert progression. LeWM gradually reaches an incorrect arm configuration on Reacher and departs from the successful path after the midpoint on Two-Room. LS-JEPA with TTR stays closer to the expert progression and reaches the goal in all three examples.

CEM ranks action sequences using the learned predictor, so more samples need not resolve errors in that predictor. Here, LeWM uses 9k sampled sequences, whereas LS-JEPA uses a decoded plan and one feedback correction without candidate sampling. These selected trajectories show how the methods can differ during execution; they neither identify the cause of failure nor estimate its frequency. We assess performance across runs using the multi-seed success rates and report prediction errors separately.

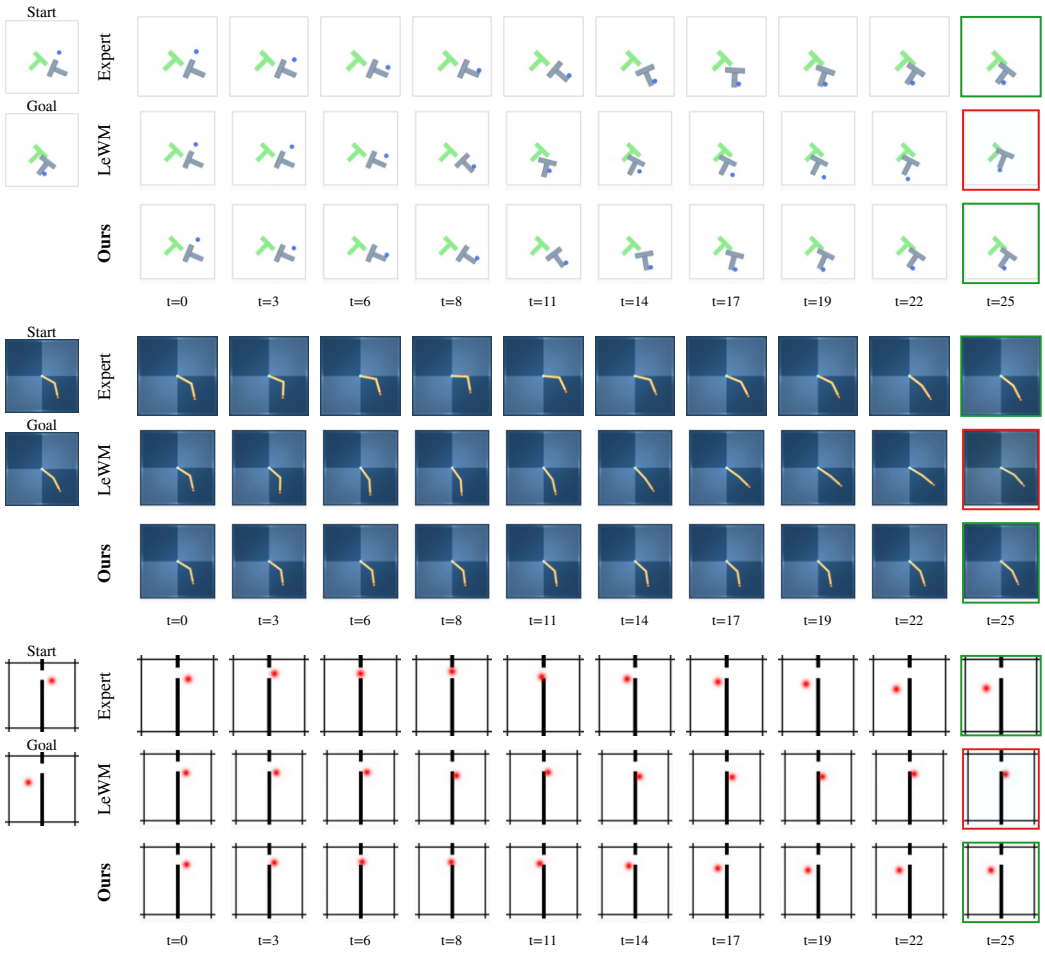


Figure 9: **Matched control trajectories on Push-T, Reacher and Two-Room.** The expert, LeWM with Pure CEM 300×30, and LS-JEPA with TTR share each start–goal pair and run for 25 simulator steps. LeWM diverges late in these examples, while LS-JEPA reaches the goal. Displayed trajectory times for LS-JEPA and LeWM are 0.5 s and 2.1 s on Push-T, respectively. The corresponding times are 0.8 s and 2.9 s on Reacher, and 0.4 s and 2.1 s on Two-Room. Red boxes mark LeWM frames after divergence; green boxes mark successful terminal states.

A.7 Additional Matched CEM Trajectories

Fig. 10 shows a second set of matched start–goal comparisons on OGBench-Cube, Push-T and Reacher. The Cube and Push-T sequences span 25 simulator steps; the Reacher sequence extends to 38.

The CEM trajectories initially look plausible, with visible divergence around $t = 22$–25 on Cube and $t = 14$ on Push-T. On Reacher, the discrepancy becomes visible around $t = 21$ and persists through $t = 38$. LS-JEPA reaches the target in each of these examples, as do the expert trajectories.

The longer Reacher sequence makes the distinction between early progress and eventual success particularly clear. An initially small discrepancy can persist through later decisions until the target is missed. The frames cannot separate prediction error from search or execution error, so we use them to illustrate failure timing rather than diagnose its cause.

