Long-Horizon Planning with Latent World Models via Graph Anchors

Syed Nazmus Sakib[1], **Abdul Monaf Chowdhury**[1], **Peizhen Li**[3], **Nafiul Haque**[1], **Chandak Chakma**[1], **Md Mehedi Hasan**[1], **Zehong Cao**[3,*], **and Shifat E. Arman**[1,2,*]

[1]Department of Robotics and Mechatronics Engineering, University of Dhaka
[2]Department of Computer Science, University of Oxford
[3]School of CSIT, Adelaide University

Project Page Code

Abstract

Latent world models plan in a compact latent space by predicting the outcomes of actions. Their planners judge a plan by the latent distance between its predicted end state and the goal image. We show that this distance fails in two distinct ways. First, it loses directional signal a few steps from the goal, causing distant goals to fail. Second, it weighs the robot arm over the manipulated object, causing manipulation to fail at contact. We measure both failures directly on recorded data by asking whether a lower latent distance corresponds to true progress toward the goal. Across five environments, six visual representations (including raw pixels and V-JEPA 2), and nine Meta-World tasks, this ranking test cleanly separates the two failures, and the graph distance to a goal predicts where closed-loop planning breaks down. Both failures share a single remedy: replacing the distant goal with a nearby target that lies on a recorded trajectory. We realize this principle with *Graph Anchors*, a training-free planner for frozen latent world models. It builds a sparse graph once over the model's own training frames and aims each planning step at a node a few moves ahead along the shortest path to the goal, keeping paths on recorded transitions wherever contact matters. On goals up to 400 steps away, Graph Anchors raise LeWM's success from 13% to 85% on PointMaze-Medium, from 8% to 87% on PointMaze-Large, and from 5% to 71% on Push-T, where stronger search toward the goal fails and where prior training-free planners break down under missing step horizons, subsampling shortcuts, or fragmented contact.

Keywords: Latent World Models, Long-Horizon Planning, Model-Predictive Control, Experience Graphs, Visual Control

1. Introduction

World models predict how actions change the world, allowing an agent to evaluate candidate plans in imagination before acting [14–17]. Recent visual world models predict in the latent space of an image encoder [25, 31, 38] and plan by sampling action sequences, rolling out their predicted futures, and keeping the actions whose final latent state lies closest to a goal image [7, 35]. This approach works reliably for nearby goals. On PointMaze-Medium, LeWorldModel (LeWM) [25] solves 50% of goals 50 steps away, yet drops to just 13% at 200 steps, as illustrated in Figure 1. On Push-T it solves only 5% at 150 steps, often parking the pusher while leaving the T-block behind.

Why does a planner that easily solves short-range goals break down here? Rather than guessing, we test the latent goal cost directly. On recorded data, before running any planner, we measure how often a lower latent distance corresponds to a move that genuinely progresses toward the goal (Section 4.1). This simple test isolates two root failures. The first is *range*: latent distance ranks local moves accurately right next to the goal, but loses directional signal within a few graph hops across every navigation

*Corresponding authors: Jimmy.Cao@adelaide.edu.au, shifat.arman@eng.ox.ac.uk.

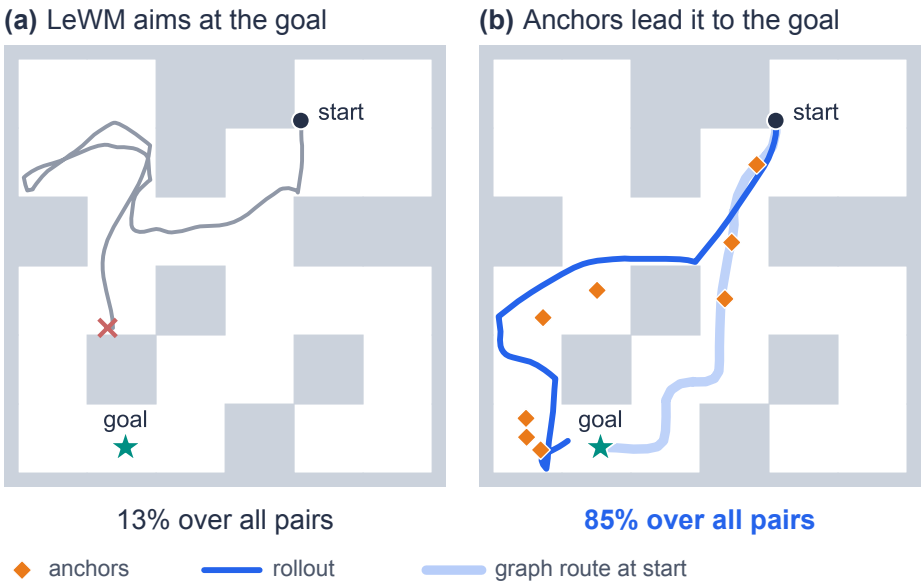

Figure 1: **A distant goal misleads the planner, while near targets on a recorded route guide it.** **(a)** LeWM aims every plan straight at a goal 200 steps away and loses its way in the maze. **(b)** Graph Anchors aim each plan at a nearby node (orange) along the shortest route through the recorded data (light blue), and the exact same frozen planner reaches the goal.

task and across six visual encoders, including raw pixels, DINOv2, CLIP, SigLIP, V-JEPA 2, and LeWM (Section A.1). The second is *focus*: whole-image distance weighs the robot arm far more heavily than the manipulated object, preferring an accurate arm pose over completing the task across all nine Meta-World [42] object tasks we measure (Section A.2).

Importantly, both failures have the exact same remedy. Whole-image distance works reliably whenever its target is nearby and lies along a route recorded in the data. On Meta-World, keeping the released cost untouched and simply aiming at a demonstration frame 20 steps ahead solves 65 of 72 object episodes that the final goal image never solves. We introduce *Graph Anchors* to provide such targets automatically without demonstrations or retraining. We construct a sparse graph once over the world model's own training frames, compute a shortest path to the goal, and aim each planning step at an anchor node a few moves ahead on that path. Where object contact matters, penalizing cross-episode jumps keeps the route on continuous recorded transitions. With the released search and world model untouched, replacing only the distant goal with our nearby anchor lifts PointMaze-Medium at 200 steps from 12% to 85%, whereas stronger search aimed at the distant goal stalls at 9%. On Push-T, preserving recorded transitions raises $d{=}150$ success from 30% to 71%.

Our main contributions are threefold:

- **A direct diagnosis of the latent goal cost.** A local ranking test on recorded data separates two distinct failures, *range* and *focus*, across five environments, six image representations, and nine Meta-World tasks. Offline graph distance predicts where closed-loop planning fails, and a formal drift analysis (Section A.3) connects local ranking to linear versus quadratic reaching times.
- **Graph Anchors.** A simple, training-free planner for frozen latent world models that cures both failures using nearby anchors on recorded routes, changing only the target and scoring rule of the

planner while leaving the encoder, predictor, and CEM optimizer untouched.

- **Long-horizon evaluation and failure analysis of alternative planners.** Across goals up to 400 steps away on navigation and manipulation, we benchmark against world-model planners, offline goal-conditioned policies, graph-guided policies (GAS [4], TTGS [30]), and training-free world-model planners (AP [23], NGC [41], Metro-WM [20]), showing how each missing diagnostic axis causes specific structural failures (Section 5.5, Section D).

2. Related Work

Latent world models and their goal costs. World models predict the consequences of actions in a learned latent space [14–16] and support planning via model-predictive control [7, 17]. For visual goal reaching, DINO-WM [45] plans with CEM over predicted DINOv2 [31] patch features, PLDM [38] learns latent dynamics from reward-free offline trajectories, and LeWM [25] trains encoder and predictor jointly from pixels with a joint-embedding objective [2]. JEPA-WM [40] studies the design choices behind such planners, and V-JEPA 2 [3] extends them to robot actions. These planners minimize latent distance to a goal image and are typically evaluated around 25 steps ahead. Recent work improves the predictor [8, 13] or questions the goal cost directly: Temporal-Distance JEPA [5] and TEMPO [28] learn temporal distances, Li *et al.* [21] learn reachability metrics, Singh [37] shows that LeWM's latent distance misrepresents progress on Two-Room, and Alrasheed *et al.* [1] find that world models rank actions reliably only near their imagined horizon. We measure directional ranking directly against graph distance and against scene elements, and keep the world model and cost completely unchanged.

Subgoals for latent planners. Earlier visual planners reached goal images with pixel-space models [11] or planned over intermediate subgoals proposed by learned generative models [29, 34]. HWM [44] trains world models at multiple temporal scales, FF-JEPA [26] predicts intermediate subgoals, SAGE [6] trains a subgoal generator and action proposal model for LeWM, and PILOT [24] generates visual subgoals with a video model. All of these require training new components. Graph Anchors instead retrieve subgoals directly from the world model's existing training data, requiring no additional training whenever the data covers the route.

Graphs of experience. SPTM [36] links observations by temporal adjacency and learned reachability, SoRB [10] plans over stored states with a goal-conditioned value function, HTM [22] and L3P [43] search graphs of generated images and learned landmarks, and sparse graphical memory [9] and landmark graphs [18] maintain compact state graphs for goal reaching. More recently, GAS [4] builds a graph in a learned temporal-distance space for a low-level policy, and TTGS [30] searches an offline dataset over learned value distances to guide a frozen policy. Three other training-free planners retarget a frozen LeWM directly: Anchored Planning (AP) [23] retrieves the single-episode segment best matching the current frame and goal given the step count d, NGC [41] replaces latent distance with shortest-path length over a subsampled undirected graph, and Metro-WM [20] retrieves subgoals from a sparse graph of recorded frames. We compare all six paradigms directly under our diagnostic framework in Section 5.5 and Section D.

3. Preliminaries

Problem setting. We study visual goal-conditioned planning from offline, reward-free trajectories [25, 33, 38]. We are given a fixed dataset of image observations and actions, with no rewards and no extra environment interaction for training. Following LeWM [25], each action is a block of five consecutive

environment steps. At test time, the agent observes o_1 and receives a goal image o_g, succeeding if it reaches the goal within its step budget.

Latent world model. A latent world model [25] predicts future states inside a compact embedding space using an encoder f and a dynamics predictor p:

$$z_t = f(o_t), \qquad \hat{z}_{t+1} = p(\hat{z}_t, a_t), \tag{1}$$

where z_t embeds observation o_t and $\hat{z}_{t+1}$ predicts the embedding after executing action block a_t. Throughout our experiments we keep the released LeWM encoder and predictor completely frozen.

Latent planning. LeWM [25] rolls out candidate sequences of H action blocks from $\hat{z}_1 = z_1$ and scores each plan by the squared latent distance between its final predicted state and the goal embedding $z_g = f(o_g)$:

$$a^*_{1:H} = \arg\min_{a_{1:H}} \|\hat{z}_{H+1} - z_g\|_2^2, \qquad H = 5. \tag{2}$$

The planner minimizes Equation (2) using the cross-entropy method (CEM) [35], iteratively sampling action sequences from a Gaussian, scoring them, and refitting the Gaussian to the lowest-cost elites. The agent executes the chosen plan and replans from its new observation. Consequently, closed-loop planning moves toward the goal only if lower latent distance actually reflects true progress.

4. Graph Anchors

We first diagnose why latent planning breaks down at long range and during object contact (Section 4.1). We then present three lightweight components that address both failures without modifying or retraining the world model (Figure 3): a sparse graph of recorded frames supplying nearby anchors on a shortest path (Sections 4.2 and 4.3), an episode-switch cost that keeps routes on continuous recorded transitions (Section 4.4), and closest-approach scoring that guides rollouts smoothly through each anchor (Section 4.5).

4.1. Where Latent Planning Fails

To check whether lower latent distance truly indicates progress, we test the goal cost directly on recorded data using the graph of Section 4.2. For a goal node g, a state v located h hops from g, and two neighbours u, u' of v where u is genuinely closer to g than u', we define the directional ranking accuracy:

$$\rho(h) = \Pr\left[\|z_u - z_g\|_2^2 < \|z_{u'} - z_g\|_2^2\right]. \tag{3}$$

A score of ρ=1 means the cost always points downhill toward the goal, whereas $\rho=\frac{1}{2}$ carries no directional information and behaves like a coin flip. Two concrete cases illustrate what happens in closed loop:

Example 1 (A far goal). On PointMaze-Medium, a goal 200 steps ahead lies 34 graph hops away. Six hops from a goal, squared latent distance orders local moves correctly only 0.59 of the time. In closed loop, just 48% of LeWM's decisions end closer to the goal. After 15 decisions the agent is still 28 hops away, and only 13% of episodes succeed.

