LATENT WORLD MODELS SHOULD PLAN WITH DISTANCES ALONG THE LATENT MANIFOLD

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models such as LeWM plan by searching for the action sequence whose predicted terminal latent is closest to the goal latent under a distance measured in the latent space, such as the Euclidean distance. Planning with such distances assumes that proximity in the latent space measures how far the state is from the goal. For LeWM, we find that this assumption holds only in a small neighborhood near the goal, while the cost forms spurious low-cost basins that trap the planner. Yet neighborhood-based embeddings show that the latents retain the physical state. We hypothesize that the latents lie near a curved manifold along which proximity differs from proximity in the ambient space. If so, the terminal cost should measure the distance along this latent manifold. We propose the Neighborhood Graph Cost (NGC) to estimate this distance by the shortest path on a mutual nearest-neighbor graph over the latents of training observations. NGC changes only the terminal cost and requires no retraining. Experimental results show that NGC improves the success rate of LeWM on all four tasks, with the gain growing as the goal moves farther from the start. The results support the manifold hypothesis and our argument that latent world models should plan with distances along the latent manifold.

1 INTRODUCTION

In the paradigm of planning with world models, an agent predicts the outcomes of candidate action sequences and executes the sequence whose predicted outcome scores best (Finn & Levine, 2017; Hafner et al., 2019; LeCun, 2022; Hansen et al., 2024). When the task is specified by a goal observation, the score is naturally a distance between the predicted outcome and the goal, without requiring a separately designed reward. A pixel-wise distance to a goal image is dominated by large objects such as the robot arm (Ebert et al., 2018), so latent world models measure the distance in a learned representation space instead (Zhou et al., 2025; Sobal et al., 2025; Assran et al., 2025; Terver et al., 2026; Maes et al., 2026b). All of these latent world models score the predicted terminal latent against the goal latent with a norm of their difference, such as the Euclidean distance (Maes et al., 2026b) or the L1 distance (Assran et al., 2025), and this terminal cost is the only signal that tells the planner whether it is getting closer to the goal. Planning with such a cost therefore assumes that a latent closer to the goal latent in the latent space corresponds to a state closer to the goal in the environment.

In this work, we study LeWM (Maes et al., 2026b), an end-to-end latent world model framework that uses the Euclidean distance for the terminal cost (the L2 cost), and find that this assumption fails on a simple task. In TwoRoom, a point agent moves from one room to the other through a doorway, and the observation is determined by the agent's 2D position alone. Yet LeWM reaches a success rate of only about 90% when the goal is the observation 25 steps after the start, and about 30% when it is 75 steps after the start. In every failed episode, the agent gets stuck in a region away from the goal and stays there. The cost maps of Figure 1 (top) show that the L2 cost forms a basin around the goal, but it also forms spurious low-cost basins far from it, and the planner leads the agent into them.

To understand these basins, we compare the L2 cost with the physical distance to the goal and find that it follows this distance only near the goal and flattens beyond it (Figure 2). Yet neighborhood-based embeddings such as Isomap (Tenenbaum et al., 2000) unfold the latents into a faithful two-

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

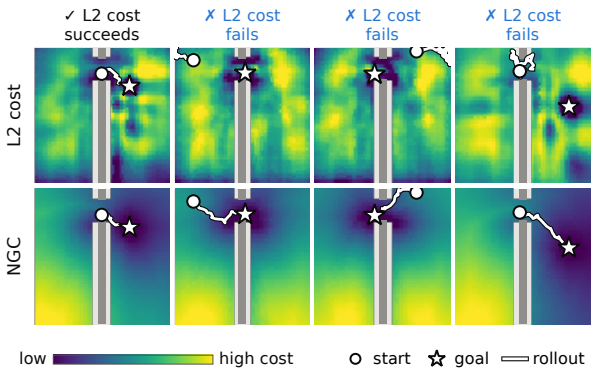

Figure 1: Cost maps and rollout trajectories of four pairs of TwoRoom. **Top:** the L2 cost (Equation 1) between the latent of the agent rendered at each feasible grid position and the goal. **Bottom:** NGC (Section 4) on the same pairs. Details in Appendix B.1.

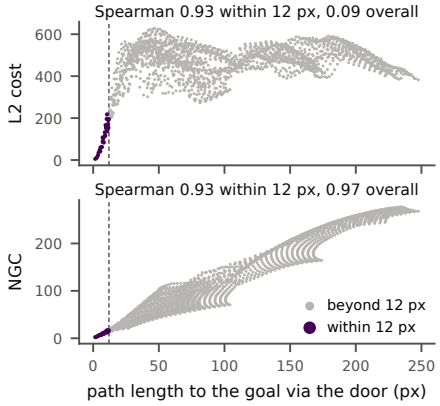

Figure 2: Terminal cost against the length of the shortest feasible path to the goal, for every feasible grid position of the rightmost pair in Figure 1. **Top:** the L2 cost. **Bottom:** NGC on the same positions.

dimensional map of the agent's position, while Euclidean-distance-based embeddings fold them. Therefore, the failures are not because the latents lose the state the task needs, but because they are read the wrong way.

These observations lead us to hypothesize that the latents of a trained LeWM lie near a curved, low-dimensional manifold, and that the terminal cost should measure the distance along this latent manifold rather than through the ambient space, the latent space in which the manifold lies. To estimate it, we propose **Neighborhood Graph Cost** (NGC). NGC connects the latents of training observations to their mutual nearest neighbors, attaches the goal latent and each predicted terminal latent to the graph, and scores a candidate by the length of the shortest path between the two, or by their Euclidean distance when they are neighbors. On the TwoRoom pairs, the NGC cost has a single basin at the goal (Figure 1, bottom).

Our manifold hypothesis predicts that the gain of NGC should grow with the distance between the start and the goal. We test the planning performance of NGC and the L2 cost on the four LeWM tasks with goals up to 100 steps after the start. The two costs perform similarly on the nearest pairs of every task, and the gain widens as the distance grows. With goals 100 steps after the start (75 on TwoRoom), NGC raises the success rate by 19, 70, 14, and 21 percentage points on Cube, TwoRoom, PushT, and Reacher. With an oracle predictor that executes every candidate in the simulator, the gain of NGC is larger still, and giving CEM four times as many candidates does not let the L2 cost reach NGC's performance. Our proposed method changes only the terminal cost and does not require any retraining.

In summary, we offer a simple explanation for the planning failures we observe in LeWM. It leads to a method that changes only the terminal cost, and the method's effectiveness in turn supports the explanation. We hope that our work leads the latent world model community to reconsider the default of measuring distances through the ambient latent space, and to plan with distances along the latent manifold instead.

2 Background and Related Work

Latent world models for goal-conditioned planning. A latent world model encodes observations into a learned latent space and predicts forward in that space, so that a planner can search over action sequences without rendering pixels (Hafner et al., 2019; LeCun, 2022). When the task is specified by a goal observation, the planner scores each candidate action sequence by a terminal cost between the predicted terminal latent and the goal latent. DINO-WM (Zhou et al., 2025) plans

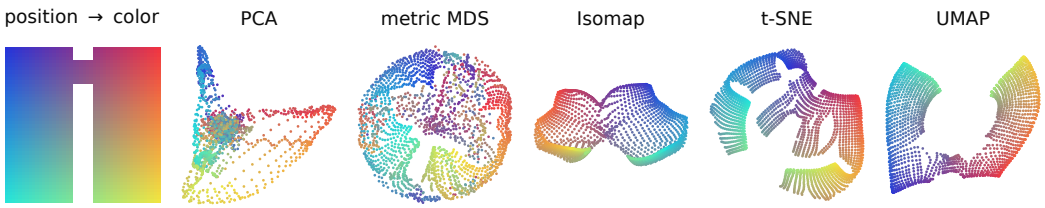

Figure 3: Two-dimensional views of the TwoRoom grid latents, each point colored by the physical position it was rendered at (**left:** the color key).

in frozen DINOv2 features; PLDM (Sobal et al., 2025) plans in a latent space trained jointly with the predictor; V-JEPA 2 (Assran et al., 2025) and Terver et al. (2026) plan with JEPA-family world models. In all cases the terminal cost is a norm of the latent difference, the Euclidean distance or the L1 distance.

We study LeWM (Maes et al., 2026b;a), which trains an encoder E and a predictor F end-to-end. Given an observation o_t, the encoder produces a latent $z_t = E(o_t)$. Given a short history of latents and actions, the predictor produces the next latent, $\hat{z}_{t+1} = F(z_{t-2:t},\, a_{t-2:t})$. By rolling F forward H steps from the current latent under a candidate action sequence $(a_t, \ldots, a_{t+H-1})$, the planner obtains a predicted terminal latent $\hat{z}_{t+H}$. The planner uses CEM (Rubinstein, 1999; de Boer et al., 2005): it samples candidate action sequences, ranks them by the terminal cost, and iteratively refines the sampling distribution toward lower-cost candidates. Writing $q = \hat{z}_{t+H}$ for the predicted terminal latent and $g = E(o_{\text{goal}})$ for the goal latent, the terminal cost is

$$C_{\text{L2}}(q, g) = \|q - g\|_2^2. \tag{1}$$

CEM ranks candidates by this cost, so planning with the L2 cost assumes that a shorter Euclidean distance to the goal latent means progress toward the goal in the environment.

Improving planning with LeWM. Many follow-ups to LeWM aim to improve its planning, most of them because the Euclidean distance between latents does not measure progress toward the goal. Their remedies fall into three families. The first *retrains the encoder* so that the distance between latents itself measures progress, whether by straightening trajectories (Wang et al., 2026b), matching latent to ground-truth state distances (Hu et al., 2026), adding reachability or trajectory heads (Li et al., 2026b; Wang et al., 2026a; Bai & Xiong, 2026; Huang et al., 2026), shaping the latent with a value (Destrade et al., 2026), or embedding in a hyperbolic space (Liu et al., 2026). The second *learns a parametric cost* on the frozen model, such as a temporal-separation regressor (Li et al., 2026a; Singh, 2026), a goal-conditioned value (Sommer & Schilling, 2026), or a ground-truth state readout (Pham & Bera, 2026). The third *changes the target* rather than the cost, planning toward a learned subgoal (Masip et al., 2026; Caselli et al., 2026; Cheng et al., 2026) or replacing the search with a learned actor (Sun et al., 2026). Each adds a learned component that must be trained beyond the original LeWM objective. We depart from all three by attributing the issue to the latents lying near a curved manifold and proposing a method to estimate the distance along it, without training a new encoder, a cost network, a subgoal generator, or a policy.