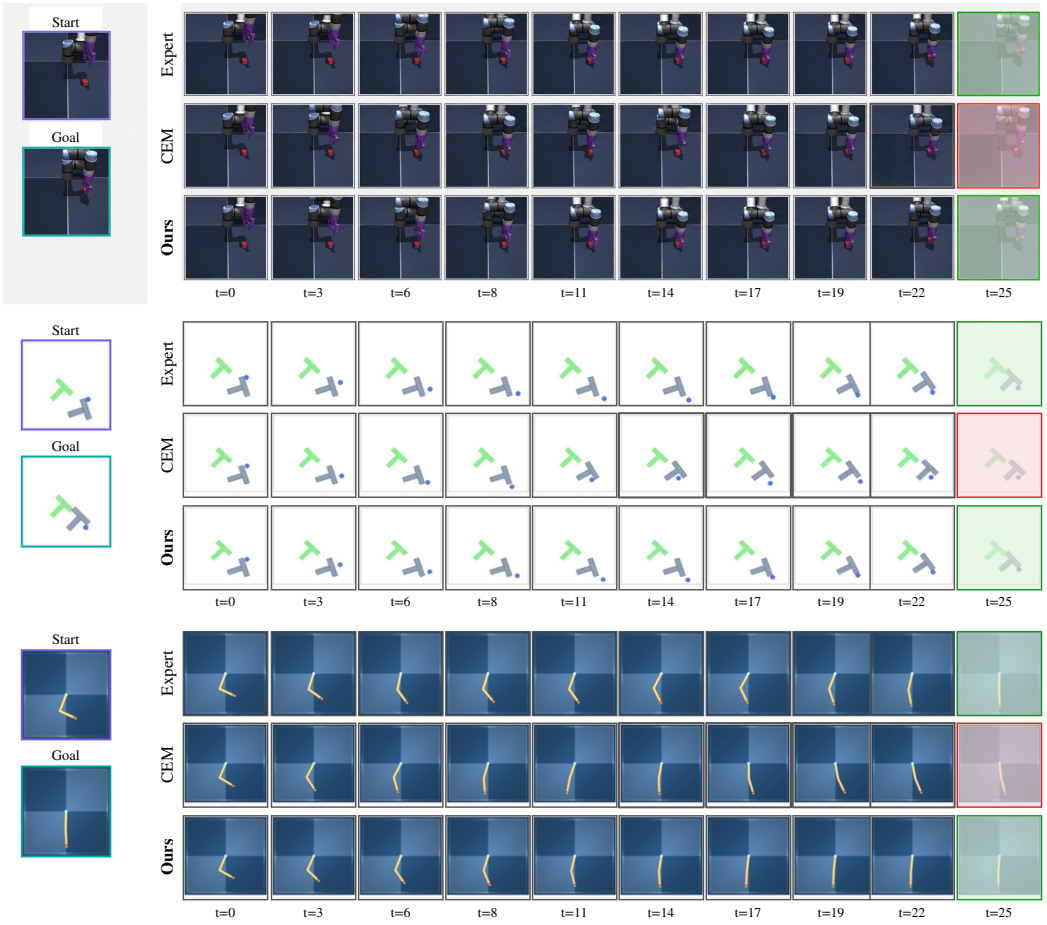


Figure 10: **Further examples of late divergence under CEM.** Rows show matched expert, CEM and LS-JEPA trajectories on OGBench-Cube, Push-T and Reacher. Cube and Push-T span 25 simulator steps; Reacher spans 38. The CEM trajectories initially approach the goal but depart later in execution. Red boxes mark frames after divergence; green boxes mark successful expert and LS-JEPA terminal states.

A.8 Action-Conditioned Latent Transition Geometry

We examine how the latent response changes when we vary the action on Push-T. The 17 categories comprise zero action and eight planar directions at magnitudes 0.5 and 1.0. Both models use the same action labels, and their responses are projected onto two principal components.

LS-JEPA's action representatives are more separated in the displayed projection, while LeWM's overlap more strongly near the centre. Several direction–magnitude pairs occupy nearby but distinct regions, and the zero-action response is separated from many non-zero actions. This is compatible with a displacement-based decoder: different actions need to produce distinguishable transitions for inverse recovery to be possible.

The projection does not test separability in the full representation or define physical action coordinates. We therefore use frozen action probes in Appendix A.12 to quantify recoverability beyond the visual comparison.

A.9 Prefix Conditioning for Ambiguous Inverse Dynamics

Alternative design. More than one action sequence can connect the same pair of endpoints. We tested whether giving the inverse decoder intermediate trajectory information would improve the

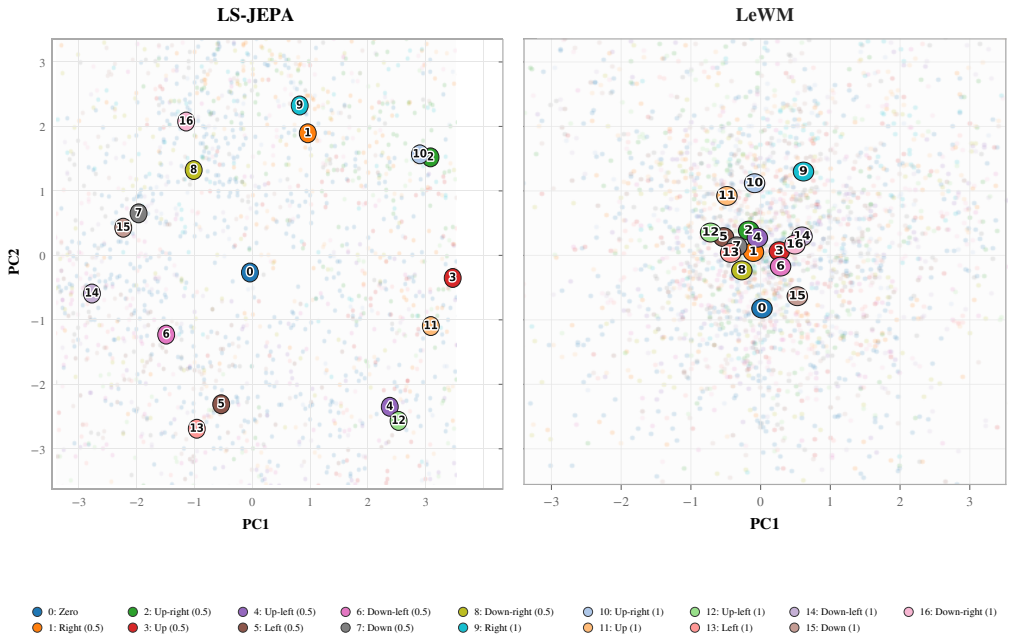


Figure 11: **Action-conditioned latent responses on Push-T.** Both models are evaluated with 17 action categories: zero action and eight planar directions at magnitudes 0.5 and 1.0. Numbered markers identify categories; background points show the projected response distribution. LS-JEPA responses are more separated in the displayed principal components, whereas LeWM responses overlap more near the centre. The PCA axes do not define physical directions or calibrated distances.