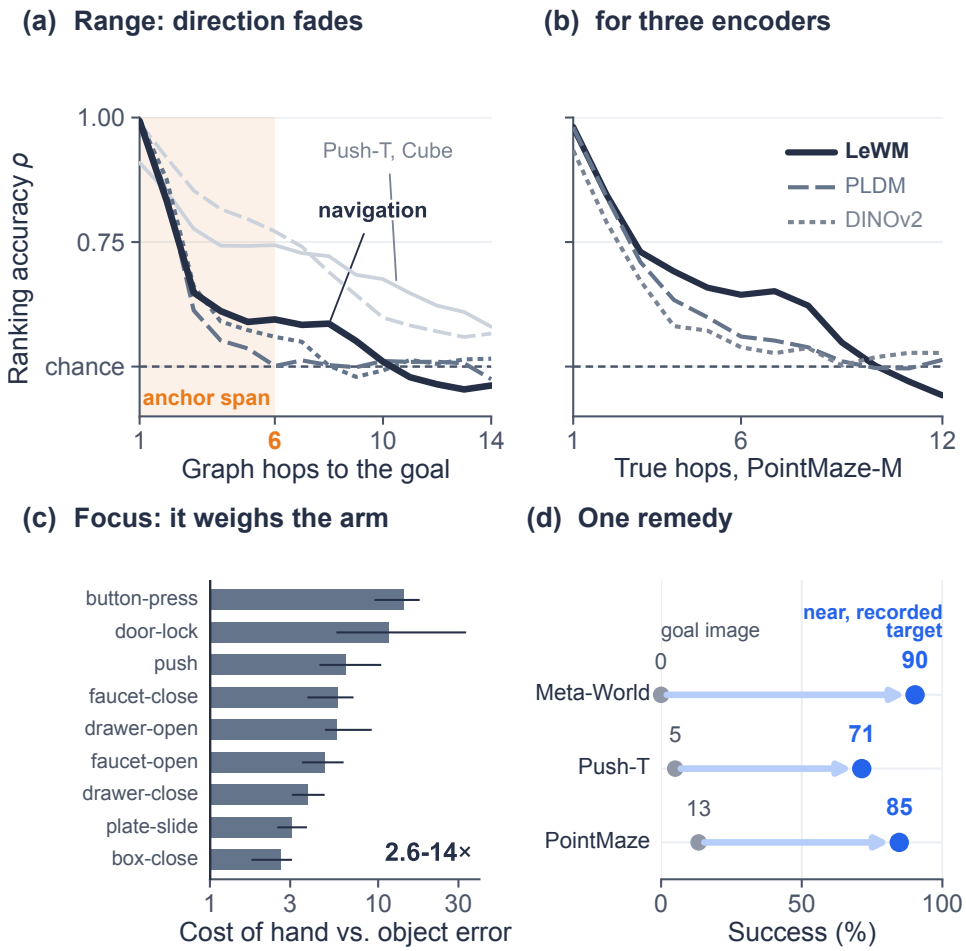

Figure 2: **The diagnosis in four steps.** **(a)** Directional ranking accuracy $\rho(h)$ of the LeWM goal cost against graph distance to the goal across 300 goals per environment, with navigation tasks in bold/dashed/dotted and manipulation tasks faded; shaded band marks our anchor span. **(b)** Ranking accuracy on PointMaze-Medium for three encoders measured on a true-position reference graph over 200 goals (extended to six encoders including raw pixels and V-JEPA 2 in Figure A1). **(c)** JEPA-WM on Meta-World: cost of moving only the hand by 5 cm divided by leaving only the object off target by its success threshold (median and IQR across 30 placements per task; extended to five encoders in Figure A6). **(d)** Closed-loop success aiming at the final goal versus aiming at a nearby target on a recorded route with the exact same cost.

Example 2 (Pushing a block). On Push-T, the T-block must reach a goal pose 150 steps ahead. Overall local ranking still looks healthy at 0.91 one hop away and 0.74 six hops away. Yet latent distance orders the block's true object progress correctly only 0.54 of the time, compared to 0.68 for graph distance along recorded routes. LeWM solves just 5% of episodes.

The planner of Section 3 fails in these settings for three coupled reasons.

First, whole-image distance has a short *range*. Across every navigation benchmark, $\rho(h)$ starts near 1

right next to the goal and drops to 0.50–0.59 within six hops (Figure 2a). Section A.1 verifies the exact same collapse directly along recorded time steps without building any graph. Measuring hops on a true-position reference graph, LeWM, PLDM, and frozen DINOv2 all start at 0.93–0.98 and fall below 0.55 within 6–9 hops on PointMaze-Medium (Figure 2b). In Section A.1 we further test raw pixels, CLIP, SigLIP, and video-trained V-JEPA 2 alongside 10-epoch training checkpoints (Figures A1 and A2). Because even raw pixels decay at the exact same rate as V-JEPA 2, short range is an inherent property of comparing two whole images across corridors and walls rather than an undertrained encoder. Far from the goal, heavier search cannot rescue the planner because candidate rollouts are ranked by a directionless score: doubling the horizon leaves Example 1 at 12%. Section 4.3 cures this by replacing the distant goal with a nearby target.

Second, whole-image distance has the wrong *focus*. In Example 2, overall ρ decays slowly, so range alone does not explain why Push-T fails. Because whole-image distance averages over all pixels, a large robot arm or agent dominates a small manipulated object. Testing the released JEPA-WM planner [40] on Meta-World [42] by placing arm and object independently with inverse kinematics shows that shifting only the hand by 5 cm costs 2.6 to 14$\times$ more than leaving only the object off target across all nine object tasks (Figure 2c), a bias that persists across raw pixels, DINOv2, CLIP, SigLIP, and V-JEPA 2 (Figure A6), even though a linear probe on the exact same frozen features locates the object cleanly (Section A.2). Under exact simulator dynamics, the released cost solves 0 of 72 object episodes. Merely picking a closer target is not enough either: on Push-T, an anchor three hops ahead without preserving recorded transitions stays at 30%. However, aiming the unchanged released cost at a nearby frame *on a recorded route*, such as the frame 20 steps ahead on the episode's own demonstration, immediately solves 65 of 72 Meta-World episodes (Figure 2d). Matching a nearby recorded arm pose naturally moves the object as recorded. Section 4.4 preserves recorded transitions automatically inside our graph.

Third, scoring only the final state of a plan rewards stopping at the target rather than moving past it. A nearby anchor is a stepping stone to pass through, not a resting point. Section 4.5 replaces terminal-step scoring with closest-approach scoring.

4.2. A Graph of Recorded Experience

We construct a sparse graph $G = (V, E)$ once from the world model's own training trajectories. Each node i represents a recorded frame sampled every five steps (one action block), storing its episode index e_i, timestamp t_i, and frozen latent embedding $z_i = f(o_i)$. Directed *temporal edges* connect consecutive frames within an episode, while bidirectional *neighbour edges* connect close embeddings across episodes:

$$\begin{aligned} E_T &= \{(i,j) : e_i = e_j,\ t_j = t_i + 5\}, \\ E_N &= \{(i,j),(j,i) : j \in \mathrm{NN}_{10}(i),\ \|z_i - z_j\|_2^2 < \tau\}. \end{aligned} \tag{4}$$

Temporal edges record real physical transitions that occurred in the dataset. Neighbour edges rely on Euclidean latent distance strictly at one-hop scale, where ρ is near 1, allowing routes to stitch across episodes without tuning any environment-specific distance threshold: τ is set automatically to the empirical median squared distance of a single recorded action block. Test episodes are always removed before evaluation so no anchor can replay a test trajectory. Building the full graph takes 51 s once on PointMaze-Medium (Section B.1).

Why does chaining local latent neighbours cure long-range failure? Frames that are close in latent space are physically close in the environment: on PointMaze-Medium and Two-Room, virtually none (0%) of a frame's ten nearest latent neighbours sits more than one maze cell away by walking distance (Figure 4a), and checking all 994,834 unique neighbour edges on PointMaze-Medium yields zero wall

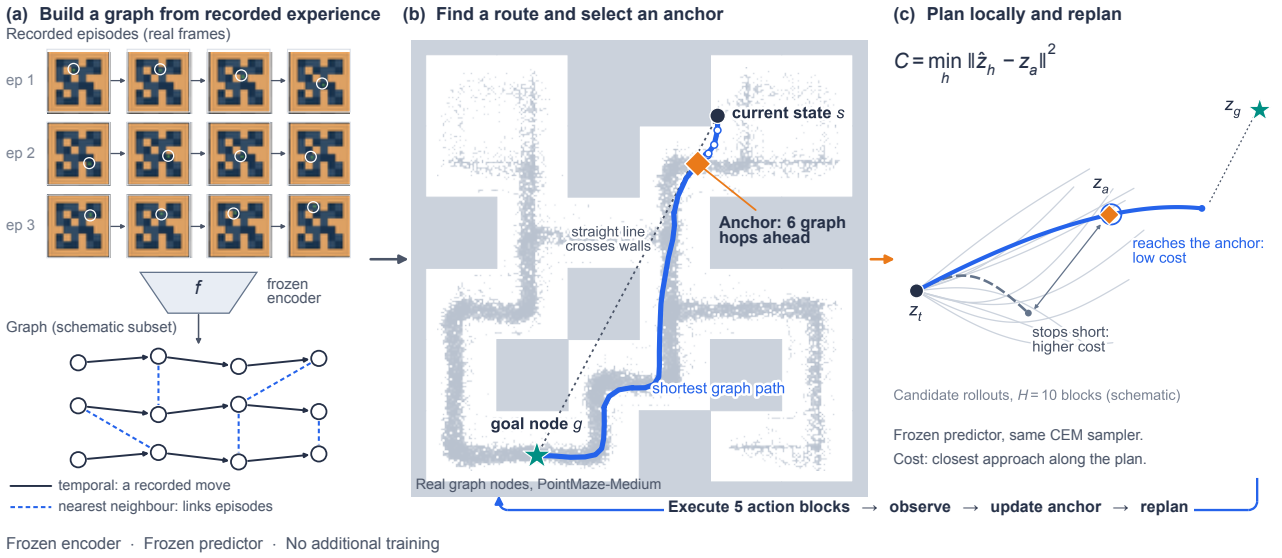

Figure 3: **Graph Anchors turn one distant target into a sequence of nearby ones. (a)** Encoded training frames are linked by directed temporal edges and local nearest-neighbour edges. **(b)** At each replanning step, the active anchor lies a few moves ahead along the shortest graph path to the goal. **(c)** CEM scores rollouts by their closest approach to the anchor, executes, and re-anchors from the new observation.

crossings (Section B.2). Over long distances, direct latent distance breaks down: rank correlation with true walking distance on PointMaze-Medium drops from 0.97 near the goal to 0.17 far from it (Figure 4b). Shortest-path hops over the local neighbour graph hold correlation at 0.69 far from the goal, and raise Two-Room from 0.05 to 0.50. The graph therefore handles global routing while the visual cost only needs to rank the last few hops to a nearby anchor.

4.3. Choosing the Anchor

At test time, we map the current embedding z_t and goal embedding z_g to their nearest graph nodes s and g, find a shortest path $\pi(s,g)$, and select the node K moves ahead as the active target:

$$z_a = z_{\pi_K(s,g)} \quad \text{if } K < d_G(s,g) < \infty, \qquad \text{else } z_g, \tag{5}$$

where d_G denotes shortest-path length. When the goal is already within K moves or unreachable in the graph, the planner aims directly at z_g. Running one reversed Dijkstra search from g caches the shortest path from every node in the graph, so each replanning step reduces to a single fast lookup (Algorithm A1 in Section B.1).