Manifold learning and shortest-path distances. NGC estimates distances along the manifold with a tool from manifold learning. Isomap (Tenenbaum et al., 2000) approximates the geodesic distance along a data manifold, by the shortest path on a k-nearest-neighbor graph weighted by Euclidean edge lengths, then embeds the resulting distance matrix with multidimensional scaling. t-SNE (van der Maaten & Hinton, 2008) and UMAP (McInnes et al., 2018) optimize different embedding objectives on similar graphs. NGC builds the same kind of graph over encoder outputs, but it never computes embeddings or coordinates. A separate line of work builds graphs over experience and plans by searching them for waypoints (Savinov et al., 2018; Eysenbach et al., 2019; Yang et al., 2020; Zhang et al., 2021). There the edges encode demonstrated transitions or learned reachability, and the output is a plan. NGC's edges encode only Euclidean proximity, and they are used to score the candidates, not to produce action sequences.

3 WHY LEWM DOES NOT SOLVE TWOROOM

We began by examining how LeWM performs on TwoRoom. Of the four tasks LeWM uses, TwoRoom appears to be the easiest: a point agent moves from one room to the other through a doorway, and since the doorway is fixed, the top-down observation is determined by the agent's 2D position alone. Yet under our evaluation protocol (Section 5.1), LeWM reaches a success rate (SR) of only about 90% on TwoRoom, compared with 99% on Reacher and 93% on PushT. The simplicity of the task also makes the failures easy to visualize and diagnose, so we set out to understand them.

Spurious low-cost basins trap the agent. We visualize the rollout trajectories of the episodes and find that in every failed episode, the agent gets stuck in a local region and stays there for the rest of the episode. To see why the planner chooses such actions, we compute a cost map for each evaluated goal: we place the agent at every feasible position on a 4 px grid, render the observation, encode it with the LeWM encoder, and compute its L2 terminal cost against the goal latent (Equation 1). The top row of Figure 1 shows four examples overlaid with the executed rollouts. First, we observe that the cost always forms a basin around the goal, which is understandable since it is zero at the goal, and nearby observations probably have similar latents. However, only a short distance away, the cost becomes high, and some regions far from the goal have a much lower cost than their surroundings. Therefore, when actions sampled by the CEM planner lead into one of these spurious basins instead of the true basin at the goal, the agent is trapped.

The L2 cost is faithful only locally. The simplicity of TwoRoom also gives a "ground-truth" cost for every position: the length of the shortest feasible path to the goal. Figure 2 plots the terminal cost against this length for the rightmost pair in Figure 1. For positions whose path length to the goal is at most 12 px, the cost ranks positions in the same order as their "ground truth" (Spearman 0.93), consistent with the basin around the goal in Figure 1. Beyond that, the cost flattens, and over all positions the Spearman correlation is near zero. The pattern holds across all 100 evaluated pairs: the two correlations are 0.82 ± 0.11 within 12 px and -0.07 ± 0.09 over all positions (mean $\pm$ standard deviation over pairs). The L2 cost flattens in the same way on all four tasks (Figure 6).

The latents retain the agent's position. Why does the cost become non-informative away from the goal? Is it because the model doesn't learn to encode the correct information, or simply because the cost fails to use it? Since a TwoRoom observation is determined by the agent's 2D position, latents that retain this information should be able to be reduced to a faithful two-dimensional embedding. Figure 3 applies five dimensionality reduction methods to the latents of the grid used in Figure 1. Isomap (Tenenbaum et al., 2000), t-SNE (van der Maaten & Hinton, 2008), and UMAP (McInnes et al., 2018) build their embeddings from each latent's nearest neighbors, and all three unfold the two rooms and recover the physical positions up to a smooth deformation (Spearman 0.80, 0.82, and 0.91 between embedding distance and physical (x, y) distance). In contrast, principal component analysis (PCA; Pearson, 1901; Hotelling, 1933), which projects the latents linearly onto their two directions of largest variance, and metric multidimensional scaling (MDS; Borg & Groenen, 2005), which preserves the Euclidean distances between the latents, both fold the room onto itself (Spearman 0.49 and 0.40). Therefore, the latents do encode the underlying physical state that the task needs, but only certain methods can read it out.

4 HYPOTHESIS AND A COST ALONG THE LATENT MANIFOLD

The observations in Section 3 lead us to the following hypothesis: the latents of a trained LeWM lie near a curved, low-dimensional latent manifold $\mathcal{M}$, and the terminal cost should measure the distance along $\mathcal{M}$ rather than through the ambient space.

The hypothesis provides clear explanations for the observations. The distance along $\mathcal{M}$ between two latents is the length of the shortest path that stays on $\mathcal{M}$, but the L2 cost (Equation 1) measures the Euclidean distance, a straight line through the ambient space. When the two latents are close, the Euclidean distance is similar to the distance along $\mathcal{M}$, so the L2 cost is faithful locally. On a curved manifold, however, the Euclidean distance stops following the distance along $\mathcal{M}$ beyond a neighborhood. Where $\mathcal{M}$ bends through many directions of the ambient space, two latents far

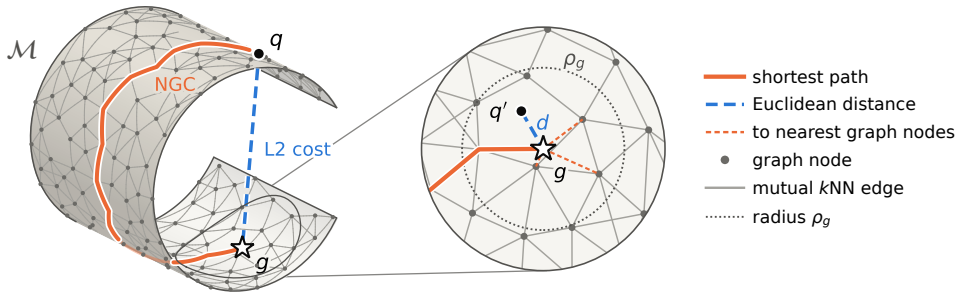

Figure 4: NGC on a curved manifold $\mathcal{M}$. **Left:** a candidate q is close to the goal g in Euclidean distance (blue, the L2 cost) but far from g along $\mathcal{M}$. NGC scores q by the shortest path through the graph (orange), whose edges join mutual 15-nearest neighbors. **Right:** the neighborhood of g, flattened. When a candidate q' and g lie within each other's neighborhood, NGC scores q' by their Euclidean distance $d = \|q' - g\|_2$. The surface is a schematic; the LeWM latents are 192-dimensional.

apart along $\mathcal{M}$ lie about as far apart as two random latents, so the L2 cost flattens away from the goal (Figure 2). Where $\mathcal{M}$ bends back, a region far from the goal along $\mathcal{M}$ can come closer to it in the ambient space than its surroundings, explaining spurious basins in the cost maps. Because neighborhoods on $\mathcal{M}$ are nearly flat, embeddings built from neighborhoods recover the position that the latents encode, while embeddings that preserve Euclidean distances fold it.

Since $\mathcal{M}$ has no explicit parameterization, we cannot compute the distance along it directly and instead estimate it from samples. On a neighborhood graph whose edges join only nearby points, each edge approximates a short stretch of $\mathcal{M}$, so the shortest-path length between two nodes estimates the distance along $\mathcal{M}$. We use this estimator to define **Neighborhood Graph Cost** (NGC, Figure 4). Offline, we encode a random subset of N training observations as graph nodes $v_1, \ldots, v_N$ and connect each pair of mutual 15-nearest neighbors with an edge weighted by their Euclidean distance $\|v_i - v_j\|_2$. We precompute the graph distance D_{ij}, the shortest-path length between v_i and v_j. Online, we attach the goal latent g and each candidate action sequence's predicted terminal latent q to their four nearest graph nodes, $\mathcal{N}_4(g)$ and $\mathcal{N}_4(q)$. The cost is

$$C(q, g) = \begin{cases} \|q - g\|_2, & \|q - g\|_2 \le \min\{\rho_q, \rho_g\}, \\ \min\limits_{\substack{i \in \mathcal{N}_4(q) \\ j \in \mathcal{N}_4(g)}} \big[\|q - v_i\|_2 + D_{ij} + \|v_j - g\|_2\big], & \text{otherwise}, \end{cases} \tag{2}$$

where ρ_q and ρ_g are the distances from q and g to their 15th-nearest graph node. When q and g lie within each other's neighborhood, their Euclidean distance is as faithful as a graph edge, so the cost uses it directly. NGC replaces only the terminal cost; the encoder and predictor stay frozen, and the planner is unchanged. Algorithm 1 in Appendix A.1 summarizes the procedure.

5 Experiments

In this section, we validate our proposed method by comparing the planning SR of the same trained LeWM with NGC against the L2 cost. We first show that the gain of NGC grows with the distance to the goal and comes from the path along the graph. We then measure the gain with an oracle predictor in place of the learned one, and ask whether the gain comes from computation.

5.1 Experimental setup

We use the four tasks and datasets released with LeWM (Maes et al., 2026b;a): **Cube**, the single-cube manipulation task of OGBench (Park et al., 2025); **TwoRoom**, a point agent moving between two rooms through a doorway; **PushT**, pushing a T-shaped block to a target pose (Chi et al., 2023); and **Reacher**, the two-link arm of the DeepMind Control Suite (Tassa et al., 2018). The world model

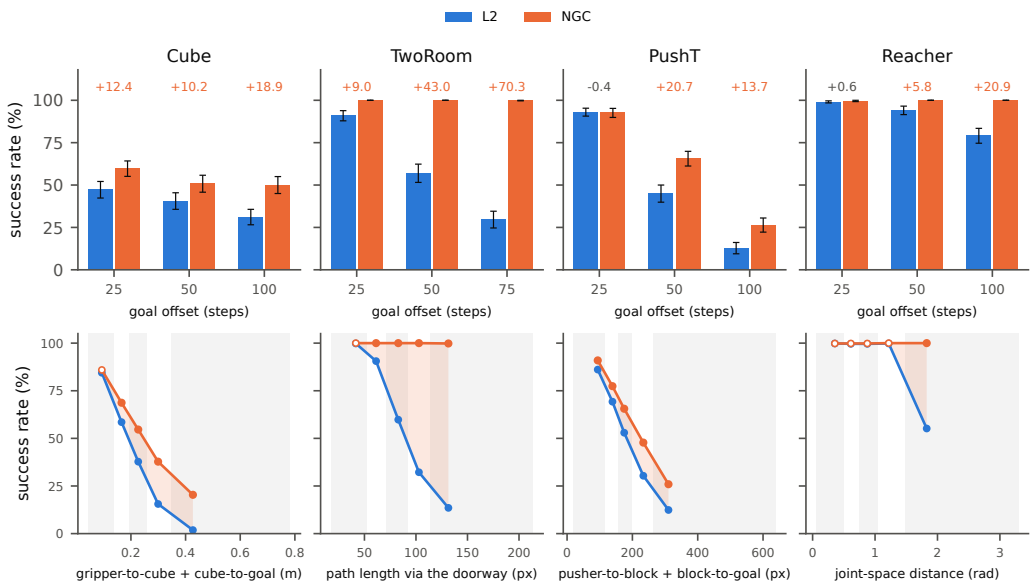

Figure 5: **Top:** Success rate against the goal offset. Bars: 300 pairs per offset; whiskers: bootstrap 95% interval over pairs. Numbers: NGC minus the L2 cost in %, orange when the paired bootstrap 95% interval over pairs excludes zero. **Bottom:** the same pairs pooled across offsets and binned into quintiles of the physical start–goal distance (180 pairs per bin); shading: NGC minus the L2 cost; filled NGC points: the gain is significant, as for the orange numbers above.

is a LeWM that we trained with the LeWM training recipe for 10 epochs. The planner setting is also inherited from the LeWM release. Details can be found in Appendix A.2.