representation learned during training. The default LSAD receives only the endpoint displacement,

$$d_{t,H} = z_{t+H} - z_t, \tag{14}$$

whereas the alternative uses cumulative displacements from the initial state,

$$d_{t,h} = z_{t+h} - z_t, \qquad h = 1, \ldots, H, \tag{15}$$

and passes their sequence to the inverse decoder,

$$\mathbf{D}_{t,H} = (d_{t,1}, d_{t,2}, \ldots, d_{t,H}) \tag{16}$$

A lower-triangular attention mask restricts each action query to its corresponding latent prefix. These prefixes can distinguish trajectories that have the same endpoint displacement but visit different intermediate states.

Matched planning evaluation. We evaluate both variants with the same Pure CEM planner, which uses the encoder and forward predictor without executing LSAD. This comparison tests whether the extra inverse-training information improves the model used for planning. It gives neither variant access to intermediate demonstration states at inference.

The added conditioning helps Reacher but does not improve performance consistently across tasks. Reacher rises from 86.00% to 90.00%, while Push-T falls from 94.00% to 90.00%. Cube changes from 80.67% to 80.00%, and Two-Room remains at 100.00%. The macro averages are almost identical, at 90.17% and 90.00%. These point estimates give us no reason to prefer prefix conditioning across the four tasks.

What the extra conditioning changes. Intermediate states can make action reconstruction easier without making the endpoint displacement more useful for planning. One possible explanation is that prefixes reduce the need to encode action information in $d_{t,H}$ itself, although this experiment does not isolate that effect. The prefixes also require demonstration states that are unavailable from a current–goal pair at deployment.

Table 5: **Endpoint versus prefix conditioning for inverse dynamics.** Both variants are evaluated with the same Pure CEM planner, which does not use the inverse decoder at inference. The prefix-conditioned variant receives intermediate cumulative latent displacements during training. Entries are success-rate point estimates (%); the macro averages differ by 0.17 percentage points.

Variant	Push-T	Cube	Reacher	Two-Room	Macro
LS-JEPA, endpoint-conditioned	**94.00**	**80.67**	86.00	100.00	**90.17**
+ prefix-conditioned multimodality module	90.00	80.00	**90.00**	100.00	90.00
Δ (prefix $-$ endpoint)	-4.00	-0.67	$+4.00$	0.00	-0.17

Table 6: **Push-T decoder-input ablation: latent displacement vs. endpoint concatenation.** Success rates (%) are mean $\pm$ sample standard deviation over three seeds. The best result in each row is shown in bold. Δ denotes endpoint concatenation minus latent displacement in percentage points.

Deployment	Δz	$[z_t; z_g]$	Δ
Pure CEM 300×30	94.00±2.00	91.33±3.51	−2.67
Direct	92.33±3.21	93.33±0.58	+1.00
Decoder-Guided CEM 16×2	94.33±2.31	96.00±1.73	+1.67
TTR	96.00±3.00	94.00±1.00	−2.00

We retain endpoint conditioning because it keeps the inverse objective tied to the information available at deployment,

$$(z_t, z_{t+H}) \longrightarrow d_{t,H} \longrightarrow a_{t:t+H-1}, \tag{17}$$

and the tested alternative gives no consistent planning gain. The improvement on Reacher still leaves room for better ways to handle inverse ambiguity. A probabilistic endpoint-conditioned decoder could represent several valid action sequences without requiring the demonstrated intermediate states.

A.10 Decoder Input: Latent Displacement or Endpoint Concatenation

Setup. On Push-T, we replace the default LSAD input $\Delta z = z_g - z_t$ with the concatenated endpoints $[z_t; z_g]$. All other model and training settings are held fixed. Evaluation follows Table 1, with three seeds and 100 episodes per seed. We compare Direct, Pure CEM, Decoder-Guided CEM 16×2, and TTR. In this ablation, Decoder-Guided CEM warm-starts full-horizon search with the decoded sequence; TTR uses one feedback decode, and Pure CEM bypasses LSAD.

Results. As shown in Table 6, neither input is better under every deployment strategy. Concatenation improves Single-Pass from 92.33% to 93.33% and Decoder-Guided CEM from 94.33% to 96.00%. Pure CEM falls from 94.00% to 91.33%, and TTR from 96.00% to 94.00%. The best-performing deployment also changes: TTR leads with displacement, whereas Decoder-Guided CEM leads with concatenation.

Pure CEM does not execute the decoder, so its change must be considered in the context of joint training rather than decoder inference alone. Changing the decoder input may alter the learned representation, but this comparison does not isolate that mechanism. Nor does the frozen linear probe predict the deployment ordering: it favours displacement on Push-T, with $R^2 = 0.279$ versus 0.213 for concatenation. A linear readout and a jointly trained nonlinear decoder need not respond to these inputs in the same way.

We keep displacement as the default transition bottleneck used in the four-task experiments. This ablation gives no consistent reason to replace it, but is limited to Push-T under the matched goal interval. We have not tested concatenation under goal-offset mismatch or on the other three tasks.

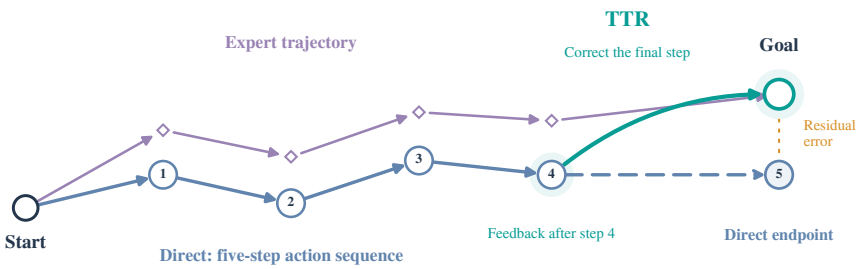


Figure 12: **A single observation-based correction.** The schematic shows an initial decoded plan that leaves a residual goal error after its fifth action block. TTR observes the state after the fourth block and replaces the fifth action with a new decode. The two branches are alternative final actions, rather than consecutive actions. This is a conceptual illustration, not a measured trajectory or a guarantee of exact goal reaching.