Moving closer to this anchor transfers progress directly to the distant goal. Because a lies on a shortest path from x to g, we have $d_G(x,g) = d_G(x,a) + d_G(a,g)$. For any reached state y closer to a than x, the directed triangle inequality immediately gives

$$d_G(y,g) \le d_G(y,a) + d_G(a,g) < d_G(x,g). \tag{6}$$

Section A.3 formalizes this transfer under metric distortion (Lemma 1 and Corollary 1) and proves that sustained local drift converts quadratic random-walk hitting times into linear $O(D)$ progress (Proposition 1). The span K balances directional reliability against step progress: six hops from a goal, ρ is 0.59 on PointMaze-Medium versus 0.51 at ten hops. We use K=6 as our default navigation span (Section 5.4).

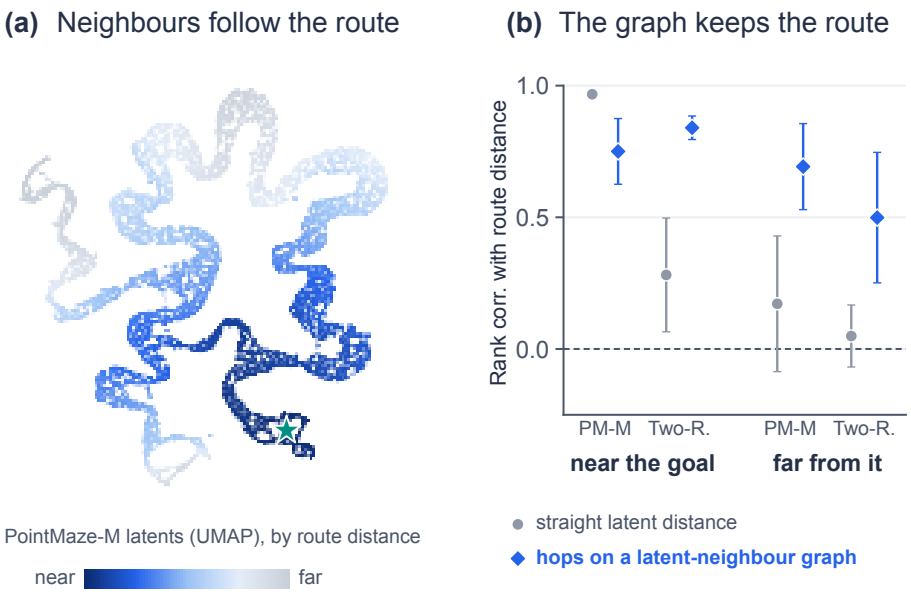

Figure 4: **Why a graph over latents works. (a)** UMAP [27] visualization of 30k PointMaze-Medium latents coloured by true route distance to one goal (star), showing smooth local corridor structure. **(b)** Rank correlation with true walking distance for direct latent distance versus shortest-path graph hops, near the goal (under 15% of maximum route distance) and far from it (over 50%), averaged over 20 random goals.

4.4. Keeping the Route on Recorded Transitions

In contact manipulation (Example 2), a nearby anchor alone is insufficient. On Push-T, unweighted shortest paths follow a recorded temporal transition on only 45% of hops, and only 29% carry recorded block motion; typical cross-episode jumps shift the T-block by 22 px with no recorded push connecting them. To keep paths on continuous recorded pushes, we charge a higher edge weight $\lambda \geq 1$ on cross-episode neighbour edges:

$$w(i,j) = \begin{cases} 1, & (i,j) \in E_T, \\ \lambda, & (i,j) \in E_N \setminus E_T, \end{cases} \tag{7}$$

and select the anchor after K *temporal* moves along this weighted shortest path so cross-episode jumps do not count as progress. Along temporal edges, arm and object move together as recorded, reproducing the Meta-World demonstration effect (Figure 2d) purely from offline training data. Because Equation (6) holds for any directed weighted graph with $w \geq 1$, progress transfer remains intact while cross-episode stitching still occurs whenever a jump saves more than $\lambda - 1$ moves. We use two task-level settings: *navigation* (K=6, λ=1) and *contact* (K=2, λ=3).

4.5. Planning Through the Anchor

When an anchor lies six hops ahead on the way to a much farther goal, the planner should move smoothly *through* the anchor rather than braking at it. CEM therefore scores each rolled-out trajectory $\hat{z}_{2:H+1}$ by its closest predicted approach to z_a:

$$C(\hat{z}_{2:H+1}, z_a) = \min_{2 \leq h \leq H+1} \|\hat{z}_h - z_a\|_2^2. \tag{8}$$

Algorithm 1 Graph Anchors

Input: Frozen world model (f, p), training frames, goal image o_g, span K, switch cost λ, horizon H

Build sparse graph G from training frames (Equation (4)) with edge weights w (Equation (7))
$z_g \leftarrow f(o_g)$; $g \leftarrow$ nearest node; compute reversed Dijkstra tree from g on G
for each replanning step **do**
    $z_t \leftarrow f(o_t)$; $s \leftarrow$ nearest node
    $z_a \leftarrow$ node K temporal moves along $\pi(s, g)$, or z_g (Equation (5))
    Run unchanged CEM over $a_{1:H}$ with closest-approach cost $\min_h \|\hat{z}_h - z_a\|_2^2$ (Equation (9))
    Execute plan (navigation: 5 blocks; contact: up to closest approach)

After executing, the agent observes the new frame, updates its anchor, and replans. Terminal-step scoring wastes action blocks after passing a waypoint and penalizes rollouts that pass cleanly through it; taking the minimum over the plan removes both effects. In navigation we use H=10 blocks and execute 5; in the contact setting we use H=2 blocks and execute up to the block of closest approach. CEM preserves LeWM's exact hyperparameters (300 samples, 30 elites, 30 iterations).

4.6. Overall Objective

Combining Equations (5), (7) and (8), Graph Anchors replace Equation (2) with

$$a^*_{1:H} = \arg\min_{a_{1:H}} \min_{2 \le h \le H+1} \|\hat{z}_h - z_a\|_2^2. \tag{9}$$

Only two elements change relative to LeWM: the target shifts from the distant goal to a nearby anchor on a recorded route, and rollout cost shifts from terminal distance to closest approach. The encoder, predictor, and CEM optimizer remain frozen. Algorithm 1 summarizes the loop.

5. Experiments

We evaluate across six visual control benchmarks: Push-T, Two-Room, OGBench-Cube [33], and Reacher [39] using the released LeWM checkpoints and datasets [25], PointMaze-Medium using D4RL [12] expert data and the public LeWM checkpoint of [19], and PointMaze-Large using OGBench navigation trajectories with LeWM trained via its default script (Section B.2). Start and goal frames come from the same held-out episode d steps apart with step budget $2d$; *stitched* goals draw start and goal from different held-out episodes (budget 400 on mazes, 200 on Two-Room; both episodes removed from the graph). Each seed fixes 50 identical start and goal pairs across all methods, evaluated over three seeds unless noted, with paired McNemar significance tests reported in Section C.1. *LeWM* denotes the released planner and *LeWM + min-over-plan* applies our closest-approach search directly to the distant goal, isolating the target change alone. We also compare against PLDM [38], DINO-WM [45], offline policies GCBC, GCIVL, GCIQL [33], and HILP [32] (trained on identical data under a matched short budget detailed in Section D.1), graph-guided policies GAS [4] and TTGS [30], and training-free world-model planners AP [23], NGC [41], and Metro-WM [20].

5.1. Does Graph Anchors Reach Distant Goals?

Table 1 and Figure 6 show both the range collapse and its removal. As goals move farther away, LeWM loses 37 to 83 percentage points across Push-T, Two-Room, and PointMaze-Medium, and LeWM + min-over-plan collapses similarly on both mazes. Graph Anchors hold 85–91% across d=50 to 200 on PointMaze-Medium and 93–96% on Two-Room, while lifting PointMaze-Large at d=400 from 8% to 87%. In Figure 5, LeWM stalls against the dividing wall on Two-Room and abandons the block on Push-T, whereas anchors guide the agent cleanly through the doorway and push the block toward

Table 1: **Success (%) on distant goals.** All methods evaluate identical start and goal pairs under budget $2d$ (400 for stitched maze goals). Entries report mean ± std over 50-pair seeds (usually three seeds), with best in **red** and second underlined; †marks one evaluation. Graph Anchors use the navigation setting (K=6, λ=1) on the first row and the contact setting (K=2, λ=3) on the second row (PointMaze-Large contact ran on the retrained checkpoint of Table A7). Full distance sweeps appear in Table A6.

Method	Push-T 150	Two-Room 75	PM-M 200	PM-M stitch.	PM-L 400	PM-L stitch.	Reacher 150
LeWM [25]	5 ±4	31 ±3	13 ±2	39 ±10	8 ±0	5 ±1	60 ±6
LeWM + min-over-plan	9 ±1	62 ±10	12 ±5	41 ±4	26 ±6	17 ±1	**77 ±5**
PLDM [38]	0 ±0	50 ±6	3 ±3	28 ±4	1 ±1	2 ±0	15 ±1
GCIVL [33]	0 ±0	81 ±2	0†	16 ±0	1 ±1	3 ±1	0 ±1
GCIQL [33]	0 ±0	37 ±4	1 ±1	12 ±3	3 ±1	3 ±1	0 ±0
HILP [32]	0 ±0	87 ±3	1 ±1	18 ±0	9 ±1	10 ±6	0 ±0
Graph Anchors (ours)	30 ±3	93 ±1	85 ±6	**95 ±4**	**87 ±1**	**66 ±6**	68 ±4
contact setting (K=2, λ=3)	**71 ±3**	**100 ±0**	**94 ±5**	91 ±4	13 ±6	9 ±5	54 ±11

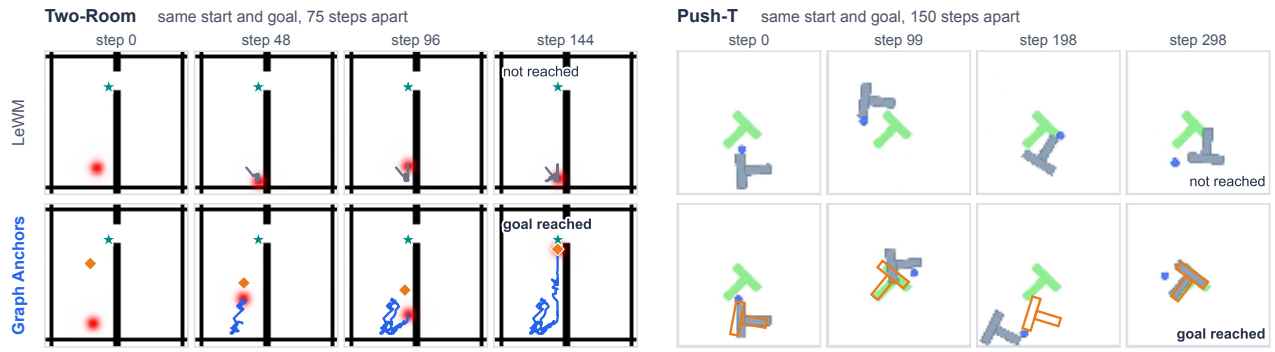

Figure 5: **What the planner sees.** Camera frames on identical start and goal pairs for LeWM (top) and Graph Anchors in the navigation setting (bottom, seed 0), showing the longest solved path missed by LeWM. Two-Room overlays the path so far, goal (star), and active anchor (diamond); Push-T overlays the active anchor block outline and green goal pose. Full videos appear on our project page.

its target. Across every seed at $d \geq 50$, Graph Anchors outperform both LeWM baselines ($p < 10^{-2}$ on Two-Room and both mazes in Section C.1) and stitch cross-episode goals cleanly, solving 95% of stitched PointMaze-Medium goals versus 39% for LeWM.

Offline policies excel on compact environments such as Two-Room, where GCBC reaches 100% and HILP reaches 87% at d=75, yet collapse on larger mazes: GCBC falls to 51% on PointMaze-Medium d=200 and 18% on PointMaze-Large d=400. Simply adding a graph does not rescue them. Even when fed waypoints from our verified latent graph or paired with GAS [4] and TTGS [30] search, offline policies reach at most 21% on PointMaze-Medium d=200 and 17% on PointMaze-Large d=400 (Section D.1 diagnoses both value saturation and policy execution failure in detail). DINO-WM's high default PointMaze-Large score (100%) relies on ground-truth agent coordinates inside its goal cost; using images only drops it to 70% versus 87% for image-only Graph Anchors (Table A12).