Evaluation protocol. As in the LeWM evaluation, a *pair* is a start and a goal observation from one training episode, the goal a fixed *offset* of steps after the start, and the agent has twice the offset in primitive steps to reach the goal. Each task and offset has 300 fixed pairs, each run with three planner seeds, and we report NGC minus the L2 cost with a paired 95% bootstrap interval over pairs. Success is a task-semantic predicate checked at every step; the simulator starts at rest from the recorded pose, and replanning uses the executed history of three frames. Appendix A gives the details and a reproduction of the LeWM evaluation for comparison with published numbers.

5.2 Latent planning performance across goal distances

Our manifold hypothesis predicts that measuring distance along the manifold should matter more when the goal is farther away from the start since the L2 cost is faithful near the goal. The LeWM evaluation leaves little room to test this: with every goal 25 steps after the start, the L2 cost already solves nearly every PushT and Reacher pair. We therefore add offsets 50 and 100 (50 and 75 on TwoRoom, whose episodes end at 101 steps).

The prediction holds. Figure 5 shows that as the offset grows, the success rate of the L2 cost falls on every task, while NGC stays at the ceiling on TwoRoom and Reacher. The gain, confined to Cube and TwoRoom at offset 25 (PushT and Reacher are near the ceiling and neither changes by more than 1%), appears on all four tasks at the longer offsets.

The offset, however, is only a proxy for how far the goal is. The bottom row of Figure 5 therefore regroups the same pairs by the physical distance between the start and goal states, computed from the simulator. The gain follows this distance: the two costs perform similarly on the nearest pairs of every task, and the gain widens as the distance grows.

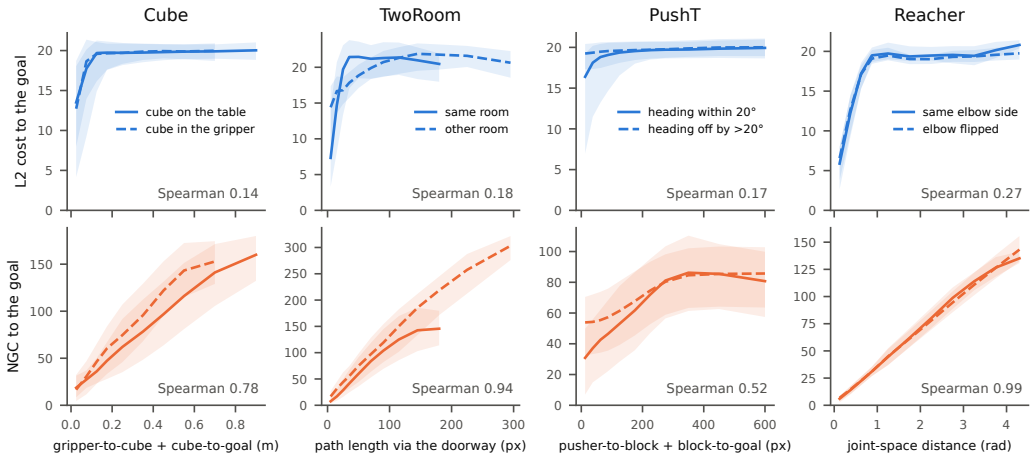

Figure 6: Terminal cost against the physical distance to the goal on the four tasks. **Top:** the L2 cost. **Bottom:** NGC. Lines are bin medians and bands the 10th–90th percentiles, for two groups of pairs per task (key in the top row); Spearman is over all pairs. Details in Appendix B.2.

Table 1: Ablation studies at the longest offset of each task. Differences in %, with a paired bootstrap 95% interval over pairs.

Task	offset	L2	rescaled L2	NGC w/o path	NGC	NGC w/o path − L2	NGC − NGC w/o path
Cube	100	31.1	33.1	30.2	50.0	-0.9 [$-3.7, +1.9$]	$+19.8$ [$+15.3, +24.2$]
TwoRoom	75	29.6	1.1	38.7	99.9	$+9.1$ [$+5.4, +12.9$]	$+61.2$ [$+56.0, +66.3$]
PushT	100	12.7	9.3	11.3	26.3	-1.3 [$-3.6, +1.0$]	$+15.0$ [$+11.2, +18.8$]
Reacher	100	79.1	78.4	81.0	100.0	$+1.9$ [$+0.4, +3.4$]	$+19.0$ [$+14.9, +23.3$]

Figures 1 and 2 give a qualitative view of why NGC helps: on the TwoRoom pairs of Section 3, the NGC cost has a single basin at the goal and grows with the path length over the whole room. On all four tasks, the NGC cost rises with the physical distance to the goal, while the L2 cost flattens (Figure 6).

On the same pairs, we also compare NGC with two methods that train a new component (Appendix D). At the longest offset, the RC-aux objective (Li et al., 2026b) continued from our checkpoint trails NGC on every task. The INTACT actor (Sun et al., 2026), which replaces the search, trails NGC on TwoRoom and PushT and ties on Reacher, but leads on Cube, where it has learned a grasp that CEM does not propose.

5.3 Ablation

Besides the path along the graph, NGC attaches q and g to graph nodes and depends on the local node density through ρ. Two controls separate these parts (Table 1). *NGC w/o path* replaces D_{ij} in Equation 2 with the Euclidean distance $\|v_i - v_j\|_2$ between the same graph nodes. The *rescaled L2 cost* $\|q - g\|_2/\sqrt{\rho_q \rho_g}$ corrects for density without the graph. NGC w/o path is indistinguishable from the L2 cost on Cube and PushT and recovers only a small part of the gain on TwoRoom and Reacher; the rescaled L2 cost gains at most 2.0% over the L2 cost and collapses on TwoRoom. Adding the path gives the rest of the gain on every task, so the gain comes from the length along the graph, i.e., the estimate of the distance along the manifold.

Table 2: Success with the learned predictor and with the oracle predictor, which executes each candidate in the simulator and encodes its terminal frame. Δ is NGC minus L2 with a paired bootstrap 95% interval over pairs.

		learned predictor			oracle predictor		
Task	offset	L2	NGC	Δ	L2	NGC	Δ
Cube	25	48	59	$+11$ $[+1, +21]$	65	87	$+22$ $[+14, +31]$
Cube	50	40	47	$+7$ $[-1, +15]$	55	63	$+8$ $[+2, +15]$
Cube	100	26	52	$+26$ $[+16, +36]$	37	73	$+36$ $[+27, +46]$
PushT	50	41	65	$+24$ $[+14, +34]$	47	76	$+29$ $[+19, +39]$
PushT	100	11	29	$+18$ $[+10, +26]$	13	38	$+25$ $[+16, +34]$

Table 3: Planning time of NGC at offset 25. Replan: median time of one CEM solve, with the ratio to L2. Evaluation: total wall-clock time of an evaluation relative to L2 on the same pairs. Objective: median time to score one pool of 300 candidates. f64 and f32: double and single precision, which give the same successes.

	Replan (s)			Evaluation ($\times$ L2)		Objective (ms per pool)		
Task	L2	f64	f32	f64	f32	L2	f64	f32
Cube	0.46	0.64 (1.41×)	0.52 (1.14×)	1.06	0.93	0.074	6.30	2.06
TwoRoom	0.46	0.53 (1.14×)	0.48 (1.05×)	0.93	0.86	0.074	2.67	1.06
PushT	0.46	0.64 (1.40×)	0.52 (1.13×)	1.35	1.11	0.074	6.30	2.06
Reacher	0.45	0.64 (1.42×)	0.52 (1.13×)	1.14	0.96	0.074	6.31	2.07

5.4 Planning with an oracle predictor

NGC replaces only the terminal cost and plans with the predictor that LeWM learned, so its gain depends on that predictor. The graph is built from encoded observations, but in planning NGC scores the terminal latents that the predictor imagines, and a poorly predicted latent attaches to the wrong graph nodes. With a poor enough predictor, NGC could not help at all. The results of Section 5.2 therefore measure NGC under an imperfect learned predictor. To remove this limit, we replace the learned predictor with an oracle predictor: the simulator executes each CEM candidate from a copy of the full simulator state, and the frozen encoder encodes its terminal frame. The encoder, the graph, the cost, and the planner are unchanged. Since the oracle simulates $9{,}000$ candidates per replan, we run it on the 100 pairs of one pair set with one planner seed, and only on Cube and PushT, where NGC stays below its ceiling.

Table 2 shows that the oracle predictor raises NGC more than the L2 cost. Therefore, the gains of Section 5.2 are measured under the limits of the learned predictor, and NGC gains more without them. However, even with the oracle predictor, NGC solves only 73 of the Cube pairs and 38 of the PushT pairs at the longest offset, indicating that the prediction accuracy is not the only bottleneck (Section 6).

5.5 Computation and planning time

Compared with the L2 cost, NGC needs extra computation, but most of it is done once before planning. The shortest paths between all graph nodes are computed once after the model is trained, and the length of the paths connecting the goal to the graph is computed once per goal. The remaining online computation for candidate scoring is small compared to predictor rollouts. As a result, NGC makes a replan only slightly slower in single precision. A whole evaluation takes about as long as with the L2 cost, because the episodes that NGC solves end early (Table 3).

Conversely, more computation does not help the L2 cost. Figure 7 scales the computation by giving CEM more or fewer candidates, from one fifth to four times the default, at the longest offset of each task. The success rate of the L2 cost barely changes across this range. Four times the default is far more computation than NGC adds to a replan. Even so, the L2 cost solves fewer pairs than NGC with the default computation on every task. What limits the L2 cost is the cost, not the computation.