Table 7: **Offline expert alignment of Direct and TTR.** Mean pointwise Action R^2 for normalized $H = 5$ action blocks. Both methods omit forward-model rollouts and candidate search; only TTR receives a real intermediate observation. The comparison therefore uses different information at inference time.

Method	Forward-model rollout	Intermediate observation	Mean Action R^2 ↑
Direct	No	No	0.922
TTR	No	Yes	0.969

A.11 Offline Expert Alignment and Feedback Correction

Action decoding and feedback. Direct and TTR use the encoder and LSAD without calling the forward predictor at inference. The predictor contributes through joint training, rather than by simulating or scoring candidate actions during deployment. Direct decodes all five blocks at once; TTR observes the state after four blocks and decodes the final block again. Fig. 12 illustrates how this last observation can change the final action.

Offline alignment metric. We measure agreement with recorded expert actions using a separate R^2 for each window. For window w, the normalized predicted and expert $H = 5$ action blocks are flattened into vectors of length D,

$$R^2_w = 1 - \frac{\sum_{j=1}^{D}(a^{\mathrm{gt}}_{w,j} - a^{\mathrm{pred}}_{w,j})^2}{\sum_{j=1}^{D}(a^{\mathrm{gt}}_{w,j} - \bar{a}^{\mathrm{gt}}_{w})^2}, \qquad \bar{a}^{\mathrm{gt}}_{w} = \frac{1}{D}\sum_{j=1}^{D} a^{\mathrm{gt}}_{w,j}. \tag{18}$$

and we report the arithmetic mean of R^2_w across windows. This differs from pooling all windows into a single regression score and from fitting a frozen linear probe.

Direct already agrees closely with the recorded actions, with mean Action $R^2 = 0.922$. TTR reaches 0.969 after using the intermediate observation to update the final action. That additional information matters when interpreting the difference: the two methods are not evaluated with identical inputs. TTR changes the action through observation-conditioned decoding, without forward simulation or parameter updates.

Agreement with one recorded sequence is only one measure of action quality. Other sequences may reach the same goal, and high offline agreement need not imply successful execution. We therefore evaluate goal reaching separately in the closed-loop control experiments.

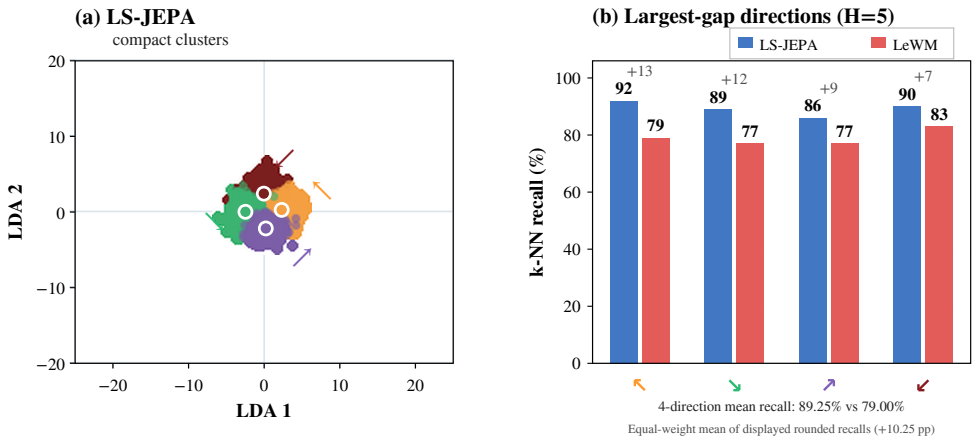


Figure 13: **Selected action-direction probes on Push-T.** The four diagonal integrated-motion classes were selected for their largest observed k-NN recall advantages over LeWM at $H = 5$. (a) Supervised LDA projection of frozen LS-JEPA endpoint displacements for these classes. (b) Per-direction recalls are $92/89/86/90\%$ for LS-JEPA and $79/77/77/83\%$ for LeWM. Their equally weighted means are 89.25% and 79.00%, giving a 10.25-percentage-point difference. These means use the displayed integer-rounded recalls; the 89.25% label in (a) refers to this mean, not classification accuracy in the projection. The selected subset does not summarize all nine direction classes.

A.12 Extended Action Recoverability Analysis

Selected direction classes on Push-T. We first inspect four diagonal integrated-motion classes using frozen endpoint displacements $\Delta z = z_{t+H} - z_t$. These are the directions with the largest observed k-NN recall advantages over LeWM, so they are a selected subset of the nine-class probe. Fig. 13 pairs a supervised LDA projection of LS-JEPA features with the per-direction recalls for both models.

LS-JEPA has higher displayed recall in each selected direction, with gaps of 13, 12, 9, and 7 percentage points. The equally weighted means are 89.25% for LS-JEPA and 79.00% for LeWM, a difference of 10.25 points. These figures summarize the displayed integer-rounded recalls, not pooled accuracy or performance across all directions. Selecting the largest gaps makes the subset optimistic as a summary of the full probe. The supervised LDA projection illustrates class organization; it supplies no independent test of generalization or comparison of compactness against LeWM.

k-NN protocol. We discretize integrated velocity over an $H = 5$ chunk into a 3×3 grid. The probe uses frozen displacements from each encoder, StandardScaler, and k-NN ($k \approx 15$) on at most 4,000 windows. It uses five-fold cross-validation, with fewer folds for small classes. The LDA visualization is fitted only to LS-JEPA features from the four selected directions. Their displayed mean is calculated from existing recalls, without retraining a four-class classifier or changing the original label space. Its decimal places reflect averaging rounded values rather than additional measurement precision.