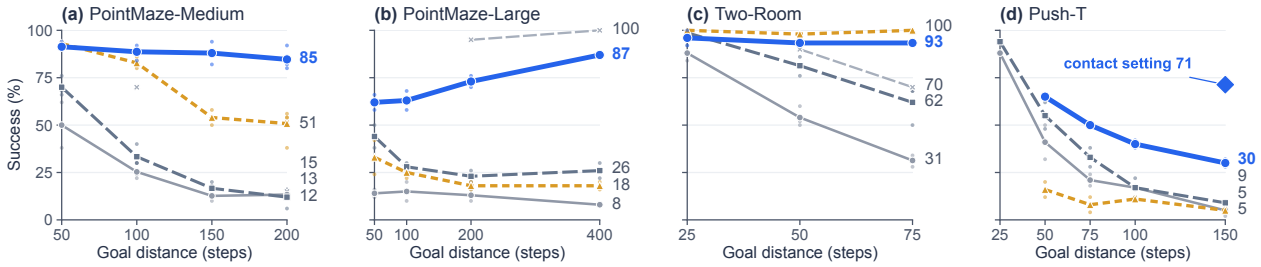

Figure 6: **Near targets eliminate the collapse with goal distance.** Blue: Graph Anchors (navigation setting; ♦ contact setting on Push-T over three seeds). Amber dashed: GCBC, the strongest offline policy. Dark gray dashed: LeWM + min-over-plan. Gray: LeWM. Light gray dashed: DINO-WM (20 pairs, seed 0; its default cost also reads proprioceptive coordinates, ablated in Table A12). Lines show seed means and dots show single 50-pair runs. PointMaze-Large rises with d as the $2d$ budget expands.

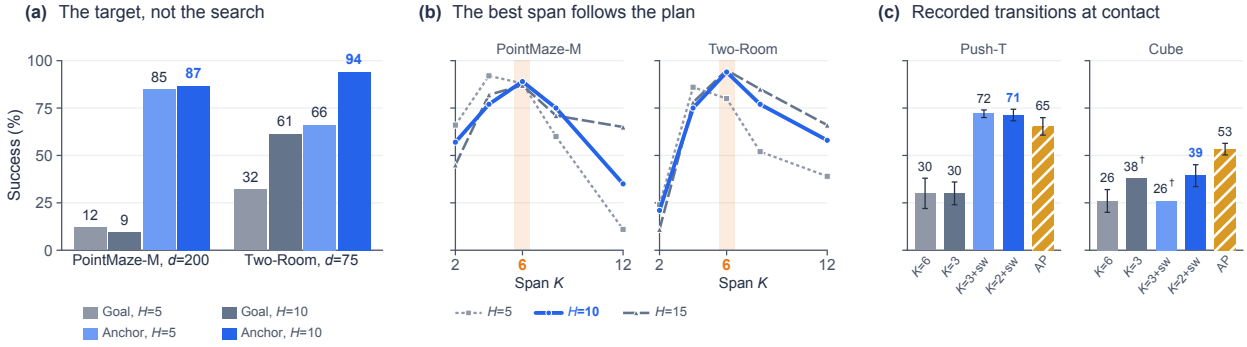

Figure 7: **Why each component is needed. (a)** 2×2 factorial crossing target choice and search rule across three seeds. **(b)** Anchor span K crossed with planning horizon H (mean of seeds 0 and 42; shaded: K=6). **(c)** Contact tasks at d=150 varying anchor span and switch cost (λ=3) against AP (hatched) on identical machine and pairs across three seeds (†seed 0 only).

Graph Anchors also transfer to other world models whenever their local dynamics are accurate: on Two-Room, adding anchors lifts PLDM from 50% to 97% and DINO-WM (over frozen DINOv2 latents) from 70% to 95%, whereas on PointMaze-Medium PLDM (4%) and DINO-WM (20%) cannot reliably reach targets six hops away. Compute overhead is minimal: building the 199k-node PointMaze-Medium graph takes 51 s once, one cached CPU Dijkstra search takes 84–93 ms, and exact per-decision GPU overhead is just 64.6 ms (Table A17b).

5.2. Does It Help at Contact?

Push-T exposes focus failure most sharply, and the contact setting (K=2, λ=3) makes a dramatic difference: at d=150 it reaches 71%, compared to 30% for the navigation setting and 65% for AP [23] on identical pairs (Figure 7c). On Cube, gains are more modest (26% to 39%, remaining below AP's 53%) because 15 of 40 navigation failures never initiate contact with the cube; a recorded trajectory alone cannot synthesize a delicate grasp. On medium mazes, the contact setting remains strong at 94% on PointMaze-Medium and 100% on Two-Room, trading off slight efficiency on stitched routes (91% versus 95%) and short-range Reacher (54% versus 68%).

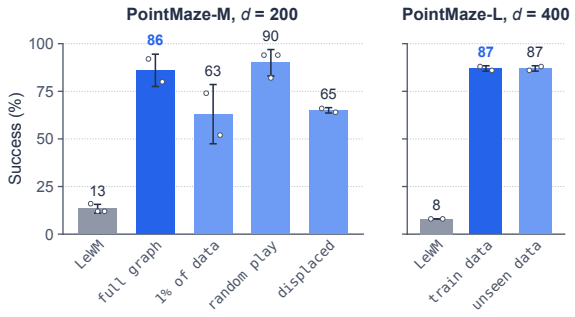

Figure 8: Little and imperfect data suffice. Navigation setting on PointMaze-Medium (d=200: 100% vs. 1% data, random-play graph, mid-episode kidnapping) and PointMaze-Large (d=400: training vs. unseen dataset). Bars show means with standard deviations and single-seed dots.

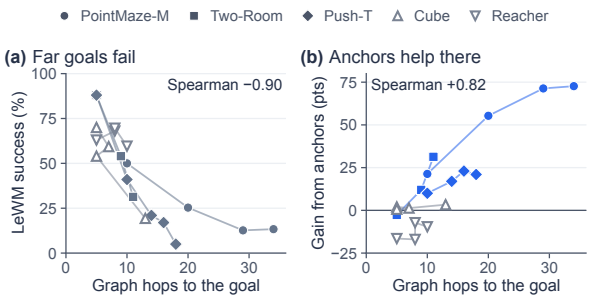

Figure 9: Graph distance predicts where anchors help. One point per environment and distance across all settings. **(a)** LeWM success drops sharply as goal graph distance grows. **(b)** Gain of Graph Anchors over LeWM + min-over-plan tracks goal graph distance.

5.3. Why Is Each Component Needed?

Ablating our three components one by one confirms their distinct roles (Figure 7). On PointMaze-Medium, changing the target alone accounts for nearly the entire gain (Figure 7a): aiming LeWM's unchanged search at our anchors lifts success from 12% to 85%, while aiming our closest-approach search at the distant goal stays at 9%. Heavier search toward the goal cannot replace anchors either: doubling the horizon and more than tripling CEM samples leaves goal-aimed planning at 28% on PointMaze-Medium and 77% on Two-Room, far below default anchors in the same runs (89% and 94%, Table A9b). On Two-Room, both target and closest-approach search contribute (66% to 94%). Finally, on Push-T, the switch cost λ=3 is the decisive contact ingredient: shrinking the anchor to K=3 without λ=3 stays at 30%, whereas adding λ=3 jumps immediately to 72% (Figure 7c).

5.4. Which Design Choices Matter?

Anchor span K balances directional signal against progress per decision (Figure 7b). Placing anchors too far (K=12) reintroduces range failure, dropping PointMaze-Medium from 83% at K=6 to 35%, whereas K=2 stalls under fixed 5-block navigation execution at 57% (resolved by short-horizon closest-approach execution in Table A10b). Optimal span increases with horizon H, from K=4 at H=5 to K=6 at H=10, with K=6 staying within 6 points of optimal across the entire grid (Table A10a). Cross-episode neighbour edges are indispensable: removing them collapses PointMaze-Medium from 86% to 6%, while varying $k \in \{5, 10, 20\}$ barely changes results (Figure A13b).

Graph Anchors are also remarkably robust to scarce or shifted data (Figure 8 and Table A11). On PointMaze-Medium d=200, keeping just 1% of training episodes still reaches 63%, and building the graph purely from random play reaches 90%. On PointMaze-Large d=400, indexing a separate unseen dataset never used during world-model training still achieves 87%. Finally, mid-episode displacement (kidnapping) requires no special recovery logic: replanning re-anchors from the new observation automatically, shifting success only from 86% to 65%.

Table 2: **Diagnostic taxonomy of experience-guided planners.** Each prior method drops at least one axis identified by our diagnosis. GAS and TTGS rely on learned value or policy heads that saturate or collapse on visual mazes. AP omits a graph and requires an oracle step count d. NGC discards temporal transitions and stores an $O(N^2)$ distance table. Metro-WM prices cross-episode bridges below recorded transitions and scores only the final step of fixed-h plans. Graph Anchors satisfies all six axes.

Method	Training-Free	Local Controller	Full-Data Sparse Graph	No Oracle d	Contact-Safe	Pass-Through
GAS [4]	×	Policy (HILP)	×	✓	×	×
TTGS [30]	×	Policy (GCIVL)	×	✓	×	×
AP [23]	✓	Frozen LeWM	×	×	✓	×
NGC [41]	✓	Frozen LeWM	×	✓	×	×
Metro-WM [20]	✓	Frozen LeWM	✓	✓	×	×
Graph Anchors (Ours)	✓	Frozen LeWM	✓	✓	✓	✓

Table 3: **Head-to-head comparison of training-free planners on identical frozen LeWM checkpoints, pairs, and $2d$ budgets** (success, %). AP receives the oracle step count d; Metro-WM tunes (h, H) per task on the evaluation pairs. PointMaze-Large runs use the shared `lewm_pml` checkpoint (seed 0; original checkpoint in brackets). Mean (far) averages PM-M d=200, Two-Room d=75, and Push-T d=150. A single unchanged Graph Anchors contact setting (K=2, λ=3) leads Two-Room, Push-T, and the three-task mean; navigation leads PointMaze-Large by +52 to +70 points.

	PointMaze-M		Two-Room	Push-T		Mean	PointMaze-L	
Method	100	200	75	50	150	(far)	400	Stitched
LeWM [25]	25	13	31	41	5	16.3	10	6
AP [23] (uses true d)	91	80	97	82†	65	80.7	18	9
NGC [41]	**97**	**100**	99	59	11	70.0	24	20
Metro-WM-style [20] (tuned)	95	96	99	72	33	76.0	6	6
Graph Anchors (nav.)	89	85	93	65	30	69.3	**76** [87]	**45** [66]
Graph Anchors (contact)	97	94	**100**	**85**	**71**	**88.3**	13	9

5.5. Diagnosing Training-Free World-Model Planners (AP, NGC, Metro-WM)

Our diagnosis identifies four practical requirements for training-free long-horizon visual planning: (i) a reliable short-range controller, (ii) a sparse graph scaling to 100% of frames without subsampling shortcuts, (iii) preserved recorded transitions where contact matters, and (iv) target-search decoupling without oracle step counts or fragile per-task horizon coupling. Table 2 maps prior graph-guided policies (GAS [4], TTGS [30]) and training-free world-model planners (AP [23], NGC [41], Metro-WM [20]) onto these exact axes. We re-implemented AP, NGC, and Metro-WM inside our shared evaluation framework on identical frozen checkpoints, 5-step training nodes, seeds, and $2d$ budgets; Table A16 verifies that all three match their published results within 9 points despite our tighter $2d$ budget.

As shown in Table 3 and Figure 10a, all four methods resolve basic range failure on medium mazes, yet each alternative breaks down precisely where it drops a diagnostic axis of Table 2:

AP: trajectory retrieval without a graph. Staying inside one recorded episode preserves contact continuity, explaining AP's strong Push-T and Cube scores when given the exact step distance d.