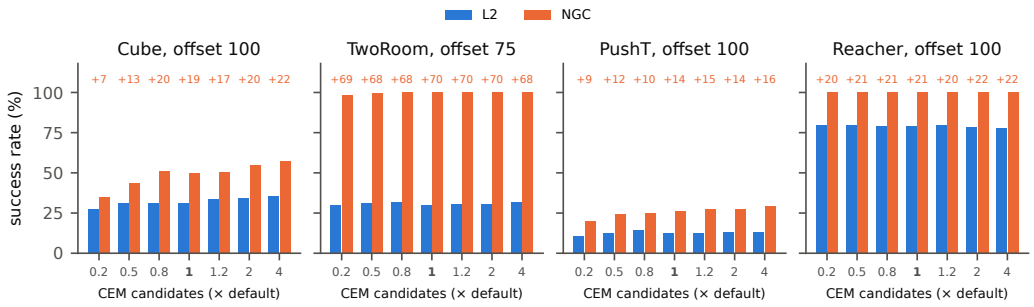

Figure 7: Success rate against the number of CEM candidates at the longest offset of each task. The number of elites is scaled with it; $1\times$ is the setting of Figure 5.

6 Discussion and Conclusion

Where does the manifold come from? LeWM is trained with two terms, and neither refers to the distance between two states. The prediction objective does not guarantee the plannability of the latents, and many LeWM follow-ups add supervision of long-range distance, such as temporal reachability or even privileged physical state (Section 2). Yet without any such supervision, distances along the latent manifold of LeWM suffice to plan on TwoRoom and Reacher and help on all four tasks. We conjecture that the manifold is formed as the model tends to learn smooth functions. As nearby states produce similar observations and similar futures, a smooth encoder and a smooth predictor fit such data, and nothing in it rewards a jump. Therefore, every latent neighborhood is faithful to the state, and because the state is low-dimensional, the neighborhoods join into a low-dimensional manifold. NGC measures the distance along this manifold as a chain of neighborhoods, so it needs nothing beyond this inherited continuity. SIGReg keeps the latents from collapsing, but also bends the manifold through many directions of the ambient space (Appendix B.3). This breaks distances measured through the ambient space but leaves the chain of neighborhoods intact.

Limitations and future work. The same argument predicts where the manifold fails, namely at contact, where the world itself jumps. A gripper a few millimeters from the cube cannot lift it, while one around it can, and a smooth world model leaves such states in one neighborhood. This is where NGC falls short. It reaches the ceiling on TwoRoom and Reacher, which move only the agent, but not on Cube and PushT, which move an object through contact, even with the oracle predictor. Contact is also where control is asymmetric, which no symmetric neighborhood can express. No readout can separate states that a neighborhood already mixes, so contact calls for a change to the representation rather than to the cost. We plan to make the neighborhoods sharp and directed at contact, for instance by orienting them with the frozen predictor. Appendix I discusses the limits of our evidence.

Conclusion. Many attempts to extend latent world models to longer horizons treat the failure as a failure of the representation, and retrain the encoder to make Euclidean distances measure progress. Our results suggest that much of this failure lies not in what the representation learned but in how the planner reads it. The prediction objective alone already yields latents whose neighborhoods are faithful, and a graph over these neighborhoods turns them into a distance that planning can use, without any new training. Therefore, we advocate that a latent world model should be built and judged around its latent neighborhoods and manifold, not by whether its latent space keeps distances meaningful at every range.

AI USE STATEMENT

In this work, we used LLM-based assistants to search for and summarize related literature, implement experiments, create visualizations, and translate the authors' intention to academic English writing. We have reviewed all AI-assisted work. Every number in the paper is copied from archived experiment reports, and every figure is generated by a script from those records. Every citation was checked against the cited work. We are aware that AI-generated texts could be painful to read and try our best to ensure the presentation is clear, correct, and readable to human readers.

ETHICS STATEMENT

This work uses only simulated benchmark environments and publicly released datasets and checkpoints. It involves no human subjects, personal data, or deployment on physical systems. We see no direct ethical concerns beyond those common to robot-learning research.

REPRODUCIBILITY STATEMENT

All experiments use the public `stable-worldmodel` package and the released LeWM datasets. Section 5.1 and Appendix A specify the checkpoints, the fixed pair sets, the graph construction rule, the planner budget, the numerical settings, and the statistical procedure. Every reported cell is backed by a per-pair outcome record. The code will be released at the anonymized repository `https://anonymous.4open.science/r/ngc-latent-manifold`.

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. URL https://arxiv.org/abs/2506.09985.

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

Ingwer Borg and Patrick J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer Series in Statistics. Springer, 2 edition, 2005. ISBN 978-0-387-25150-9. doi: 10.1007/0-387-28981-X. URL https://link.springer.com/book/10.1007/0-387-28981-X.

Niccolò Caselli, Francesco Massafra, Samuele Punzo, Salvatore Lo Sardo, Ippokratis Pantelidis, and Sathya Kamesh Bhethanabhotla. Mind the gap: Promises and pitfalls of hierarchical planning in LeWorldModel. *arXiv preprint arXiv:2607.12547*, 2026. URL https://arxiv.org/abs/2607.12547.

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

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://roboticsproceedings.org/rss19/p026.html.

CLEAR-LeWM contributors. CLEAR-LeWM: fixed start–goal manifests and task-semantic success predicates for LeWM evaluation. Software, version 0.8.0, commit e8292bb0eb796f52220c061c2568ec1984a48011, 2026. URL https://github.com/DavidSunok/CLEAR-LeWM/tree/e8292bb0eb796f52220c061c2568ec1984a48011.

Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005. doi: 10.1007/s10479-005-5724-z. URL https://link.springer.com/article/10.1007/s10479-005-5724-z.

Matthieu Destrade, Oumayma Bounou, Quentin Le Lidec, Jean Ponce, and Yann LeCun. Value-guided action planning with JEPA world models. In *The World Modeling Workshop*, Montreal, QC, Canada, February 2026. URL https://arxiv.org/abs/2601.00844. Poster.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018. URL https://arxiv.org/abs/1812.00568.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32. Curran Associates, Inc., 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/5c48ff18e0a47baaf81d8b8ea51eec92-Abstract.html. arXiv:1906.05253.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2786–2793. IEEE, May 2017. doi: 10.1109/ICRA.2017.7989324.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, pp. 47376–47405, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/cf73d57b6dcda32b293df7c2d5341f49-Abstract-Conference.html.

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933. doi: 10.1037/h0071325. URL https://psycnet.apa.org/doi/10.1037/h0071325.

Jiaming Hu, Yan Zheng, and Tian Wang. SCALE: State-calibrated latent embeddings for JEPA planning in the right geometry. *arXiv preprint arXiv:2608.16287*, 2026. URL https://arxiv.org/abs/2608.16287.

Xiaodi Huang, Ziyi Ding, Jingtian Wan, Yuchen Liu, Yuan Zhang, Xiao-Ping Zhang, Jiayu Chen, Zhang Zhang, and Tao Huang. Traj-LeWM: Path-aware world-model planning via latent trajectory cost. *arXiv preprint arXiv:2608.14125*, 2026. URL https://arxiv.org/abs/2608.14125.

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*, 2026. URL https://arxiv.org/abs/2608.22764.

Yann LeCun. A path towards autonomous machine intelligence. OpenReview preprint, 2022. URL https://openreview.net/forum?id=BZ5a1r-kVsf. Version 0.9.2.

Liangyu Li, Shengzhi Wang, Libin Qiu, Mingliang Xiong, and Qingwen Liu. World model control by trajectory reachability metrics. *arXiv preprint arXiv:2605.22164*, 2026a. URL https://arxiv.org/abs/2605.22164.

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. URL https://arxiv.org/abs/2605.07278.

Zihan Liu, Yuzhe Zhuang, Yuanzu Li, Wanshuang Gou, Jiahong Liu, Min Zhou, and Menglin Yang. ProWorld: Progress-aware hyperbolic world models for long-horizon visual goal reaching. *arXiv preprint arXiv:2608.01926*, 2026. URL https://arxiv.org/abs/2608.01926.

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

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026b. URL https://arxiv.org/abs/2603.19312.

Sergi Masip, Jonathan Swinnen, Yutong Hu, Renaud Detry, and Tinne Tuytelaars. FF-JEPA: Long-horizon planning in world models with latent planners. *arXiv preprint arXiv:2606.09311*, 2026. URL https://arxiv.org/abs/2606.09311.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. URL https://arxiv.org/abs/1802.03426.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ecd92623ac899357312aaa8915853699-Abstract-Conference.html.

Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. doi: 10.1080/14786440109462720. URL https://www.tandfonline.com/doi/abs/10.1080/14786440109462720.

Phu Pham and Aniket Bera. Latent energy action planning with world models. *arXiv preprint arXiv:2609.03294*, 2026. URL https://arxiv.org/abs/2609.03294.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999. doi: 10.1023/A:1010091220143. URL https://link.springer.com/article/10.1023/A:1010091220143.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://github.com/nsavinov/SPTM. arXiv:1803.00653.

Joyjeet Singh. The objective is the bottleneck: Latent world models encode what their planners cannot use. *arXiv preprint arXiv:2608.12959*, 2026. URL https://arxiv.org/abs/2608.12959.

Uladzislau 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 *Advances in Neural Information Processing Systems*, volume 38, pp. 43905–43941. Curran Associates, Inc., 2025. doi: 10.52202/085713-1465. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3e7cf447f21cd11c846463affefce665-Abstract-Conference.html.

Armin Sommer and Jannik Schilling. Reinforced planning with latent world models. *arXiv preprint arXiv:2608.18669*, 2026. URL https://arxiv.org/abs/2608.18669.

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

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018. URL https://arxiv.org/abs/1801.00690.

Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. doi: 10.1126/science.290.5500.2319. URL https://www.science.org/doi/10.1126/science.290.5500.2319.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *Transactions on Machine Learning Research*, May 2026. URL https://openreview.net/forum?id=cHZn5Gdh8e.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL https://www.jmlr.org/papers/v9/vandermaaten08a.html.

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

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

Ge Yang, Amy Zhang, Ari Morcos, Joelle Pineau, Pieter Abbeel, and Roberto Calandra. Plan2Vec: Unsupervised representation learning by latent plans. In Alexandre M. Bayen, Ali Jadbabaie, George Pappas, Pablo A. Parrilo, Benjamin Recht, Claire Tomlin, and Melanie Zeilinger (eds.), *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 935–946. PMLR, 10–11 Jun 2020. URL https://proceedings.mlr.press/v120/yang20b.html.

Lunjun Zhang, Ge Yang, and Bradly C Stadie. World model as a graph: Learning latent landmarks for planning. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12611–12620. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/zhang21x.html.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

A Implementation and experimental details

A.1 NGC and its graph

For each task and pair seed, the graph nodes are N training observations from distinct episodes that the seed never evaluates ($N = 5{,}000$ on Cube, PushT, and Reacher; $1{,}882$ on TwoRoom). Each is a random frame of its episode. Every checkpoint encodes the same observations. We accept the first of up to five fixed sampling seeds whose mutual-15NN graph is connected under every checkpoint, and we never add edges. The three pair seeds are thus three replicates, each with its own pairs and its own graph.