Linear-probe protocol. We also test how well a linear map can recover expert actions from frozen features. For each environment, we sample 2,000 trajectory windows and split them into 1,600 training and 400 validation examples. A Ridge regressor ($\alpha = 1.0$) predicts scaled actions from z_t, Δz, $[z_t; z_{t+H}]$, or z_{t+H}. The target is either the first action block or the full five-block chunk, which spans 25 simulator steps. For LS-JEPA, we separately report the native LSAD decoder's long-horizon NMSE. LSAD is nonlinear and jointly trained with the encoder, so its score is not a controlled comparison with frozen-probe R^2.

Table 8: **Long-horizon action recoverability from frozen latent transitions.** Validation R^2 measures linear prediction of the full $H = 5$ action chunk from Δz. LSAD NMSE is reported separately because LSAD is nonlinear and jointly trained with the encoder.

Task	LeWM R^2	LS-JEPA R^2	ΔR^2	LSAD NMSE
Push-T	0.258	**0.279**	+0.021	0.173
Cube	0.105	**0.369**	+0.264	0.182
Reacher	−0.070	−0.123	−0.053	0.963
Two-Room	−0.008	−0.014	−0.006	0.904

Table 9: **Conditioning variables for the LS-JEPA long-horizon linear probe.** Validation R^2 for predicting the full $H = 5$ action chunk from different frozen latent conditions.

Task	z_t	Δz	$[z_t; z_{t+H}]$	z_{t+H}
Push-T	0.159	**0.279**	0.213	0.060
Cube	0.342	0.369	**0.453**	0.360
Reacher	−0.128	−0.123	−0.277	−0.124
Two-Room	−0.061	−0.014	−0.129	−0.094

Manipulation tasks. The largest probe improvement occurs on OGBench-Cube (Table 8). For the full chunk, the displacement probe improves from $R^2 = 0.105$ with LeWM to 0.369 with LS-JEPA. The first-block score rises from 0.009 to 0.360 on the same task. Push-T shows smaller improvements, from 0.258 to 0.279 for the full chunk and from 0.270 to 0.312 for the first block. Thus, action information is more accessible to a linear readout on these two tasks, especially Cube.

LSAD has long-horizon NMSEs of 0.173 on Push-T and 0.182 on Cube, below the normalized errors of the corresponding linear probes. This suggests that the trained nonlinear decoder can use transition structure that the linear readout does not recover as effectively. It does not imply that this structure is itself linearly encoded.

Reacher and Two-Room. The same explanation does not carry over to Reacher and Two-Room. Both models have near-zero or negative probe R^2, and LS-JEPA does not improve the displacement probe. LSAD NMSE is also higher, at 0.963 on Reacher and 0.904 on Two-Room. Yet both tasks achieve high control success, so expert-action reconstruction alone cannot account for goal reaching.

The deployment results help distinguish these cases without resolving the mechanism. Reacher benefits substantially from TTR, consistent with feedback correcting residual open-loop errors or ambiguity. Two-Room already saturates under Direct despite weak action reconstruction, suggesting that matching the demonstrated sequence is unnecessary for these evaluated goals. These observations leave open how much the result depends on alternative valid actions or tolerance in the success criterion.

Choice of conditioning variable. As shown in Table 9, displacement has the highest long-horizon probe R^2 on Push-T, Reacher and Two-Room. For Reacher and Two-Room, however, even the highest score is negative. Cube instead favours concatenated endpoints, with $R^2 = 0.453$ versus 0.369 for displacement. Absolute endpoint information therefore helps the linear probe on Cube, while subtraction is sufficient for the better Push-T score. We use these probes to describe task-dependent action accessibility, rather than to claim that one coordinate system is uniformly best for control.

A.13 Spectral Concentration of the Learned Representation

Protocol. We examine whether variance is spread across the embedding or concentrated in a few directions. For each task, we freeze the encoder and sample 2,000 offline windows, analysing both z_t and $\Delta z = z_{t+H} - z_t$. We summarize each covariance spectrum with its participation-ratio effective

Table 10: **Effective rank and non-degeneracy diagnostics.** LS-JEPA uses a 192-dimensional embedding and rank utilization is $r_{\text{eff}}(z_t)/192$. Low effective rank indicates spectral concentration, while the feature-spread, cosine, and dead-dimension diagnostics test against trivial constant collapse.

	$r_{\text{eff}}(z_t)$		$r_{\text{eff}}(\Delta z)$		LS-JEPA diagnostics			
Task	**LS-JEPA**	**LeWM**	**LS-JEPA**	**LeWM**	**Rank util.**	**Feat. std**	**Mean cos.**	**Dead dims**
Push-T	4.85	85.67	7.52	74.43	2.5%	0.3699	0.068	0.0%
Cube	4.98	100.25	7.72	75.05	2.6%	0.4614	0.049	0.0%
Reacher	6.51	71.95	12.19	68.62	3.4%	0.1461	0.113	0.0%
Two-Room	2.60	79.53	3.54	81.97	1.4%	0.1692	0.060	0.0%

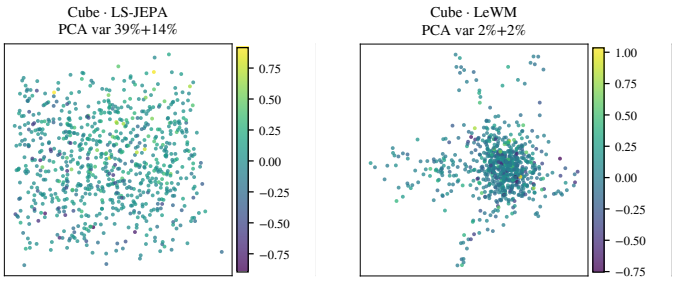


Figure 14: **Leading principal components on OGBench-Cube.** The first two components explain $39\% + 14\% = 53\%$ of LS-JEPA variance and $2\% + 2\% = 4\%$ of LeWM variance. This comparison measures spectral concentration; action recoverability is evaluated separately in Appendix A.12.

rank,

$$r_{\text{eff}}(\Sigma) = \frac{(\sum_i \lambda_i)^2}{\sum_i \lambda_i^2}, \tag{19}$$

where $\{\lambda_i\}$ are the covariance eigenvalues. We also measure feature standard deviation, mean pairwise cosine similarity, dead dimensions and cumulative variance, and inspect low-dimensional projections.