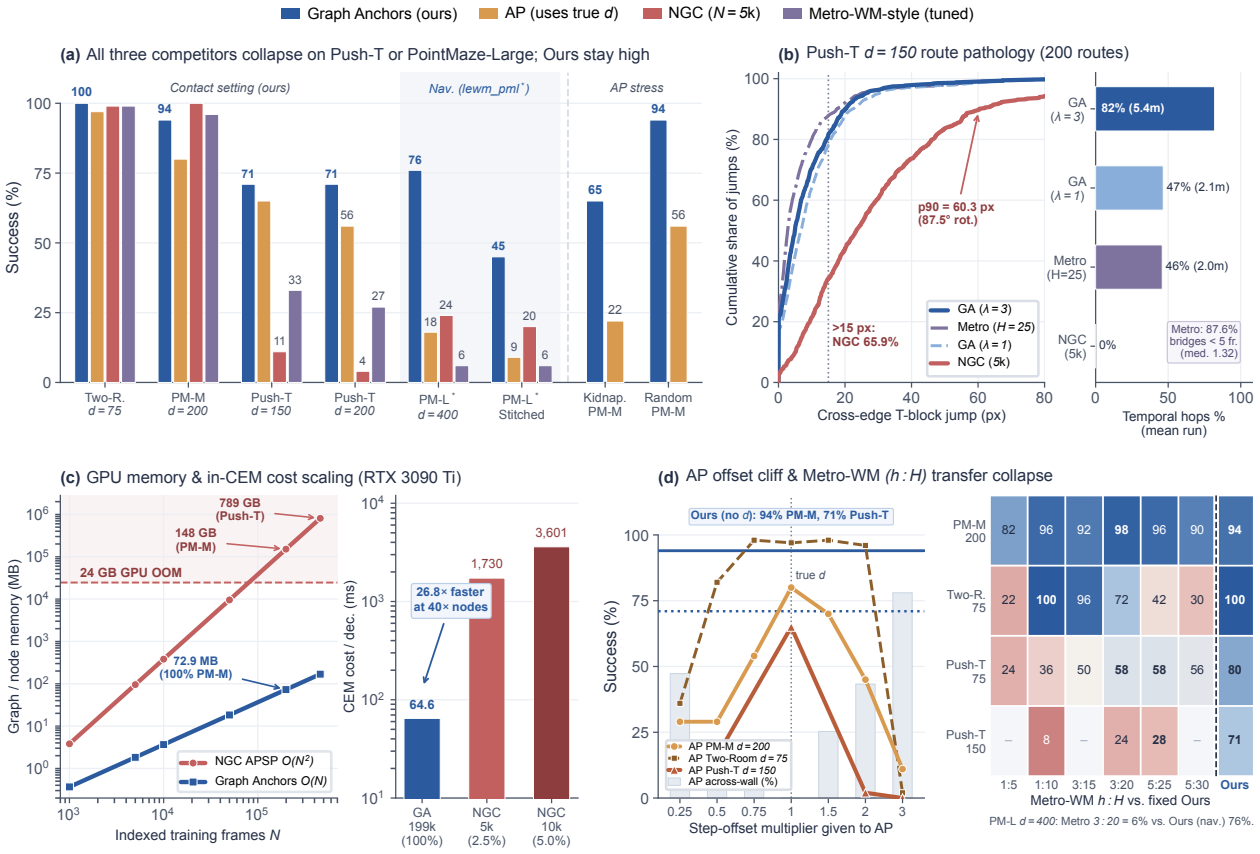

Figure 10: **Quantitative failure profile, scaling wall, and fragility of AP, NGC, and Metro-WM compared to Graph Anchors. (a)** Four-way closed-loop success across six long-range benchmarks (contact setting on Two-Room, PointMaze-Medium, and Push-T; navigation on PointMaze-Large `lewm_pml` and AP stress tests). **(b)** Push-T d=150 route continuity over 200 held-out routes: cross-edge block-jump CDF (left) and recorded temporal share with unbroken push length (right). **(c)** RTX 3090 Ti GPU memory and per-decision CEM cost latency: Graph Anchors indexes 40× more nodes at 26.8× lower latency than NGC. **(d)** AP's sharp drop under mis-specified step distance d tracking across-wall retrieval (left), and Metro-WM's full $(h{:}H)$ sweep heatmap (right) where every column has collapse cells versus Graph Anchors' single unchanged contact column.

However, omitting a graph forces AP to know d in advance and limits it to routes covered by a single training episode. When d is off by 2–3× (Figure 10d, Table A17c), the eligible episode bank shrinks drastically and 21–78% of retrieved starts fall across solid maze walls, dropping PointMaze-Medium from 80% to 11%, Two-Room from 97% to 2%, and Push-T from 65% to 16% (0.5×) and 0% (3×). AP breaks similarly whenever single episodes cannot span the task: stitched goals (23.3% across-wall starts, Figure 11d), PointMaze-Large (18% and 9% versus our 76% and 45%), mid-episode kidnapping (22% vs. 65%), random-play data (56% vs. 94%), and Reacher (34% vs. 68%).

NGC: undirected manifold cost under $O(N^2)$ subsampling. Replacing Euclidean distance with shortest-path length over 5,000 undirected snapshots solves medium mazes (100% on PointMaze-Medium), but violates two axes. First, discarding directed temporal edges creates a clear *geometry-vs.-*



Figure 11: **Visual breakdown of planner failures on real encoded datasets. Top (PointMaze-Medium, exact** 8×8 **walls):** On 1% sparse data, NGC **(a)** and Metro-WM (H=25) **(b)** admit red long-jump shortcuts exceeding a real 5-step move (1.37% up to 2.1× and 1.88% up to 3.0×), whereas Graph Anchors' 1-block τ **(c)** admits 0.00% shortcuts. **(d)** On a stitched pair, AP retrieves a start 6.07 units away across solid walls (red dotted), while Graph Anchors stitches cleanly around the wall (blue path, orange diamonds). **Bottom (Push-T** d=150**, identical held-out pair with true T-block poses):** NGC **(e)** teleports the block by up to 217 px and 131°; Metro-WM **(f)** breaks the path into 17 episode switches (colours); Graph Anchors (λ=3) **(g)** follows continuous pushes with just 1 switch, lifting Push-T d=150 from 11% and 33% up to 71%.

executability gap on Push-T (Figure 10b, Table A17a): NGC scores higher offline object-pose ranking accuracy (ρ_{obj}=0.715) than Graph Anchors (0.649) because undirected snapshots hop across state space without obeying pushing physics, yet 65.9% of its route hops teleport the block over 15 px (up to 217 px / 131° in Figure 11e), collapsing closed-loop Push-T to 11% at d=150 and 4% at d=200. Second, storing an $N\times N$ distance table and re-attaching every CEM candidate hits an $O(N^2)$ wall (Figure 10c): on an RTX 3090 Ti, N=5,000 already takes 95.4 MB and 1,729.9 ms per decision, while full datasets would require 148 GB on PointMaze-Medium and 789 GB on Push-T. Capping N=5,000 on 200k-frame PointMaze-Large forces **9.51%** of edges into false long-jump shortcuts up to 22.8 maze units, collapsing NGC to 24% / 20%. Graph Anchors indexes all 199k–200k nodes in 72.9 MB and 64.6 ms per decision, running 26.8× faster.

Metro-WM: underpriced cross-episode bridges and terminal-h planning. Metro-WM builds a sparse frame graph like ours, directly isolating why our three design choices matter. First, checking

Table 4: **Diagnosis and closed-loop outcome per environment.** $\rho(1)$ and $\rho(6)$ report directional ranking one and six hops from the goal. Hops gives median goal graph distance. LeWM and Ours report success (%) at the longest distance (^{c}contact setting). Closer gives the percentage of decisions ending closer to the goal (LeWM $\rightarrow$ Ours).

Environment	$\rho(1)$	$\rho(6)$	Hops	Failure	LeWM	**Ours**	Closer (%)
PM-M 200	0.99	0.59	34	range	13	**85**	48→72
Two-Room 75	0.99	0.50	11	range	31	**93**	33→74
Reacher 150	1.00	0.56	10	–	60	**68**	–
Push-T 150	0.91	0.74	18	focus	5	**71**^{c}	31→53
Cube 150	0.99	0.77	13	focus	19	**39**^{c}	–

80,000 Push-T bridges shows that **87.6%** are priced below a real 5-frame temporal edge (median 1.32 frames; 100% on PointMaze-Large). Dijkstra therefore abandons recorded pushes immediately (Figure 10b, Figure 11f), splitting Push-T into 7.03 switches per 6 moves and stalling success at 33% (d=150) and 27% (d=200). Our switch cost λ=3 cuts switches 4.2$\times$ to 1.68, stretches unbroken pushes 2.7$\times$ to 5.40 moves (Figure 11g), and lifts Push-T d=150 to 71%. Second, scaling bridge radius with budget H admits sparse-data shortcuts: on 1% PointMaze-Medium data (Figure 11b), 1.88% of bridges exceed a real 5-step move up to 3.0$\times$ (dropping success from 98% to 76–78%), versus 0.00% shortcuts for our fixed 1-block τ. Third, executing fixed-h terminal-cost plans prevents any (h, H) pair from transferring across tasks (Figure 10d): Two-Room's best (1:10, 100%) drops to 8% on Push-T d=150, while PointMaze-Medium's best (3:20, 98%) drops to 72% on Two-Room, 33% on Push-T, and 6% on PointMaze-Large d=400. Full reproduction checks, Push-T distance curves through d=200, and complete numbers appear in Section D.2.

5.6. When Does It Help?

Our diagnosis also predicts where Graph Anchors help before running any evaluation. First, aiming at a nearby target restores positive closed-loop drift. Counting progress in hops on a true-state reference graph, fewer than half of LeWM's decisions end closer to the goal on all three traced benchmarks. Adding Graph Anchors raises improving decisions to 72% on PointMaze-Medium, 74% on Two-Room, and 53% on Push-T (Table 4 and Section A.3). As Proposition 1 proves, positive net drift changes expected hitting time from quadratic $\Omega(D^2)$ random wandering to linear $O(D)$ progress.

Second, offline goal graph distance strongly predicts both failure and anchor gain across all environments and distances (Figure 9). It correlates with LeWM success at Spearman -0.90 across 20 settings, and with the gain of Graph Anchors over LeWM + min-over-plan at Spearman 0.82 across 19 settings. Table 4 pairs each offline diagnosis directly with its closed-loop outcome: where range dominates, navigation anchors solve the task; where focus dominates, contact anchors solve Push-T and improve Cube; and on compact Reacher (goals staying within 10 hops at every distance), LeWM already reaches 60% and intermediate anchors only add small detours (removable via optional gated goal handoff in Section C.2).

6. Conclusion

The visual goal cost of a latent world model ranks local progress accurately only when its target is nearby and lies on a recorded route. Measuring this directly separates two root failures, *range* and *focus*, explains why distant goals and contact tasks collapse, and predicts where retrieved subgoals help. Graph Anchors put this finding to work without modifying or retraining the world model, using a

sparse full-dataset graph, cross-episode switch penalties, and closest-approach scoring to remove range failure across mazes and focus failure on Push-T while avoiding the step-horizon fragility, quadratic scaling wall, and contact fragmentation of alternative training-free planners.

Limitations. Graph Anchors rely on recorded trajectories covering the route and use two task-level settings for navigation versus contact; the contact setting trades off efficiency on long stitched mazes and short-range Reacher. Because a recorded route cannot synthesize a delicate grasp when the local world model misses initial contact, Cube remains below AP, and our Meta-World demonstration test serves as an exact-dynamics upper bound rather than a deployable manipulation controller. Extending training-free graph guidance to dexterous grasping and open-world video world models remains an exciting direction for future work.

References

[1] Ali Alrasheed, Basim Azam, and Naveed Akhtar. The planning limits of latent world models. *arXiv preprint arXiv:2609.39235*, 2026.

[2] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15619–15629, 2023.

[3] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[4] Seungho Baek, Taegeon Park, Jongchan Park, Seungjun Oh, and Yusung Kim. Graph-assisted stitching for offline hierarchical reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pages 2391–2408. PMLR, 2025.