Algorithm 1 Neighborhood Graph Cost (NGC)

Offline, once per encoder

Require: a random subset of N training observations $o_1, \dots, o_N$

$v_i \leftarrow E(o_i)$ for $i = 1, \dots, N$ ▷ graph nodes

$\rho_i \leftarrow$ distance from v_i to its 15th nearest graph node

join i and j by an edge of length $\|v_i - v_j\|_2$ if each is among the 15 nearest neighbors of the other

$D_{ij} \leftarrow$ shortest-path length between i and j for all i, j ▷ Dijkstra

Online, for each candidate scored by CEM

Require: predicted terminal latent q, goal encoding g

$d \leftarrow \|q - g\|_2$; $\rho_q, \rho_g \leftarrow$ distances from q, g to their 15th nearest graph nodes

if $d \leq \min\{\rho_q, \rho_g\}$ **then return** d ▷ Equation 2, first case

end if

return $\min_{i \in \mathcal{N}_4(q),\, j \in \mathcal{N}_4(g)} \left[\|q - v_i\|_2 + D_{ij} + \|v_j - g\|_2\right]$ ▷ Equation 2, second case

A.2 World model and planner

The project-trained world models follow the released LeWM recipe. The encoder is a ViT-tiny (patch size 14, image size 224, width 192) trained from scratch. The predictor is a six-layer transformer (16 heads, MLP width 2048) with a history of three frames. The action embedder has hidden width 768, the two projector heads have hidden width 2048, and SIGReg has weight 0.09 (17 knots, 1024 projections). Training uses AdamW (learning rate 5×10^{-5} with a linear warmup over the first 1% of steps and cosine decay, weight decay 10^{-3}), batch size 128, bf16, gradient clipping at 1.0, and frameskip 5. Cube, PushT, and Reacher train for 10 epochs. TwoRoom runs the 100-epoch schedule and we stop it after epoch 10, where its learning rate is still at 98% of the peak. We use the epoch-10 checkpoint of one training run (the project checkpoint) unless stated otherwise. The released LeWM checkpoints are the HuggingFace `quentinll/lewm-*` weights, pinned by revision and SHA-256.

Planner. CEM uses the same settings on every task. A plan spans $H = 5$ predictor steps of five primitive actions each (25 primitive steps). Each of 30 iterations samples 300 candidate action sequences and refits the sampling distribution to the 30 with the lowest terminal cost. The planner executes all five blocks of the final elite mean before it replans.

A.3 Evaluation protocol

Pair sets. Each task, offset, and pair seed has one fixed pair set of 100 pairs from 100 distinct episodes, identified by its SHA-256 (the manifest files of our code release). All pair sets come from the pair generator of the CLEAR-LeWM software release (v0.8.0, pinned commit; CLEAR-LeWM contributors, 2026) with its filters unchanged. Only the goal offset and the budget ($2\times$ the offset) vary. The generator drops pairs that already satisfy the predicate at the start (38.4% of the Cube candidates at offset 25). TwoRoom episodes end at 101 steps, and only 328 admit an offset of 75, so the three seeds share episodes at that offset. On the pairs of the development seed (Appendix A.4), the median physical start–goal distance rises with the offset on TwoRoom (path length via the door $50 \to 82 \to 108$ px), PushT (block distance $131 \to 188 \to 222$ px), and Reacher (joint distance $0.57 \to 0.80 \to 1.11$ rad). On Cube it barely rises ($0.18 \to 0.17 \to 0.22$ m), because the expert moves the block once per episode, so there a longer offset mostly raises the horizon and the budget.

Execution and success. The simulator starts from the recorded pose of the start observation with zero velocity (*rest start*), because a single start image cannot show velocity. The first plan uses a single-frame context, and later replans use the executed history of three frames at stride 5, the history length of training. Success is the predicate of Table 4. Of the two evaluator repairs of CLEAR-LeWM, evaluation keeps the Reacher angle topology, and the TwoRoom swept-disk collision test only filters candidate pairs. Following its evaluator, we also suppress the dm-control termination and auto-reset between the two inner simulator steps of a Reacher policy step. Every evaluation runs in single precision with TF32 disabled, on one CPU thread, and fixed seeds fix the CEM sampler.

TwoRoom physics. Execution uses the collision rule of the simulator that collected the training data, because the world model was trained on it. The rule tests only the end position of a step and clamps an agent that ends inside a wall back to its face, which admits a small overlap with the door jamb. In the 200 rollouts of Figure 1, all 112 room changes pass through the doorway. The swept-disk test of the pair generator filters candidate pairs and is not used during execution.

Table 4: Success predicates. The episode succeeds at the first step satisfying the predicate.

Task	Predicate
TwoRoom	endpoint within 16 px of the goal position
Reacher	joint error < 0.05 rad, angles compared on the circle
PushT	pusher and T within 20 px of the goal pose and T angle within 20°
Cube	cube center within 4 cm of the goal position

A.4 Statistics and development pairs

We average the three planner seeds of each pair into one success rate and report absolute differences of success rates (%) with a paired bootstrap 95% interval over pairs (50,000 resamples), without multiplicity correction. Different offsets use different pairs and are never paired, and pair seeds that ran as separate studies are never pooled. An interval $[0, 0]$ only means that every pair succeeds under both arms.

Development pairs. We developed NGC on the offset-25 pair set of one pair seed, the *development seed*, with one planner seed. We call this pair set the *development pairs*. The other two pair seeds and the longer offsets were never used to select a method or a hyperparameter. The graph of that development, the *development graph*, excluded only these pairs (on TwoRoom it is the rendered grid of Section 3). Secondary studies that use these pairs, or the pair sets of the development seed at other offsets, say so.

A.5 Comparability with the LeWM evaluation

To compare with published numbers, we also reproduce the random pair draw of the LeWM evaluation exactly. Each planner seed draws 50 starts with the goal 25 steps ahead, and we use a single-frame context and the native success rule. The drawn episodes are excluded from the graph. Table 5 places our L2 cost next to the numbers reported by Maes et al. (2026b). With the released checkpoint, it lies 2 to 8% below them. The gap is within one standard deviation over pair draws on TwoRoom, Reacher, and Cube, and larger on PushT (88.0 against 96%). The re-evaluation of LeWM by Li et al. (2026b) reaches 90.4% on PushT and stays within 4% of our released-checkpoint L2 column on every task, so the PushT gap is not specific to our evaluator. With the project checkpoint, the L2 cost lies within 4% of the reported numbers on every task. NGC does not depend on the evaluator repairs of our protocol. It keeps the TwoRoom gain ($+10.0$ and $+14.7$% with the project and released checkpoints, both intervals excluding zero) and leaves Reacher and PushT within 2% (all four intervals contain zero). On Cube it gains less than on the fixed pair sets ($+6.7$% $[0.0, +13.3]$ and $+9.3$% $[+3.3, +15.3]$, against $+10.6$ and $+11.1$%), because the random draw keeps pairs that already satisfy the native success rule at the start (38% of sampled Cube pairs; Sun et al., 2026). These count as successes for every cost.

Table 5: The LeWM evaluation. Success (%) is the mean ± SD over three planner seeds, each with its own draw of 50 pairs. The last column is the success rate that Maes et al. (2026b) report for LeWM in their Figure 6, rounded to integers there. Their CEM runs 10 iterations on TwoRoom, Reacher, and Cube and 30 on PushT, whereas we run 30 on every task.

	project checkpoint		released LeWM checkpoint		
Task	L2	NGC	L2	NGC	reported
TwoRoom	90.0 ± 2.0	100.0 ± 0.0	85.3 ± 3.1	100.0 ± 0.0	87
Reacher	82.7 ± 6.1	82.7 ± 7.6	81.3 ± 6.1	82.7 ± 4.2	86
PushT	94.0 ± 4.0	92.0 ± 3.5	88.0 ± 5.3	86.7 ± 3.1	96
Cube	70.7 ± 10.1	77.3 ± 8.3	68.7 ± 7.0	78.0 ± 7.2	74

B GEOMETRY DIAGNOSTICS

B.1 TWOROOM COST MAPS AND PROJECTIONS

Cost maps. The cost maps evaluate $C(E(\mathrm{render}(x, y)), E(o_{\mathrm{goal}}))$ with the project checkpoint on a 46×46 grid with 4 px stride, restricted to the $1{,}882$ positions where the collision rule of Appendix A.3 can place the agent center. Each pixel takes the cost of its nearest grid position (at most 2 px away). Each panel is scaled to its own 2nd–98th percentile, so colors are comparable only within a panel. Dark gray is the wall, and light gray marks positions the dynamics never reach. The curve joins the recorded positions of the executed rollout, one per planner step. Both rows come from one evaluation of the development pairs, in which the L2 cost solves 88 pairs and NGC all 100. The four columns are three of the 12 L2-cost failures and one success. In the three failures, the L2 cost ends in the cheapest 5, 6, and 22% of the map, 100, 108, and 70 px from the goal, twice in a low-cost band along a far corner and once in a pocket beside the doorway.

Projections. Every embedding of Figure 3 is fit once on the $1{,}882$ grid latents, stored, and plotted without refitting. PCA is not whitened. t-SNE is an openTSNE 1.0.2 fit (perplexity 50, PCA initialization, fixed random seed), and new points are mapped with its out-of-sample transform (perplexity 50, 25 neighbors, 250 iterations). Isomap is scikit-learn 1.9.0 with $k = 8$, an undirected union graph, and classical MDS. Metric MDS minimizes the raw stress $\sum_{i<j}(\|y_i - y_j\| - \|z_i - z_j\|)^2$ by SMACOF from the classical MDS solution (scikit-learn 1.9.0, relative tolerance 10^{-9}, Kruskal stress-1 0.376). Three random starts reach higher stress (0.377–0.381) and fold the room in the same way. UMAP is umap-learn 0.5.12 with 15 neighbors, minimum distance 0.1, and a fixed random seed.

B.2 COST AGAINST PHYSICAL DISTANCE ON FOUR TASKS

Figure 6 uses one graph per task over observations with recorded simulator state, built with the rule of Section 4. The nodes are the TwoRoom grid, $5{,}000$ training observations (one per episode) on Reacher and PushT, and a coverage-balanced sample of $5{,}000$ on Cube, whose mutual graph has four components; we keep the $4{,}996$-node main one. These are not the graphs used in planning, whose nodes we sampled without their simulator state. The physical distance is the path length via the doorway on TwoRoom, the joint-space distance on Reacher, the gripper-to-cube plus cube-to-goal distance on Cube (zero once the cube lies within the 4 cm success radius), and the pusher-to-block plus block-to-goal distance on PushT. We draw 300 goals from the graph nodes and score every node against every goal. Because the queries are graph nodes, NGC equals the graph distance except in the first case of Equation 2 (Spearman between the two above 0.999). Correlations over all ordered pairs share points and are descriptive. The median per-goal Spearman of the L2 cost and NGC is 0.19 and 0.97 on TwoRoom, 0.21 and 0.99 on Reacher, 0.15 and 0.82 on Cube, and 0.19 and 0.53 on PushT. NGC correlates best where the physical distance is the exact shortest feasible path length (TwoRoom, Reacher) and worst on PushT, whose proxy ignores the rotation of the block, so the Cube and PushT panels likely understate how well NGC orders states.