A concentrated spectrum. LS-JEPA's state effective rank ranges from 2.60 to 6.51 in a 192-dimensional embedding, compared with 71.95–100.25 for LeWM. For endpoint displacements, the ranges are 3.54–12.19 and 68.62–81.97, respectively. The leading two LS-JEPA components explain approximately 62%, 53%, 51% and 87% of variance on Push-T, Cube, Reacher and Two-Room. The corresponding LeWM projections explain only about 4–6%. The Cube example in Fig. 14 and the cross-task projections in Fig. 15 show this concentration directly.

Low effective rank raises a reasonable concern about loss of information. It does not, on its own, mean that the encoder maps every input to the same point. Here, feature standard deviations remain non-zero, mean pairwise cosine similarities are low (0.049–0.113), and no dead dimensions are detected. These checks rule against a trivial constant representation, while leaving open whether useful information has been lost in low-variance directions. Fig. 16 shows the individual diagnostics behind the heuristic warning score.

What low rank tells us about control. Effective rank measures how variance is distributed, not what that variance represents. Cube is a useful example: LS-JEPA's displacement rank is 7.72, compared with 75.05 for LeWM, yet its action probe is better. The probe rises from $R^2 = 0.105$ to 0.369, and Direct and TTR both reach 100% success. Fig. 17 shows the associated projected geometry, including latent trajectories and endpoint displacements. PCA and UMAP make this structure visible but do not identify a unique physical coordinate system.

The other tasks show why rank alone is an incomplete diagnostic. Push-T combines low rank with a positive action probe and success rates of 92.3% for Direct and 96.0% for TTR. Reacher has feature

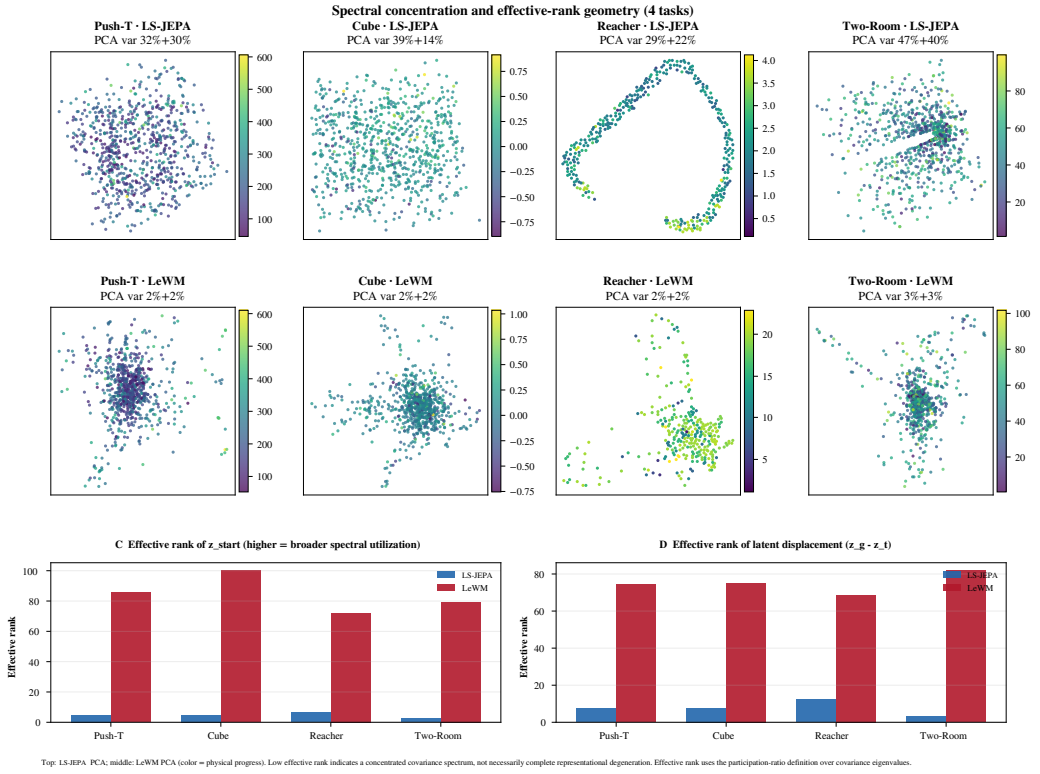


Figure 15: **PCA projections and effective rank.** LS-JEPA (top) and LeWM (middle) projections are coloured by physical progress. The lower panels report effective ranks for states z_t and endpoint displacements Δz. LS-JEPA places more variance in its leading components than LeWM.

standard deviation 0.1461 and weak linear probing; Direct reaches 80%, but TTR raises success to 99.7%. Two-Room has the lowest state effective rank and weak probing, yet Direct, Pure CEM and TTR all reach 100%. The relation between the spectrum, action reconstruction and goal reaching differs across tasks.

Interpretation. We describe LS-JEPA's representation as spectrally concentrated and anisotropic. LeWM's anti-collapse regularization favours broader spectral use, whereas LS-JEPA also trains endpoint displacements to recover actions while remaining predictive. That combination may concentrate control-related variation in fewer directions, but the present diagnostics do not test this explanation directly. They also do not establish low intrinsic dimensionality or show that every suppressed direction is irrelevant. A more direct test would project onto the top-k components and measure how control performance changes with k.

A.14 Cross-Model Rollout Comparability

Protocol and error measures. We initialize both models from the same ground-truth context and roll them forward autoregressively under the recorded offline actions. This measures recursive prediction without changes in action selection from closed-loop feedback. Table 11 reports error reductions at matched environment steps using three normalizations.