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

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

[7] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[8] Tianqi Du, Qi Zhang, Yifei Wang, and Yisen Wang. Beyond the next step: Variable-length latent world models for long-horizon planning. *arXiv preprint arXiv:2606.21775*, 2026.

[9] Scott Emmons, Ajay Jain, Misha Laskin, Thanard Kurutach, Pieter Abbeel, and Deepak Pathak. Sparse graphical memory for robust planning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 5251–5262, 2020.

[10] 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, 2019.

[11] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793, 2017.

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

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

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

[15] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 2555–2565. PMLR, 2019.

[16] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020.

[17] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

[18] Zhiao Huang, Fangchen Liu, and Hao Su. Mapping state space using landmarks for universal goal reaching. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[19] Yoo Sung Jang and Michael Ryoo. Latent projection for latent planning. *arXiv preprint arXiv:2610.26979*, 2026.

[20] Royson Lee, Fady Rezk, Titouan Parcollet, Timothy Hospedales, and Cristina Cornelio. Metro-WM: Long-horizon latent planning with realisable sub-goals. *arXiv preprint arXiv:2609.31868*, 2026.

[21] Liangyu Li, Shengzhi Wang, Libin Qiu, Mingliang Xiong, and Qingwen Liu. World model control by trajectory reachability metrics. *arXiv preprint arXiv:2605.22164*, 2026.

[22] Kara Liu, Thanard Kurutach, Christine Tung, Pieter Abbeel, and Aviv Tamar. Hallucinative topological memory for zero-shot visual planning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 6259–6270. PMLR, 2020.

[23] Xvyuan Liu, Jianjie Fang, Wei Wu, Chen Gao, and Yong Li. Aim short to reach far: Your frozen world model can plan better than you think. *arXiv preprint arXiv:2609.30036*, 2026.

[24] Yang Liu, Hanchen Cui, and Karthik Desingh. PILOT: Guiding long-horizon world-model planning for robot manipulation with imagined subgoals and policy priors. *arXiv preprint arXiv:2610.14335*, 2026.

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

[26] 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.

[27] Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[28] Lama Moukheiber, Haotian Xue, and Yongxin Chen. How long, not how close: A learned temporal metric for planning in latent world models. *arXiv preprint arXiv:2610.04988*, 2026.

[29] Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[30] Evgenii Opryshko, Junwei Quan, Claas A. Voelcker, Yilun Du, and Igor Gilitschenski. Test-time graph search for goal-conditioned reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, volume 306 of *Proceedings of Machine Learning Research*, pages 94754–94774. PMLR, 2026.

[31] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024.

[32] Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with Hilbert representations. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 39737–39761. PMLR, 2024.

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

[34] Karl Pertsch, Oleh Rybkin, Frederik Ebert, Shenghao Zhou, Dinesh Jayaraman, Chelsea Finn, and Sergey Levine. Long-horizon visual planning with goal-conditioned hierarchical predictors. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 17321–17333, 2020.

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

[36] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations (ICLR)*, 2018.

[37] Joyjeet Singh. The objective is the bottleneck: Latent world models encode what their planners cannot use. *arXiv preprint arXiv:2608.12959*, 2026.

[38] 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 (NeurIPS)*, volume 38, pages 43905–43941, 2025.

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

[40] 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 (TMLR)*, 2026.

[41] Zunzhi You, Linwei Tao, Siyu Xu, Yunke Wang, and Chang Xu. Latent world models should plan with distances along the latent manifold. *arXiv preprint arXiv:2610.25013*, 2026.

[42] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey

Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, pages 1094–1100. PMLR, 2020.

[43] Lunjun Zhang, Ge Yang, and Bradly C. Stadie. World model as a graph: Learning latent landmarks for planning. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 12611–12620. PMLR, 2021.

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

[45] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pages 79115–79135. PMLR, 2025.

Appendix Table of Contents

A Extended Goal-Cost Diagnosis 21
A.1 Range Failure Across Encoders and Epochs 22
A.2 Focus Failure Across Encoders and Pushing 27
A.3 Formal Analysis and Closed-Loop Drift 28
B Graph Anchors: Algorithm and Implementation 32
B.1 Algorithm and Walkthrough 32
B.2 Implementation Details and Graph Quality 34
C Extended Results and Ablations 35
C.1 All Distances, Retraining Sensitivity, and Paired Tests 35
C.2 Contact Setting, Switch Cost, and Goal Handoff 38
C.3 Ablations, Scaling, and Robustness Stress Tests 39
D Additional Baselines and Planner Details 41
D.1 World-Model Baselines and Graph-Guided Policies (GAS, TTGS) 41
D.2 Implementation Fidelity, Push-T Scaling, and Full Metrics for AP, NGC, Metro-WM 44
E Full Per-Evaluation Results, Qualitative Rollouts, and Reproducibility 46

A. Extended Goal-Cost Diagnosis

On the standard 25-step LeWM evaluation setup summarized in Table A1, both visual world-model planners and offline policies solve most goals easily. Once the goal is placed further away, however, those same planners break down sharply. As shown in Figure 6 and Table A6, LeWM drops to 13% at d=200 on PointMaze-Medium, 31% at d=75 on Two-Room, 5% at d=150 on Push-T, and 8% at d=400 on PointMaze-Large.

Simply throwing more search compute at the distant goal does not rescue the planner. Doubling the planning horizon and more than tripling the number of CEM samples lifts LeWM to only 28% on PointMaze-Medium and 77% on Two-Room in Table A9b. Even switching to our closest-approach scoring while still aiming at the distant goal stays at 9% on PointMaze-Medium in Table A9a. Tracing closed-loop rollouts in true-state graph hops reveals what goes wrong at runtime. Fewer than half of LeWM's decisions actually end closer to the goal, with only 48% improving on PointMaze-Medium, 33% on Two-Room, and 31% on Push-T, as documented in Table 4 and Table A4.

In this appendix we expand our diagnosis of the two root causes inside the latent goal cost. Section A.1 tests *range* across six image representations and ten training epochs, Section A.2 tests *focus* across five encoders and nine Meta-World tasks, and Section A.3 formalizes how local ranking translates into closed-loop drift.

Table A1: **Published success (%) on the standard LeWM benchmark.** Results are taken from Fig. 6 of [25], where goals lie 25 steps ahead with a 50-step budget. These numbers provide short-horizon context only, since none of these methods was evaluated at long range in that work.

Environment	LeWM	PLDM	DINO-WM	GCBC	GCIQL	GCIVL
Push-T	96	78	74	75	20	33
Two-Room	87	97	100	100	100	100
OGBench-Cube	74	65	86	84	64	56

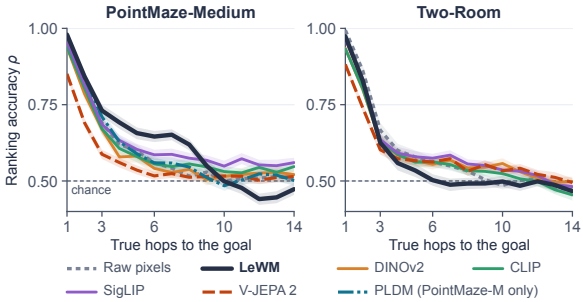

Figure A1: **Every image encoder loses its direction a few hops from the goal.** Ranking accuracy $\rho(h)$ against true-position reference hops for six representations on identical frames and 200 goals per environment, together with one-epoch PLDM on PointMaze-Medium. Bands show 95% bootstrap intervals and dashed line marks chance.

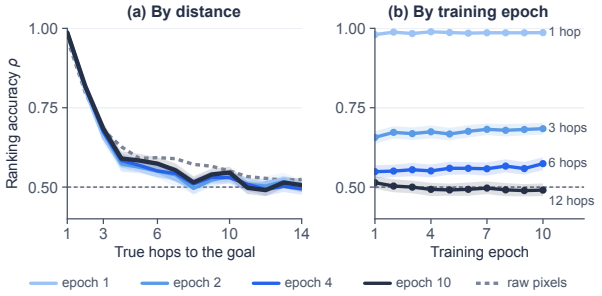

Figure A2: **Training longer does not extend range.** LeWM checkpoints from epochs 1 to 10 of one PointMaze-Large training run on identical frames and 200 goals, alongside raw pixels as a control. Panel (a) plots $\rho(h)$ against true-position hops, and panel (b) tracks ρ at 1, 3, 6, and 12 hops across epochs.

A.1. Range Failure Across Encoders and Epochs

Measuring directional ranking. To test whether a latent distance actually guides an agent toward a distant goal, we ask a simple local question. Suppose the agent is at a recorded frame v that lies h graph hops from a goal frame g, and consider two neighbours u and u' of v where one is genuinely closer to g than the other. We count a latent distance as correct on this pair if it assigns a smaller cost to whichever neighbour is truly closer to g. The ranking accuracy $\rho(h)$ is the share of pairs ordered correctly. A value of $\rho{=}1$ means the distance always points downhill toward the goal, whereas $\rho{=}\frac{1}{2}$ carries no directional information and is no better than flipping a coin.

To avoid any circularity with a model's own latent graph, all cross-encoder and training-epoch checks below measure hops on a fixed *reference graph* built strictly from ground-truth positions. This reference graph connects consecutive recorded frames along each episode and adds edges between frames closer in physical position than the median five-step move. Every encoder is tested on the exact same five-step frames drawn from a seeded fifth of the episodes, over 200 goals with up to four frames per goal and hop. Shaded bands in the plots mark 95% bootstrap confidence intervals over goals.

Six encoders, including raw pixels. Figure 2b evaluates LeWM, PLDM, and frozen DINOv2. Here we expand that comparison to six image representations on identical frames and goals: raw pixels average-pooled to 32×32 with no learned network at all, alongside frozen DINOv2-S, CLIP ViT-B/16,

Table A2: **Graph-free ground-truth checks.** For a state k action blocks before a goal in the same recorded episode, $\rho_T(k)$ reports how often the latent cost prefers one block forward at $t{+}1$ over one block backward at $t{-}1$ across 4,000 states per k, where 0.5 is chance. The Spearman columns report rank correlation between latent cost and true steps to the goal locally within 5 blocks and globally over all $k{\leq}30$.

	$\rho_T(k)$			Spearman	
Environment	$k{=}3$	6	10	local	all
Push-T	0.96	0.83	0.60	0.69	0.48
Two-Room	0.62	0.56	0.56	0.26	0.27
PointMaze-M	0.98	0.66	0.32	0.82	0.27
Reacher	0.74	0.59	0.52	0.43	0.33
OGBench-Cube	0.84	0.68	0.61	0.54	0.44

SigLIP-B/16, video-trained V-JEPA 2 ViT-L, and LeWM. For each vision backbone, we compute squared distance over patch features pooled to $4{\times}4$, matching how the planners score images.

As shown in Figure A1, all six representations follow the exact same curve. One hop from the goal, every encoder ranks local moves reliably, with $\rho(1)$ spanning 0.85 to 0.98 on PointMaze-Medium and 0.88 to 0.99 on Two-Room. Within just a few hops, however, that directional signal fades toward chance. Six hops from the goal, $\rho(6)$ falls to 0.52–0.65 on PointMaze-Medium and 0.50–0.57 on Two-Room, crossing below 0.55 between hops 5 and 10. At twelve hops on PointMaze-Medium, LeWM drops to $\rho{=}0.44$ and even leans slightly away from the goal. PLDM behaves identically on the same frames, dropping from $\rho(1){=}0.98$ to $\rho(6){=}0.56$ after one epoch and 0.54 after five epochs.

The raw-pixel control highlights why this happens. Because raw pixel distance decays just like every learned encoder, including video-trained V-JEPA 2, short range is not a flaw of a specific architecture or training loss. It is a basic property of judging progress by direct distance between two whole images when walls and corridors lie between them. Figure A4 and Figure A5 show this geometry visually. Far from the goal, every encoder treats frames in distant corridors or behind solid walls as just as close to the goal as frames sitting on the true path.