B.3 DISTANCE CONCENTRATION

On the four graphs of Figure 6, the latents projected on $1{,}024$ random unit directions have a median standard deviation of 0.99 to 1.06 and a median excess kurtosis of -0.06 to -0.14, close to the standard normal that SIGReg targets. The covariance Σ is not isotropic. Its participation ratio $(\operatorname{tr}\Sigma)^2/\operatorname{tr}(\Sigma^2)$ is 33 on TwoRoom, 63 on Reacher, 58 on Cube, and 94 on PushT, out of 192. Two independent samples lie $\sqrt{2\operatorname{tr}\Sigma}$ apart in expectation, 19.4 to 21.0 here. From the second to the last decile of physical distance, the median physical distance grows by a factor of 5.3, 3.5, 3.2, and 2.8 on the four tasks, while the median Euclidean distance changes by 3.4, 4.9, 1.1, and 1.9% and stays between 19.3 and 21.7. Above the median physical distance, the coefficient of variation is 0.04 to 0.07 for the Euclidean distance and 0.21 to 0.25 for the physical distance, and the median Euclidean distance (19.6 to 21.5) lies within 3% of $\sqrt{2\operatorname{tr}\Sigma}$. Physically far pairs are thus as far apart as independent samples. This uses one checkpoint and no SIGReg ablation, so it shows the concentration but not its cause.

C Latent planning performance across goal distances

Table 6 gives the cells of Figure 5. The three replicates agree in sign in all twelve cells and are within a few points of each other. The development-seed replicate is the most favorable one in four cells and the least favorable in three. On TwoRoom and Reacher, no pair at any offset is solved by the L2 cost and lost by NGC.

Gain by distance. The bottom row of Figure 5 is a post hoc stratification of the per-pair records by the physical start–goal distance of Appendix B.2, computed from the recorded start and goal states (Cube 0.04–0.78 m, TwoRoom 17–214 px, PushT 19–639 px, Reacher 0.06–3.30 rad), with quintile edges per task. The farthest-bin gains are $+18.5$ $[+13.7, +23.7]$ on Cube, $+86.3$ $[+81.5, +90.7]$ on TwoRoom, $+13.5$ $[+8.5, +18.5]$ on PushT, and $+44.8$ $[+38.1, +51.5]$ on Reacher. Table 7 bins within each offset, where the budget is fixed. The farthest PushT tercile at offset 25 is the only bin of either binning in which NGC loses ($[-7.0, -1.3]$).

Table 6: Success rate (%) of the L2 cost and NGC at the offsets of Figure 5 (project checkpoint, 300 pairs per offset, three planner seeds averaged within each pair). Δ is NGC minus L2 with a paired bootstrap 95% interval over pairs; gained/lost counts the pairs whose mean success rose/fell.

Task	Offset	Budget	L2	NGC	Δ [95% CI]	gained/lost
Cube	25	50	47.2	59.7	$+12.4$ [$+8.4$, $+16.6$]	97/42
Cube	50	100	40.6	50.8	$+10.2$ [$+6.7$, $+13.9$]	70/28
Cube	100	200	31.1	50.0	$+18.9$ [$+14.8$, $+23.1$]	105/21
TwoRoom	25	50	91.0	100.0	$+9.0$ [$+6.1$, $+12.1$]	37/0
TwoRoom	50	100	57.0	100.0	$+43.0$ [$+37.7$, $+48.4$]	143/0
TwoRoom	75	150	29.6	99.9	$+70.3$ [$+65.3$, $+75.2$]	223/0
PushT	25	50	93.1	92.7	-0.4 [-1.9, $+1.0$]	11/15
PushT	50	100	44.9	65.6	$+20.7$ [$+16.2$, $+25.1$]	123/25
PushT	100	200	12.7	26.3	$+13.7$ [$+10.0$, $+17.4$]	84/18
Reacher	25	50	99.1	99.7	$+0.6$ [-0.1, $+1.3$]	8/3
Reacher	50	100	94.2	100.0	$+5.8$ [$+3.3$, $+8.4$]	20/0
Reacher	100	200	79.1	100.0	$+20.9$ [$+16.6$, $+25.4$]	72/0

Table 7: NGC minus the L2 cost (%) within each offset, by tercile of the physical start–goal distance (near / middle / far).

Task (offsets)	shortest offset	middle offset	longest offset
Cube (25/50/100)	$+1.3$ / $+10.0$ / $+26.0$	$+5.3$ / $+15.0$ / $+10.3$	$+7.7$ / $+27.3$ / $+21.7$
TwoRoom (25/50/75)	0.0 / $+4.0$ / $+23.0$	$+8.3$ / $+40.0$ / $+80.7$	$+46.7$ / $+74.0$ / $+90.3$
PushT (25/50/100)	$+1.0$ / $+1.7$ / -4.0	$+15.0$ / $+21.0$ / $+26.0$	$+14.0$ / $+15.0$ / $+12.0$
Reacher (25/50/100)	0.0 / $+0.3$ / $+1.3$	0.0 / 0.0 / $+17.3$	0.0 / $+2.7$ / $+60.0$

D COMPARISON WITH TRAIN-TIME ALTERNATIVES

RC-aux (Li et al., 2026b) adds multi-horizon prediction and a reachability head to the LeWM objective. At planning time it multiplies the L2 cost by $\max(1 - \lambda R, 0.05)$, where R is the predicted probability that the goal can be reached within the remaining horizon. **INTACT** (Sun et al., 2026) replaces the search with an actor that maps the current latent, the goal displacement, and the previous action block to the next action block. Table 8 compares both with NGC, pair for pair, under the pairs, predicate, and budget rule of Figure 5.

Setup. INTACT uses the task-specific checkpoints of its release (trained with the same seed as ours, the actor on eight-frame windows) through its interface, with one planner seed because the actor is deterministic. The released RC-aux weights briefly continue the released LeWM checkpoint, so comparing them with our checkpoint would mix the method with the checkpoint. We instead continued our project checkpoint with the unchanged RC-aux objective for $2{,}000$ steps per task, with the data, split, and optimizer of the checkpoint's own training, a 20-step warmup, and a cosine schedule (22–27 minutes on one A100). Its horizon curriculum is scheduled in epochs and never advances in a run shorter than one epoch, so we disabled it and sampled every horizon from the first step. We plan with the RC-aux gate at the weights of the RC-aux paper (0 on Cube, 0.35 elsewhere) and our CEM budget. We did not run the continuation without the auxiliary objective, so RC-aux minus L2 includes the effect of $2{,}000$ more training steps, and the PushT loss of RC-aux cannot be split between its training and its gate. Each method is one training seed.

Results. RC-aux gains a few points over the L2 cost on Cube ($+3.8$ to $+5.6\%$) and at the long Reacher offsets ($+4.8$ and $+10.7$), changes nothing on TwoRoom, and loses on PushT (-2.8 to -11.6). It trails NGC on every task at every offset, with intervals excluding zero except on Reacher at offset 25. INTACT leads both costs on Cube. Its lead lies on pairs whose goal holds the cube while the start does not, where it succeeds on 48–100% across the three offsets against 8–10% for the L2 cost and 7–18% for NGC. The actor has learned the grasp, which CEM rarely finds under either cost. Four times as many candidates add only 7 points to NGC on Cube at offset 100 (Appendix H.2), so CEM more likely lacks a cost that rewards the grasp than more samples. NGC's own Cube gains come from pairs that hold the cube at both ends (77 against 56% at offset 25, 75 against 39% at offset 100) and from pairs that release it, where at offset 50 it also leads INTACT (81 against 70%). On the other tasks INTACT trails NGC, on TwoRoom by 2–13 points (growing with the offset) and on PushT by 18 points at offsets 50 and 100, and ties on Reacher. Where distances along the manifold order the states that the task needs, NGC beats both a retrained model and a learned actor. Where the cost does not yet reward a step such as the grasp, an actor that has learned the step wins.

Table 8: Two train-time methods against NGC on the pairs of Table 6. L2, NGC, and RC-aux use the project checkpoint; RC-aux is the RC-aux objective continued from it for $2{,}000$ steps and planned with its gate (three planner seeds). INTACT uses its released checkpoints and search-free actor (one planner seed). The two right columns give the method minus NGC with paired bootstrap 95% intervals over pairs.

Task	Offset	L2	NGC	RC-aux	INTACT	RC-aux − NGC	INTACT − NGC
Cube	25	47.2	59.7	52.8	100.0	-6.9 [-10.8, -3.0]	$+40.3$ [$+35.8$, $+45.0$]
Cube	50	40.6	50.8	44.4	63.3	-6.3 [-10.1, -2.6]	$+12.6$ [$+6.0$, $+19.0$]
Cube	100	31.1	50.0	34.9	73.0	-15.1 [-19.4, -10.9]	$+23.0$ [$+18.9$, $+27.2$]
TwoRoom	25	91.0	100.0	92.8	98.3	-7.2 [-10.1, -4.6]	-1.7 [-3.3, -0.3]
TwoRoom	50	57.0	100.0	56.8	92.0	-43.2 [-48.8, -37.8]	-8.0 [-11.3, -5.0]
TwoRoom	75	29.6	99.9	30.6	87.3	-69.3 [-74.2, -64.2]	-12.6 [-16.4, -8.9]
PushT	25	93.1	92.7	85.9	89.0	-6.8 [-10.2, -3.3]	-3.7 [-8.0, $+0.6$]
PushT	50	44.9	65.6	33.3	47.3	-32.2 [-37.1, -27.3]	-18.2 [-24.6, -11.9]
PushT	100	12.7	26.3	9.9	8.0	-16.4 [-20.6, -12.4]	-18.3 [-23.0, -13.8]
Reacher	25	99.1	99.7	99.6	99.3	-0.1 [-0.7, $+0.4$]	-0.3 [-1.4, $+0.6$]
Reacher	50	94.2	100.0	99.0	98.3	-1.0 [-2.1, -0.1]	-1.7 [-3.3, -0.3]
Reacher	100	79.1	100.0	89.8	99.0	-10.2 [-13.7, -7.0]	-1.0 [-2.3, $+0.0$]

E OTHER CHECKPOINTS AND REPRESENTATIONS

E.1 THE RELEASED LEWM CHECKPOINT AND OTHER TRAINING SEEDS

Table 9 repeats Figure 5 with the checkpoint released by Maes et al. (2026b) in place of ours. The pairs, planner seeds, predicate, and budget rule are the same, and each pair seed has one graph re-encoded by that checkpoint. The offset-25 L2 cells are reused from an earlier evaluation of the same pairs and checkpoint. NGC gains on Cube by about as much as on ours, and on TwoRoom at every offset and on PushT and Reacher where the L2 cost collapses, with every one of these intervals excluding zero. At offset 25 PushT and Reacher stay within 1%. How much NGC gains at long range does depend on the checkpoint. On TwoRoom and PushT the long-offset gains are two fifths to two thirds of those on ours, and NGC reaches 79.1 and 53.7% at TwoRoom offsets 50 and 75, where it solves every pair on ours.