Raw L2 measures each predictor's error in its own latent coordinates. For temporal growth, we use $\tilde{e}_k = e_k/(e_1 + \epsilon)$, which expresses later error relative to each model's one-step error. Variance-normalized MSE instead divides per-dimension squared errors by ground-truth feature variance. We use the one-step ratio as the primary comparison of error growth, while retaining the other two measures to expose their different behaviour. None of these normalizations creates a shared physical-state metric across encoders.

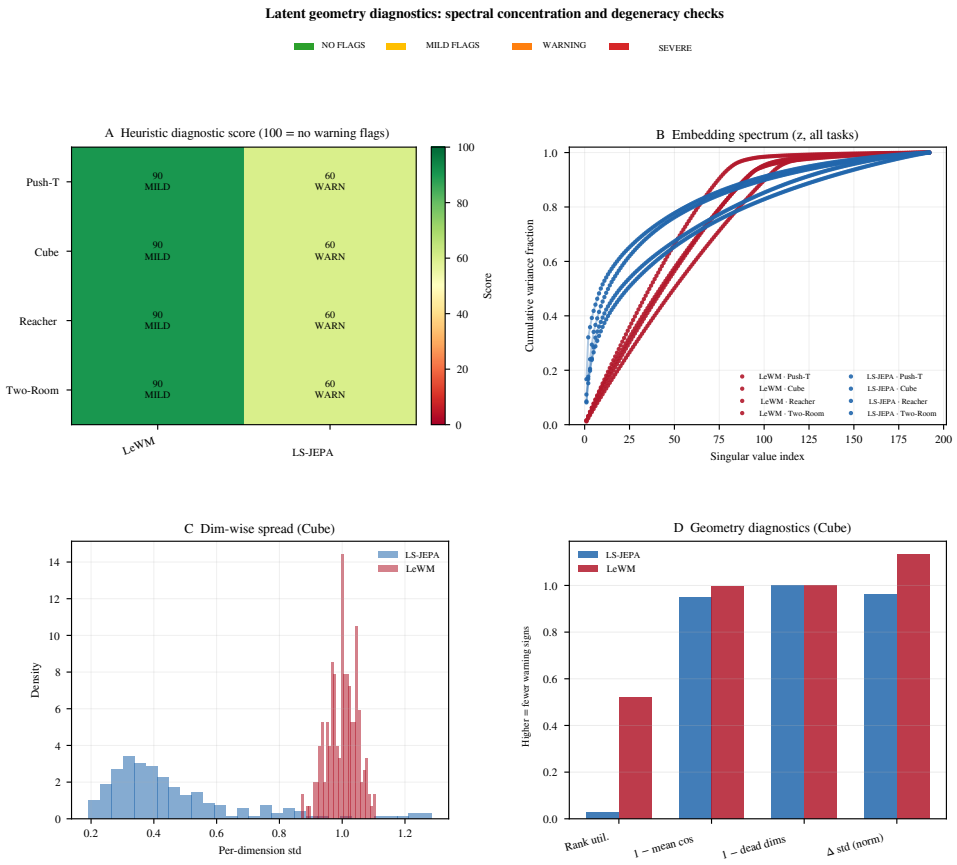


Figure 16: **Feature spread and spectral diagnostics.** The heuristic warning score combines several checks, with the spectrum, per-dimension spread and Cube metrics showing their individual contributions. LS-JEPA triggers low-rank and anisotropy warnings, but has no detected dead dimensions or near-unity mean pairwise cosine similarity. The score summarizes these checks and is not calibrated to downstream control performance.

Absolute errors and representation scale. Table 12 gives the absolute quantities behind the normalized results in Sec. 4.4. The feature variances differ substantially, especially on Reacher: 0.026 for LS-JEPA and 0.987 for LeWM. Smaller raw distances can therefore reflect a smaller representation scale as well as prediction quality. On Reacher, LS-JEPA has lower raw error and slower relative growth, but LeWM has lower variance-normalized MSE at all three reported horizons. This disagreement limits what we can conclude from the raw rollout curves.

Attempted alignment to LeWM coordinates. We also tried evaluating LS-JEPA predictions after mapping them into the frozen LeWM space with a batch-fitted ridge regression. This is a post-hoc alignment of separately learned embeddings, rather than an evaluation through a shared frozen encoder. Mapped LS-JEPA errors were larger than LeWM's at every reported horizon. At 10/25/50 environment steps, the reductions were $-276.94/-103.29/-38.19\%$ on Push-T and $-575.14/-294.38/-199.16\%$ on Cube. They were $-457.58/-335.32/-270.21\%$ on Reacher and $-197.43/-132.54/-96.55\%$ on Two-Room.

The map already has a large reconstruction residual when applied to ground-truth embeddings, before any predicted rollout is introduced. Mapped errors consequently combine prediction error with the error of the coordinate alignment. We could not use this map to obtain a clean comparison of prediction quality.

The spectral differences in Appendix A.13 offer one possible explanation for the alignment difficulty. LS-JEPA and LeWM have markedly different effective ranks and anisotropy, which a single