Training longer does not extend range. One might wonder whether short range simply reflects an undertrained world model. To check this, we run the exact same ranking test across every epoch from 1 to 10 of our second PointMaze-Large LeWM training run from Section C.1, keeping raw pixels as a control. As shown in Figure A2, extended training does not fix the decay. One hop from the goal, every checkpoint is already highly accurate, moving from 0.98 at epoch 1 to 0.99 at epoch 10. Six hops away, $\rho(6)$ shifts only from 0.55 to 0.57, staying slightly below raw pixels at 0.59. Twelve hops away, accuracy remains right at chance from 0.51 to 0.49. Over all ten epochs, the hop where ranking drops below 0.55 moves only from hop 6 to hop 8. Combined with the five-epoch PLDM check above, this confirms that longer training does not extend visual range.

Graph-free checks on recorded time. Because Equation (3) measures distance using graph hops, it is important to confirm that the range collapse is not an artifact of building a graph. We therefore run a completely graph-free test directly along recorded episodes and report the results in Table A2. Take a frame at time t and a future goal k blocks ahead at $t{+}k$ within the same episode, testing $k{\geq}2$ across 4,000 states per separation. We simply check whether stepping one block forward to $t{+}1$ yields

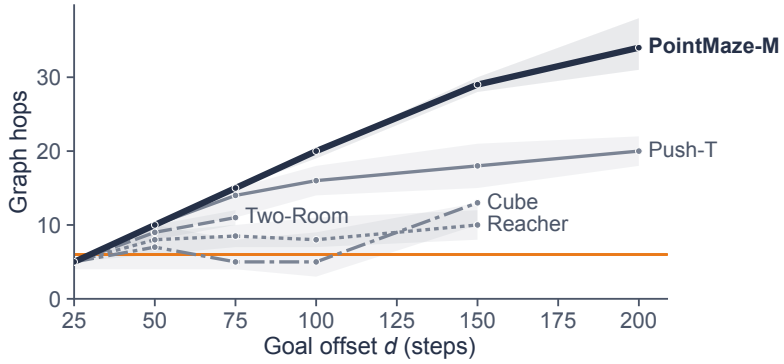

Figure A3: **Goal distance in the graph.** Curves show the median and interquartile range of graph hops between a start and a goal d steps later in the same episode across 200 pairs, with the orange line marking our 6-hop anchor span. On PointMaze-Medium and Push-T, goals move steadily farther in graph hops as d grows, whereas on Reacher they saturate near ten hops.

a lower latent distance to the goal than stepping one block backward to $t-1$.

The exact same collapse appears without any graph. Close to the goal, ρ_T stays near 1, so the latent cost reliably distinguishes forward from backward. By 10 blocks away, however, ρ_T drops to chance or below on every environment except Cube, falling to 0.32 on PointMaze-Medium. On Cube, ρ_T rises back to 0.77 at 20 blocks because the arm often swings back near an earlier pose during longer trajectories. Spearman correlation with true steps to the goal confirms the same drop, falling on PointMaze-Medium from 0.82 locally within 5 blocks to 0.27 globally across $k \leq 30$. Finally, Figure A3 shows why Reacher behaves differently from the other benchmarks. Even when a Reacher goal lies 150 steps ahead in time, its graph distance plateaus around ten hops inside the compact workspace, whereas PointMaze-Medium and Push-T stretch far past the reliable local range.

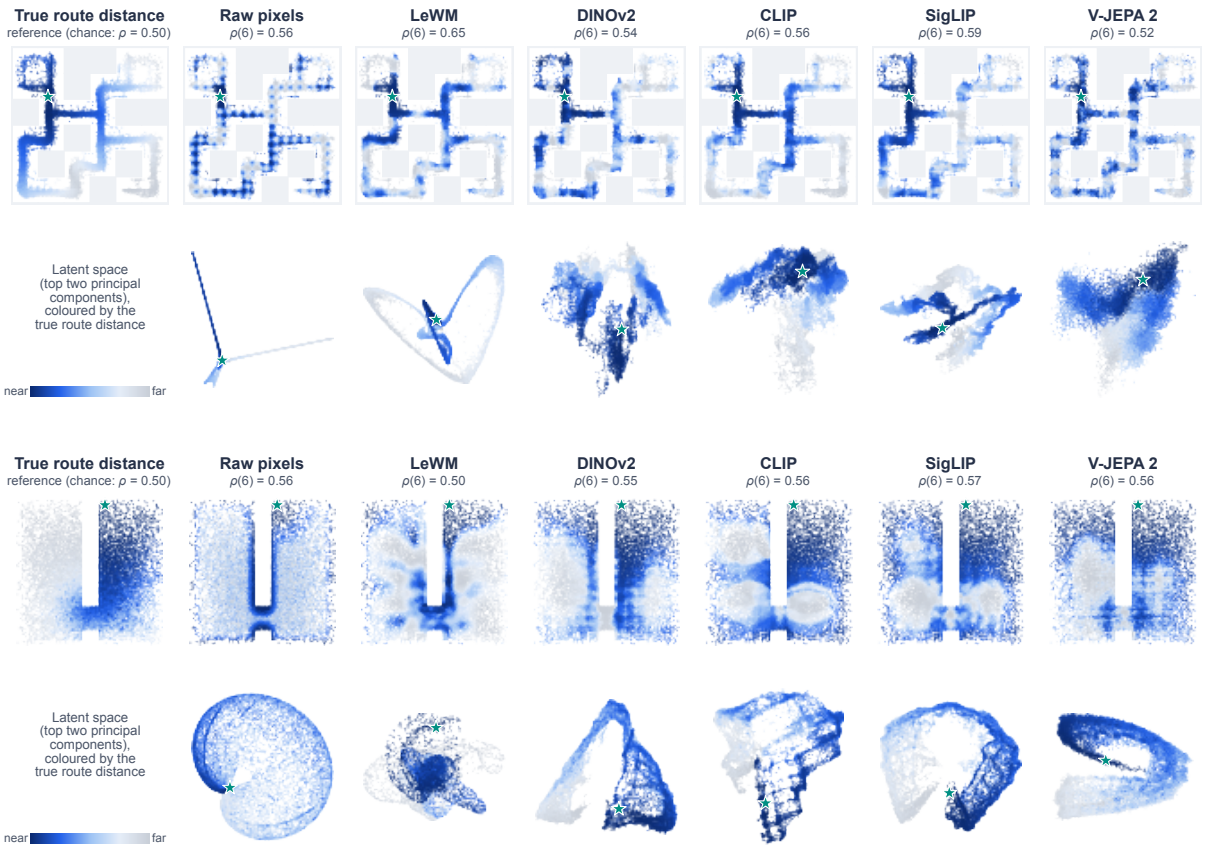

Figure A4: **Where each encoder thinks the goal is near.** The top two rows show PointMaze-Medium and the bottom two rows show Two-Room for one goal marked by a star. In each block, the first row shows true route distance followed by each encoder's latent distance on the same 20k recorded frames plotted as ranks, where darker shades are closer. The second row projects each latent space onto its top two principal components coloured by true route distance, and each title reports $\rho(6)$ over all 200 goals. On Two-Room the goal sits in the right room with only a bottom doorway, yet every encoder treats much of the left room as close to the goal.

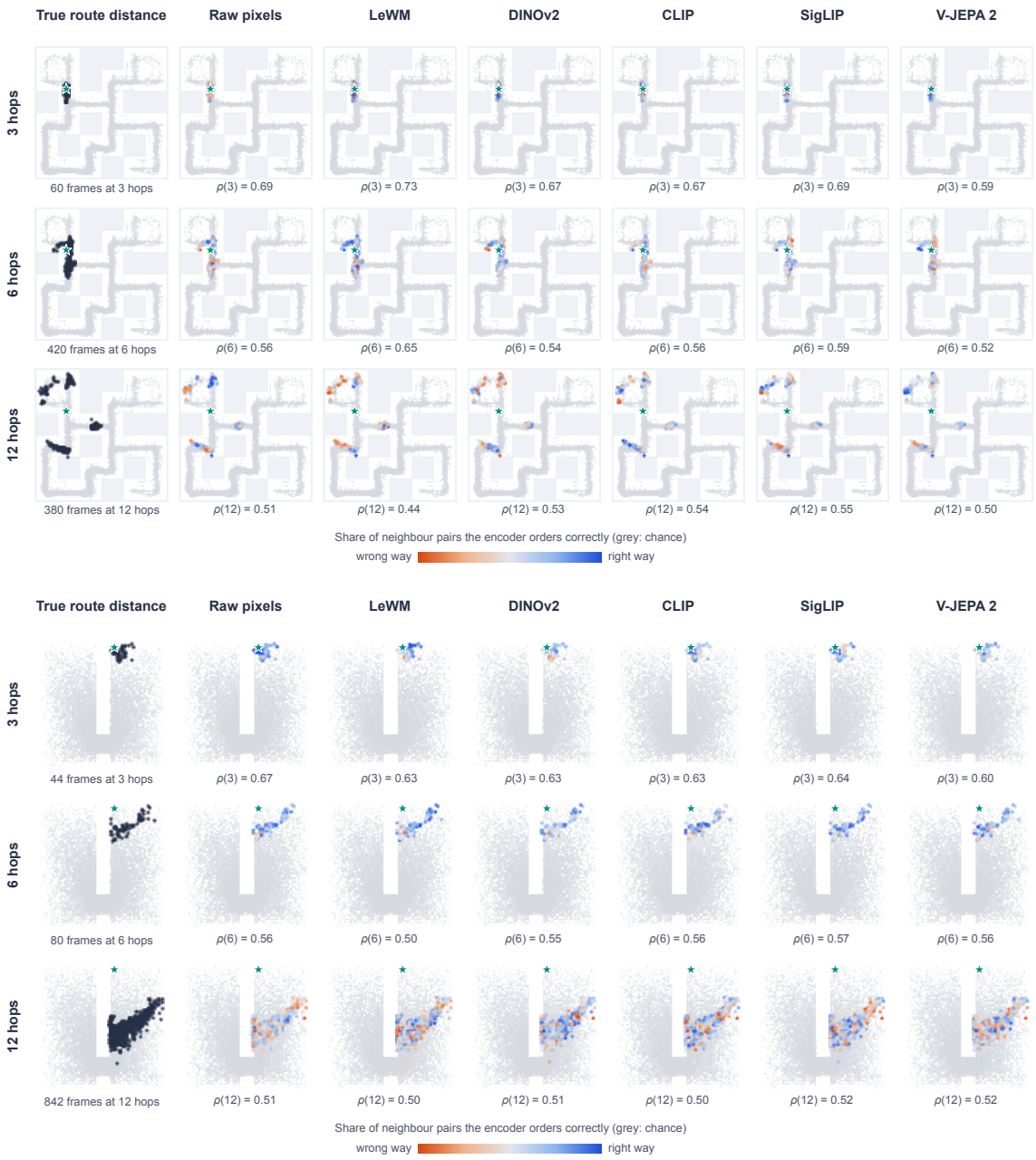

Figure A5: **Frame-by-frame collapse of directional ranking** on the goals of Figure A4, with PointMaze-Medium on top and Two-Room on the bottom. Each row displays recorded frames located exactly 3, 6, or 12 true hops from the goal, coloured by the fraction of reference-graph neighbour pairs ordered correctly by each encoder. Blue points toward the goal, orange points away, grey marks chance, and each panel lists $\rho(h)$ over all 200 goals. At 3 hops every encoder is mostly blue, but by 12 hops blue and orange mix together randomly so decreasing latent distance is no better than chance.

A.2. Focus Failure Across Encoders and Pushing

Short range explains why planners fail on mazes, but Push-T exposes a second obstacle. On Push-T, the latent cost still ranks local moves well six hops from the goal with $\rho(6)$=0.74, yet closed-loop planning still falls to 5% at d=150. This second failure concerns *what* whole-image distance actually weighs.

Setup and exchange rate across five encoders. To isolate focus failure cleanly on manipulation, we evaluate the released JEPA-WM Meta-World model of [40], which encodes images into 256 patch tokens using frozen DINOv2 ViT-S/14. Its CEM planner optimizes 6 action chunks with 300 samples, 10 elites, and 15 iterations across 100-step episodes. To remove dynamics prediction error completely, every candidate sequence is rolled out inside a copy of the true simulator and only its final frame is encoded and scored. We evaluate six object tasks and two controls over 12 identical test episodes per task.