Training seeds. Two more training seeds of our recipe, trained and evaluated on a separate machine, gain +11.3 and +11.9% on Cube and +10.4 and +9.1 on TwoRoom on the offset-25 pairs of the three seeds with one graph per task, where the project checkpoint gains +10.6 and +9.0 under the same setup.

Table 9: The released LeWM checkpoint on the pairs of Figure 5 (300 pairs per offset, three planner seeds averaged within each pair). Δ is NGC minus L2 with a paired bootstrap 95% interval over pairs. The last column repeats Δ of Table 6 for the project checkpoint.

Task	Offset	L2	NGC	Δ [95% CI]	Δ, ours
Cube	25	47.1	60.3	+13.2 [+9.2, +17.3]	+12.4
Cube	50	44.4	55.8	+11.3 [+7.8, +15.1]	+10.2
Cube	100	32.7	50.7	+18.0 [+14.0, +22.1]	+18.9
TwoRoom	25	85.4	99.3	+13.9 [+10.6, +17.4]	+9.0
TwoRoom	50	50.0	79.1	+29.1 [+23.0, +35.2]	+43.0
TwoRoom	75	26.2	53.7	+27.4 [+20.9, +33.8]	+70.3
PushT	25	86.6	86.4	−0.1 [−1.9, +1.7]	−0.4
PushT	50	39.6	53.7	+14.1 [+10.0, +18.3]	+20.7
PushT	100	13.1	18.8	+5.7 [+2.6, +8.9]	+13.7
Reacher	25	98.8	99.6	+0.8 [+0.0, +1.7]	+0.6
Reacher	50	96.6	100.0	+3.4 [+1.7, +5.6]	+5.8
Reacher	100	81.8	100.0	+18.2 [+14.1, +22.6]	+20.9

E.2 A SECOND WORLD MODEL

The hypothesis predicts a gain only where the L2 cost loses its order at long range and the latent neighborhoods are faithful, since chaining unfaithful neighborhoods adds error. Neither condition is a property of the task, so we test a second world model with our architecture, data, split, and seed but a different latent geometry. LpWM (Kuang et al., 2026) applies a rectified link (ReLU forward, GELU gradient backward) to the projector and predictor outputs, and replaces SIGReg with a sliced Wasserstein distance to a rectified Laplace product distribution (RDMReg). Its latent is non-negative, with a terminal zero fraction of 0.42 (Reacher) to 0.57 (TwoRoom). We train one LpWM per task in the Deep-AdaLN setting of the LpWM paper (ten epochs, batch 128, AdamW 5×10^{-5}, RDMReg weight 0.5 with 1,024 projections) and freeze it. NGC is applied unchanged to the LpWM latents of the graph observations of Figure 5. The LpWM graphs of all Cube and PushT seeds split into a main component of 4,992 to 4,999 nodes and a few outliers. We joined the components by their shortest inter-component edges (one to three per graph) instead of resampling, which leaves every other edge and path unchanged.

Table 10 shows that the LpWM L2 cost is better than ours on TwoRoom and Reacher and worse on Cube and PushT. NGC on LpWM gains where that L2 cost still collapses with the horizon, at TwoRoom offsets 50 and 75 (122 pairs gained and one lost) and Reacher offset 100, and there reaches the ceiling of NGC on our model. On Cube and PushT it changes success by at most +2.2%. The three pair seeds agree in sign in every cell.

Offline geometry on the same graphs. Table 11 correlates, on the graph nodes of each pair seed, the Euclidean and the graph distance of both encoders with the physical distance (path length via the door on TwoRoom, cube position on Cube, wrapped joint angle on Reacher, T-block position on PushT), and measures how faithful their neighborhoods are. Globally the LpWM Euclidean distance orders pairs better on every task, and after the graph the two encoders are close. The neighborhoods differ. On TwoRoom and Reacher most of a node's 15 latent neighbors are among its 15 physically nearest nodes under either encoder, but on Cube and PushT far fewer are, and there the LpWM latent neighbors of a node lie 2.7 and 5.7 times farther in state space than its physical neighbors. The gain of NGC thus follows how faithful a representation's neighborhoods are. The study uses one training seed and our choice of regularizer weight (the LpWM release uses 10 with a temporal Jaccard term for Cube), its PushT distance ignores the agent, and the bridging edges touch a few outlier nodes.

Table 10: LpWM on the pairs of Figure 5 (300 pairs per offset). Δ is NGC on the LpWM latent minus the LpWM L2 cost with a paired bootstrap 95% interval; bold intervals exclude zero. L2 and NGC on our checkpoint are repeated for reference.

Task	offset	L2	NGC	LpWM L2	LpWM + NGC	Δ (%)	95% CI
Cube	25	47.2	59.7	39.8	41.4	$+1.7$	$[-1.6, +4.9]$
	50	40.6	50.8	35.8	36.6	$+0.8$	$[-1.9, +3.4]$
	100	31.1	50.0	31.4	33.1	$+1.7$	$[-0.8, +4.1]$
TwoRoom	25	91.0	100.0	98.3	100.0	$\mathbf{+1.7}$	$[\mathbf{+0.4}, \mathbf{+3.1}]$
	50	57.0	100.0	89.4	100.0	$\mathbf{+10.6}$	$[\mathbf{+7.3}, \mathbf{+14.0}]$
	75	29.6	99.9	64.2	99.0	$\mathbf{+34.8}$	$[\mathbf{+29.8}, \mathbf{+39.9}]$
PushT	25	93.1	92.7	78.7	80.9	$\mathbf{+2.2}$	$[\mathbf{+0.1}, \mathbf{+4.4}]$
	50	44.9	65.6	27.0	29.1	$+2.1$	$[-0.2, +4.4]$
	100	12.7	26.3	8.0	8.0	$+0.0$	$[-1.6, +1.6]$
Reacher	25	99.1	99.7	98.9	99.3	$+0.4$	$[-0.3, +1.2]$
	50	94.2	100.0	99.3	100.0	$+0.7$	$[0.0, +1.7]$
	100	79.1	100.0	93.0	100.0	$\mathbf{+7.0}$	$[\mathbf{+4.4}, \mathbf{+9.8}]$

Table 11: Offline geometry of the LpWM latent and ours on the graph nodes of Figure 5 (medians over the three graphs). Spearman correlation of the Euclidean distance and of the graph distance with the physical distance over all node pairs, and the fraction of a node's 15 latent neighbors among its 15 physically nearest nodes.

	Euclidean distance		graph distance		neighbors in place	
Task	LpWM	ours	LpWM	ours	LpWM	ours
Cube	0.35	-0.03	0.81	0.87	0.32	0.46
TwoRoom	0.63	0.48	0.90	0.92	0.79	0.84
PushT	0.33	0.09	0.59	0.60	0.14	0.24
Reacher	0.36	0.16	0.71	0.84	0.81	0.93

F Ablations

F.1 Which part of NGC carries the gain

Table 12 extends Table 1 to every offset. CEM selects elites by rank, so any strictly monotone transform of a cost leaves the planner unchanged, and the Euclidean distance gives the cells of the L2 cost. NGC w/o path stays within 2% of the L2 cost on Cube and PushT at every offset and gains at most $+11.9\%$ on TwoRoom and $+2.2\%$ on Reacher. Adding the path gives the rest wherever NGC gains, with every interval excluding zero except at offset 25 on PushT and Reacher. The first case of Equation 2 is rare in the first CEM pool (3.5% of the initial Cube candidates on 16 diagnostic pairs).

Why the rescaled L2 cost collapses on TwoRoom. In the first CEM pool of the offset-25 evaluation on each TwoRoom graph, the rescaled cost ranks candidates more by their own ρ_q than by their distance to the goal (rank correlation -0.73 to -0.87 with ρ_q, -0.33 to -0.35 with the Euclidean

distance). A graph over training observations is dense along frequently visited routes and sparse elsewhere, and dividing by ρ_q makes the sparse regions cheap. With the uniform rendered grid of Section 3 as the graph, on a separate set of 100 offset-25 pairs with three planner seeds, the same control had recovered most of the gain (97.3% against 86.0% for the L2 cost and 100% for NGC). A local-scale correction is only as reliable as the uniformity of the graph nodes. NGC avoids this failure because ρ enters it only through the condition of the first case of Equation 2.

Table 12: Which part of NGC carries the gain, on the pairs and graphs of Table 6. The rescaled L2 cost is $\|q-g\|_2/\sqrt{\rho_q \rho_g}$. NGC w/o path replaces D_{ij} with the Euclidean distance between the same graph nodes. Δ in % with a paired bootstrap 95% interval over pairs.