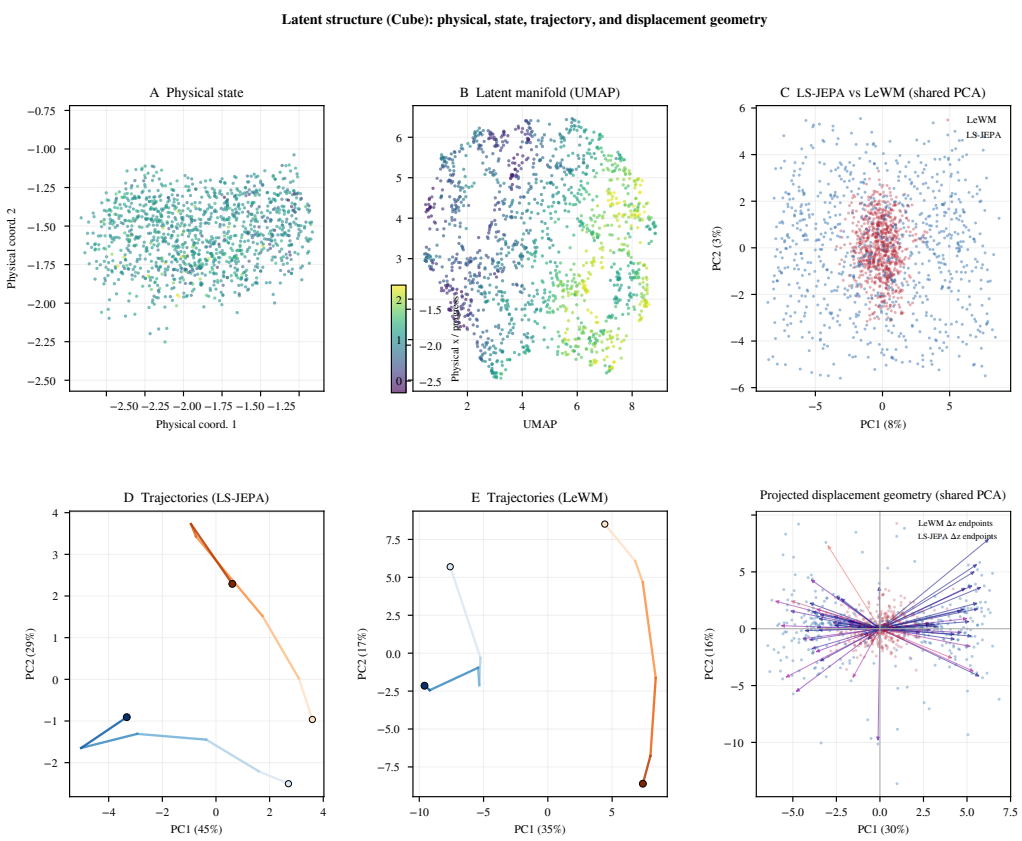


Figure 17: **Latent geometry on OGBench-Cube.** Panels show physical states, UMAP embeddings, a shared-PCA comparison, example latent trajectories and projected endpoint displacements. LS-JEPA retains structured variation despite its low effective rank on this task. Its frozen action probe also improves over LeWM, and Direct and TTR both reach 100% success. The projections do not establish a globally linear or physically calibrated latent space.

fitted linear map must accommodate. The failed alignment alone does not establish which differences cause the residual.

What the rollout comparison supports. Under recorded actions, LS-JEPA has less drift in its own latent space and slower error growth relative to its one-step error. Variance normalization supports this comparison on some tasks but favours LeWM on Reacher. We therefore do not interpret the rollout results as uniformly better physical-state prediction accuracy. Whether the learned models support successful goal reaching is assessed separately in the closed-loop experiments.

A.15 AI DISCLOSURE

In this work, we used generative AI tools to improve the clarity and readability of the manuscript, assist with literature retrieval and discovery, support algorithm implementation execution, and draft portions of the manuscript. We have not used generative AI tools for generating synthetic datasets and proving mathematical claims. We have reviewed all AI-assisted work. We examined the retrieved paper, verified the LLM-generated code by 2 of the authors and revised the generated content to achieve accurate illustration by all the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Table 11: **Reduction in rollout error relative to LeWM at matched environment steps.** Positive values indicate lower error for LS-JEPA. Raw L2 is measured in each model's native latent space and is retained for its operational relevance to model-space prediction. Relative error normalizes by each model's one-step error, whereas Var.-norm. MSE divides per-dimension squared error by ground-truth feature variance, without one-step normalization. None of these metrics removes dependence on representation geometry.

Task	Metric	10 steps	25 steps	50 steps
Push-T	Raw L2	$+48.80\%$	$+71.40\%$	$+81.28\%$
	Relative e_k/e_1	$+28.22\%$	$+59.91\%$	$+73.76\%$
	Var.-norm. MSE	-32.01%	$+62.69\%$	$+81.22\%$
Cube	Raw L2	$+54.15\%$	$+72.35\%$	$+78.16\%$
	Relative e_k/e_1	$+24.96\%$	$+54.75\%$	$+64.26\%$
	Var.-norm. MSE	$+31.57\%$	$+78.77\%$	$+85.84\%$
Reacher	Raw L2	$+65.38\%$	$+72.27\%$	$+77.82\%$
	Relative e_k/e_1	$+20.76\%$	$+36.53\%$	$+49.22\%$
	Var.-norm. MSE	-342.42%	-202.87%	-80.38%
Two-Room	Raw L2	$+88.92\%$	$+89.99\%$	$+91.21\%$
	Relative e_k/e_1	$+10.61\%$	$+19.18\%$	$+29.08\%$
	Var.-norm. MSE	$+65.62\%$	$+72.37\%$	$+76.55\%$

Table 12: **Absolute open-loop rollout statistics.** e_k is raw Euclidean model-space error, $\tilde{e}_k = e_k/e_1$, and e_k^{var} is variance-normalized per-dimension MSE. With frame skip five, $k = 5$ and $k = 10$ correspond to 25 and 50 environment steps.

Task	Model	Feat. var.	e_1	e_{25}	e_{50}	$\tilde{e}_{25}$	$\tilde{e}_{50}$	e_{25}^{var}	e_{50}^{var}
Push-T	LS-JEPA	0.148	0.768	0.910	1.084	1.185	1.413	0.0326	0.0508
	LeWM	0.996	1.076	3.181	5.794	2.956	5.384	0.0874	0.2704
Cube	LS-JEPA	0.269	0.531	0.577	0.636	1.087	1.197	0.0068	0.0082
	LeWM	1.005	0.869	2.087	2.910	2.402	3.350	0.0318	0.0582
Reacher	LS-JEPA	0.026	0.486	0.503	0.505	1.036	1.040	0.0569	0.0550
	LeWM	0.987	1.111	1.814	2.277	1.632	2.049	0.0188	0.0305
Two-Room	LS-JEPA	0.034	0.428	0.485	0.542	1.133	1.267	0.0421	0.0660
	LeWM	0.954	3.454	4.844	6.172	1.402	1.787	0.1523	0.2816