Task	offset	L2	rescaled L2	NGC w/o path	NGC	NGC w/o path − L2	NGC − NGC w/o path
Cube	25	47.2	46.0	45.4	59.7	-1.8 [$-4.6, +1.0$]	$+14.2$ [$+10.3, +18.2$]
	50	40.6	43.8	40.8	50.8	$+0.2$ [$-2.2, +2.7$]	$+10.0$ [$+6.6, +13.6$]
	100	31.1	33.1	30.2	50.0	-0.9 [$-3.7, +1.9$]	$+19.8$ [$+15.3, +24.2$]
TwoRoom	25	91.0	11.6	94.4	100.0	$+3.4$ [$+1.2, +5.8$]	$+5.6$ [$+3.3, +8.0$]
	50	57.0	2.9	68.9	100.0	$+11.9$ [$+7.8, +16.1$]	$+31.1$ [$+26.1, +36.1$]
	75	29.6	1.1	38.7	99.9	$+9.1$ [$+5.4, +12.9$]	$+61.2$ [$+56.0, +66.3$]
PushT	25	93.1	89.8	92.1	92.7	-1.0 [$-2.9, +0.9$]	$+0.6$ [$-1.1, +2.2$]
	50	44.9	35.2	45.4	65.6	$+0.6$ [$-3.6, +4.7$]	$+20.1$ [$+15.2, +25.1$]
	100	12.7	9.3	11.3	26.3	-1.3 [$-3.6, +1.0$]	$+15.0$ [$+11.2, +18.8$]
Reacher	25	99.1	99.3	99.3	99.7	$+0.2$ [$-0.6, +1.0$]	$+0.3$ [$-0.2, +0.9$]
	50	94.2	94.6	96.4	100.0	$+2.2$ [$+0.8, +3.9$]	$+3.6$ [$+1.8, +5.6$]
	100	79.1	78.4	81.0	100.0	$+1.9$ [$+0.4, +3.4$]	$+19.0$ [$+14.9, +23.3$]

F.2 Graph construction

Node sample and neighbor rule. On the development pairs and the development graph (one planner seed), we replaced the mutual rule by a *directed* rule (each node to its 15 nearest) or a *union* rule (an edge if either node lists the other), and on Cube the random nodes by a pose-balanced sample. On Cube these variants change success by -6 to 0 points, with every paired 95% interval containing zero, and all stay above the L2 cost. On TwoRoom every variant, and also three graphs over training observations in place of the rendered grid, solves every pair. Offline, the union graph destroys the cross-room order on TwoRoom (whole-graph Spearman with physical distance 0.08), yet success is unchanged. CEM ranks candidates imagined from one state, so it uses the local order around the current state and the goal, not the order over the whole graph. The offline diagnostics of Figure 6 are therefore not sufficient for control, and the planning result does not require the mutual rule. We keep it because it keeps edges local, as the manifold interpretation needs. On the TwoRoom grid the longest mutual-15 edge spans 32 px, while a union-15 edge can span 180 px.

Graph size and neighborhood size. Table 13 varies both around the graph of the development seed (Cube 5,000 nodes, TwoRoom 1,882, $k = 15$), on that seed's pair sets at offset 25 and at the longest offset with three planner seeds. Smaller graphs are nested subsets from one permutation of the nodes, accepted only if all three subsets give a connected graph (the first permutation on Cube, the fifth on TwoRoom). k sets both the graph and ρ. Where the mutual-kNN graph is disconnected ($k = 5$ on both tasks and $k = 10$ on Cube), NGC is undefined, and joining the components by their shortest edges stays within the interval of the reference. At offset 25, every connected k from 15 to 60 and every Cube subset down to one eighth stay within the interval of the full graph; on TwoRoom only the two smallest subsets lose (16 and 7%). At the longest offset, one eighth of the nodes loses 14% on Cube and 65% on TwoRoom while half stays within 5%, and on TwoRoom $k = 30$ and $k = 60$ lose 47 and 59%. This matches the known failure of Isomap when neighborhoods are large relative to the curvature and edges join points that are far apart along the manifold (Tenenbaum et al., 2000), although we have not verified which edges the larger graphs add.

Table 13: Graph size and neighborhood size on the pair sets of the development seed (100 pairs × 3 planner seeds per cell). Each cell gives success (%) and, in small type, Δ against the reference (full graph, $k = 15$) with a paired bootstrap 95% interval. Fractions are nested node subsets (Cube 625/1,250/2,500, TwoRoom 235/470/941). Mutual-5NN graphs are disconnected on both tasks.

Arm	Cube, offset 25	Cube, offset 100	TwoRoom, offset 25	TwoRoom, offset 75
L2	48.0	26.0	88.0	29.7
Reference (full, $k = 15$)	59.7	50.0	100.0	100.0
$\frac{1}{8}$ of the nodes	57.3 $-2.3\,[-8.3, +3.3]$	36.0 $-14.0\,[-21.7, -6.7]$	84.0 $-16.0\,[-23.0, -9.3]$	35.0 $-65.0\,[-73.7, -56.0]$
$\frac{1}{4}$ of the nodes	59.7 $0.0\,[-6.0, +6.0]$	46.7 $-3.3\,[-8.3, +1.7]$	93.0 $-7.0\,[-12.0, -2.7]$	47.7 $-52.3\,[-61.3, -43.3]$
$\frac{1}{2}$ of the nodes	59.3 $-0.3\,[-6.0, +5.3]$	54.7 $+4.7\,[+0.3, +9.3]$	99.7 $-0.3\,[-1.0, 0.0]$	96.7 $-3.3\,[-6.0, -1.3]$
$k = 10$	disconnected	disconnected	100.0 $0.0\,[0.0, 0.0]$	99.3 $-0.7\,[-1.7, 0.0]$
$k = 30$	61.7 $+2.0\,[-2.7, +7.0]$	48.7 $-1.3\,[-5.7, +3.0]$	100.0 $0.0\,[0.0, 0.0]$	52.7 $-47.3\,[-56.7, -38.0]$
$k = 60$	58.3 $-1.3\,[-6.7, +4.0]$	46.0 $-4.0\,[-10.0, +1.7]$	98.3 $-1.7\,[-4.0, 0.0]$	41.3 $-58.7\,[-67.3, -49.7]$

G Oracle predictor

The oracle predictor replaces the learned predictor inside CEM and nothing else. For each candidate in each CEM iteration, it copies the full simulator state of the executed episode into a scratch simulator, executes the candidate, renders the terminal frame, and encodes it with the frozen encoder. On Cube the copy is the full MuJoCo data structure. On PushT, the physics engine keeps internal solver state that setting the simulator state does not restore, so we rebuild the scratch simulator by replaying every executed action from reset, and a run aborts if the rebuilt state differs from the live one. The oracle runs on the pair sets of the development seed, with the graph of that replicate and one planner seed. The learned columns of Table 2 are the cells of the same pairs and planner seed.

Table 14 gives the change of each cost and of the gain from the learned to the oracle predictor. It adds PushT offset 25, where both costs solve nearly every pair with either predictor (93 and 93 learned, 98 and 97 oracle). In every other row the oracle predictor helps NGC more than the L2 cost, but with one planner seed the interval of the change of the gain reaches or covers zero in every row.

Table 14: Oracle minus learned predictor (100 pairs, one planner seed). Gain is $(\text{NGC} - \text{L2})_{\text{oracle}} - (\text{NGC} - \text{L2})_{\text{learned}}$. Differences in % with a paired bootstrap 95% interval over pairs.

Task	Offset	L2	NGC	Gain
Cube	25	+17 [+8, +26]	+28 [+19, +37]	+11 [0, +22]
Cube	50	+15 [+7, +23]	+16 [+7, +25]	+1 [−9, +11]
Cube	100	+11 [+4, +19]	+21 [+11, +31]	+10 [0, +21]
PushT	25	+5 [−1, +11]	+4 [0, +9]	−1 [−8, +5]
PushT	50	+6 [−3, +15]	+11 [+2, +20]	+5 [−6, +16]
PushT	100	+2 [−2, +6]	+9 [−1, +19]	+7 [−3, +17]

H Computation and planning time

H.1 Implementation and planning time

Implementation. The implementation keeps the node latents and the all-pairs table D (100 MB for $N = 5{,}000$) on the planning device. The nearest graph nodes of the goal are fixed for an episode, so it precomputes $\phi_i(g) = \min_{j \in \mathcal{N}_4(g)}[D_{ij} + \|v_j - g\|_2]$ for every graph node once per goal. Scoring a candidate then needs one nearest-neighbor query and $\min_{i \in \mathcal{N}_4(q)}[\|q - v_i\|_2 + \phi_i(g)]$. It computes exact pairwise distances rather than the matrix-product expansion, so an exact goal match keeps its zero cost. The development-seed replicate of Table 6 was scored by an earlier CPU implementation,

which the device implementation reproduces: on the candidate pools of Table 3, its double-precision scores equal the CPU scores and its single-precision scores differ by at most 2×10^{-5} with identical top-30 elites, and in planning both precisions reproduce the CPU success list on all 400 development-seed pairs.

Timing. Timings come from one RTX 4090 with the arms of the four tasks interleaved. On this card double-precision throughput is $1/64$ of single precision, and the $300 \times N$ distance matrix makes f64 three times as costly as f32. Evaluation-time ratios also depend on the number of replans. Where NGC solves more pairs it ends more episodes early (146 against 154 replans on Cube, 101 against 121 on TwoRoom, 105 against 113 on Reacher). On PushT both costs solve 93 pairs, and the f64 evaluation-time ratio (1.35) matches the replan ratio (1.40).

H.2 Computation sweep

Figure 7 multiplies the number of CEM candidates and elites by 0.2 to 4 ($60/6$ to $1{,}200/120$), keeping the elite fraction and the 30 iterations, at the longest offset of each task on the pairs, planner seeds, and graphs of Table 6. The $1\times$ cells are those of Table 6, and every other cell is paired with them. From $1\times$ to $4\times$, the L2 cost moves by $+4.3\%$ $[+2.1, +6.6]$ on Cube, $+2.0$ $[+0.4, +3.7]$ on TwoRoom, $+0.1$ $[-1.4, +1.7]$ on PushT, and -1.1 $[-2.6, +0.2]$ on Reacher. At $1.2\times$, which covers the computation that NGC adds to a replan in single precision, it moves by $+2.6$, $+0.8$, -0.2, and $+0.6\%$. NGC at $1\times$ leads the L2 cost at $4\times$ by $+14.6\%$ $[+10.2, +19.0]$, $+68.3$ $[+63.2, +73.3]$, $+13.6$ $[+9.7, +17.6]$, and $+22.0$ $[+17.6, +26.7]$, and even at $0.2\times$ it leads the L2 cost at $4\times$ on TwoRoom, PushT, and Reacher. NGC itself gains from more candidates only below its ceiling ($+7.3\%$ $[+4.8, +10.0]$ on Cube and $+2.9$ $[+0.1, +5.7]$ on PushT at $4\times$; -15.3 and -6.6 at $0.2\times$). Below $1.2\times$, the wall-clock time of an evaluation barely changes, because the candidates of an iteration are rolled out as one GPU batch.

I Limitations of the evidence

Evaluation pairs come from training episodes, as in the LeWM evaluation and in every LeWM follow-up we know of. The graph excludes those episodes, but the encoder has seen them. We have not tested out-of-distribution goals, a new TwoRoom map, or a graph that does not cover the evaluation distribution.

The comparison across offsets (Table 6) and the ablation (Table 12) use all 300 pairs per offset, and the computation sweep (Figure 7) uses them at the longest offset. The other studies use fewer. The oracle predictor (Appendix G) uses the pair sets of the development seed with one planner seed. The graph and neighborhood sizes (Table 13) use the same pair sets with three planner seeds, on Cube and TwoRoom only. The graph variants (Appendix F.2) and the timings (Appendix H.1) use offset-25 pairs only. At the longest offset, Table 13 finds NGC more sensitive to the graph size and the neighborhood size than at offset 25, so the graph variants do not show that NGC is insensitive to the node sample and the neighbor rule at the longer offsets. Evaluations ran on two GPU models, and the two arms of a cell were not paired by GPU model.