We next measure how strongly an image distance favors the robot arm over the object. Using inverse kinematics, we place the arm and the object independently across 30 configurations in each of eleven Meta-World tasks. We then compute the *exchange rate*, defined as the feature distance caused by shifting only the hand by 5 cm divided by the distance caused by leaving only the object off target by its success threshold. Any value above 1 means the visual cost prefers fixing a small hand error over finishing the task.

As shown in Figure A6, every image distance sits far above 1 across all 9 object tasks. Shifting the hand by 5 cm costs 6.3 to 35$\times$ more than missing the object in raw pixels, and roughly 2 to 14$\times$ more across DINOv2, SigLIP, and V-JEPA 2, with CLIP neutral at 1.0 on PLATE-SLIDE and 1.9 to 16$\times$ on the other eight tasks. Inspecting real start and goal frames explains why. Robot patches fill roughly half the image yet carry 72 to 79% of the squared feature distance, even though a linear readout on those exact same frozen DINOv2 features locates the object within 0.14 to 0.77 of the success threshold.

The same bias stalls pushing. Push-T displays the exact same imbalance in a 2D workspace. Even though overall local ranking looks healthy at $\rho(1)$=0.91 and $\rho(6)$=0.74, the latent cost orders the T-block's true object progress correctly only 0.54 of the time because the moving agent circle dominates image changes. By contrast, graph distance along recorded routes orders true object progress correctly 0.68 of the time, as plotted in Figure A7c. Without keeping routes on continuous recorded pushes, merely placing an anchor three moves ahead leaves Push-T success stuck at 30% in Table A8.

Meta-World remedy: learned costs vs. near recorded targets. Table A3 compares two possible fixes under exact dynamics. Because the focus failure lives in the cost function, we first test alternative costs built on the same frozen DINOv2 features. Both the released visual distance and TEMPO [28], a learned temporal-distance cost that remains dominated by arm travel time, solve 0% of the six object tasks. A learned progress head trained on planner-visited states, which scores hand-to-object distance before contact and object-to-goal distance after contact, lifts mean success to 68% while keeping both controls intact. However, once exact simulator rollouts are replaced with the learned predictor, success falls back near zero because predicted latent futures fail to resolve fine contact.

Second, we leave the released cost, the CEM search, and exact dynamics completely unchanged, and replace only the final goal image with the encoded frame 20 steps ahead on the test episode's own demonstration. That single change solves 65 of 72 object episodes, averaging 90% with 50% on PUSH. Because this check reads from the test demonstration, it acts as a privileged upper bound rather than

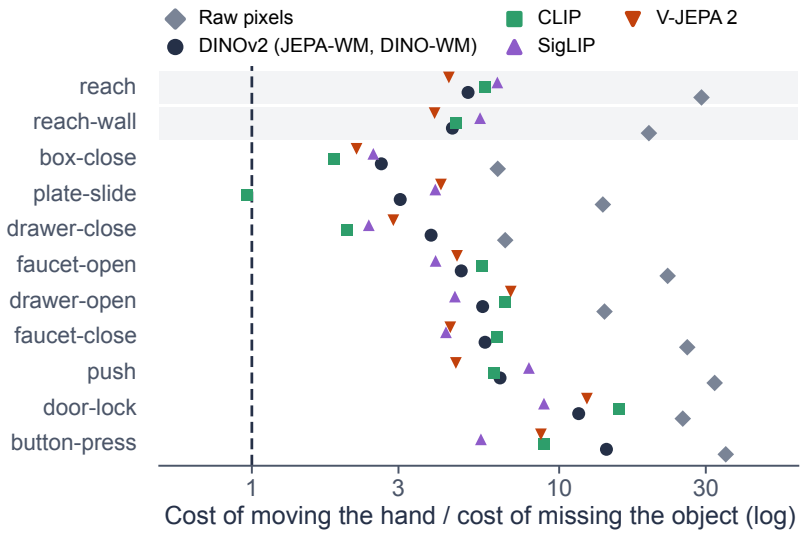

Figure A6: **Every image distance weighs the robot arm over the object.** Bars show the median exchange rate over 30 inverse-kinematics placements per task across five visual representations. This ratio divides the feature distance of shifting only the robot hand by 5 cm by the distance of leaving only the object off target by its success threshold. Shaded rows mark reach tasks where positioning the arm is the entire task. Anywhere to the right of the dashed line, the cost prefers fixing a small hand error over completing the task.

Table A3: **Meta-World closed-loop success (%) under exact dynamics.** Each entry averages 12 identical test episodes per task. The two controls verify baseline behavior when no focus failure exists, since DRAWER-CLOSE is already solved by the released cost and REACH-WALL depends only on the arm. Aiming the unchanged released cost at a demonstration frame 20 steps ahead serves as a privileged upper bound and solves 65 of 72 object episodes, averaging 90% with 50% on PUSH. Controls were not re-run for this upper bound.

	Object tasks							Controls	
Goal cost	BUTTON	PUSH	DRAWER-OPEN	PLATE-SLIDE	FAUCET-OPEN	WINDOW-OPEN	Mean	DRAWER-CLOSE	REACH-WALL
Released latent cost [40]	0	0	0	0	0	0	0.0	100	96
TEMPO temporal distance [28]	0	0	0	0	0	0	0.0	100	42
State head (privileged labels)	0	0	0	67	0	0	11.1	100	58
Image-only reach-then-change	25	0	8	0	0	0	5.6	83	67
Cost trained on planner states, one round	25	58	**83**	**92**	17	58	55.6	100	92
Cost trained on planner states, two rounds	**83**	**83**	25	**92**	**33**	**92**	**68.1**	100	83

a deployable method. Even so, it establishes a unifying principle. *Even the unchanged whole-image distance succeeds at manipulation when its target is nearby and lies on a recorded route*, because matching the arm pose along a recorded trajectory automatically moves the object as recorded. Graph Anchors turns this principle into a deployable, training-free planner (Section 4 and Section B).

A.3. Formal Analysis and Closed-Loop Drift

We now formalize why aiming at a nearby graph anchor moves the agent closer to a distant goal, how approximation error in graph distance affects that guarantee, and why losing directional pull forces slow quadratic hitting times. Let $d(\cdot,\cdot) \in [0,\infty]$ be the directed shortest-path quasimetric on the graph

G with edge weights $w(i,j) \geq 1$. Even when temporal edges are directed and cross-episode edges carry a higher weight $\lambda > 1$, the directed triangle inequality $d(y,g) \leq d(y,a) + d(a,g)$ still holds. Write x_t for the state at decision t and $h_t = d(x_t,g) \geq 0$ for the remaining graph distance to g. In plain terms, Lemma 1 says that moving closer to any stepping stone along a shortest path guarantees at least as much progress toward the final goal.

Lemma 1 (Progress transfer). *Suppose $d(x,g) < \infty$ and let a lie on a shortest directed path from x to g. Then, for every state y,*

$$d(x,g) - d(y,g) \;\geq\; d(x,a) - d(y,a).$$

In particular, $d(y,a) < d(x,a) \implies d(y,g) < d(x,g)$.

Proof. Since a lies on a shortest path from x to g, $d(x,g) = d(x,a) + d(a,g)$. By the directed triangle inequality, $d(y,g) \leq d(y,a) + d(a,g)$. Subtracting the two gives $d(x,g) - d(y,g) \geq d(x,a) - d(y,a)$.□

Because our latent graph distance d approximates the true physical route distance d^*, Corollary 1 checks how much distortion a single planning step can absorb while still guaranteeing true physical progress.

Corollary 1 (Transfer under metric distortion). *Let d^* be a true distance and suppose there exists $\epsilon \in [0,1)$ such that*

$$(1-\epsilon)\, d^*(u,g) \;\leq\; d(u,g) \;\leq\; (1+\epsilon)\, d^*(u,g)$$

for $u \in \{x,y\}$. Let $m = d(x,a) - d(y,a)$ be the progress toward the anchor in Lemma 1. If $m > 2\epsilon\, d^*(x,g)$, then $d^*(y,g) < d^*(x,g)$.

Proof. By Lemma 1, $d(y,g) \leq d(x,g) - m \leq (1+\epsilon)\, d^*(x,g) - m$. Since $1-\epsilon > 0$, dividing by $1-\epsilon$ yields

$$d^*(y,g) \;\leq\; \frac{d(y,g)}{1-\epsilon} \;\leq\; \frac{(1+\epsilon)\, d^*(x,g) - m}{1-\epsilon},$$

which is strictly less than $d^*(x,g)$ if and only if $m > 2\epsilon\, d^*(x,g)$. □

Empirical stretch and multi-step behavior. On PointMaze-Medium, using hops on a true-position reference graph as d^*, the ratio d/d^* stays tightly concentrated near 1. From the 5th to the 95th percentile, it lies between 0.92 and 1.08 at 10–19 hops and between 0.97 and 1.03 beyond 30 hops. That corresponds to roughly $\epsilon \approx 0.03$, so even 34 hops from the goal a single-step margin $m > 2\epsilon d^* \approx 2$ hops suffices for most pairs.

More importantly, that linear-in-d^* margin is only a worst-case single-step artifact caused by distortion pushing in opposite directions at x and y. Distortion does *not* compound over an episode. Whenever $(1-\epsilon)d^* \leq d \leq (1+\epsilon)d^*$ holds across T decisions, we have $h_0 = d(x_0,g) \leq (1+\epsilon)\, d^*(x_0,g)$ and $d^*(x_t,g) \leq h_t/(1-\epsilon)$ at every step. Any sustained negative drift $-\mu$ in graph distance d from Proposition 1b therefore drives $d^*(x_t,g)$ to zero with no $2\epsilon D$ step barrier, inflating the total horizon by at most the constant factor $1+\epsilon$.

Proposition 1 connects per-decision progress to total episode length. Intuitively, an agent without downhill pull wanders like a random walk and needs quadratically many decisions to reach a distant goal, whereas even a modest steady drift toward the goal cuts the reaching time down to linear.

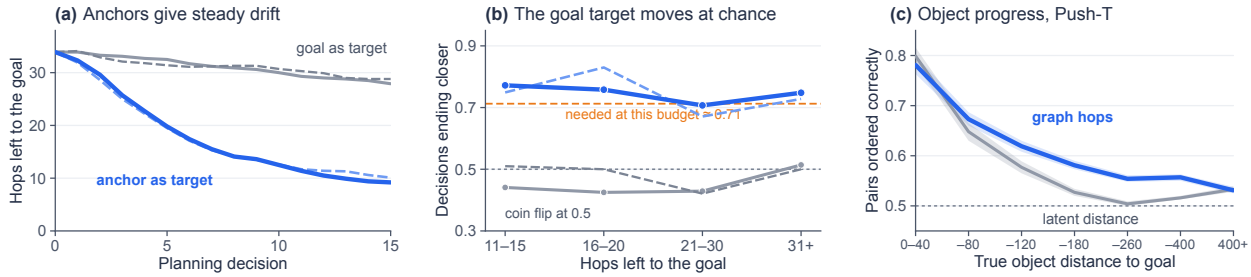

Figure A7: **Closed-loop drift and object progress.** Panel (a) plots mean remaining true-position graph hops per decision on PointMaze-Medium at d=200, where finished episodes retain their final value and dashed curves show the alternate search horizon per target. Panel (b) tracks the share of decisions ending closer to the goal against remaining distance. Panel (c) evaluates Push-T, showing how often state pairs are ordered consistently with true object distance defined as block position plus 100 px/rad, binned by object distance with 95% intervals over 289 goals and ties scored as $\frac{1}{2}$.

Proposition 1 (No drift vs. negative drift)**.** *Let $(h_t)_{t=0}^{T}$ be a non-negative process ($h_t \geq 0$) with $h_0 = D$ and bounded increments $|h_{t+1} - h_t| \leq c$ for all $t < T$.*

(a) No drift. *Let $0 \leq r < D$. If $\mathbb{E}[h_{t+1} - h_t \mid x_{0:t}] \geq 0$ whenever $h_t > r$, then*

$$\Pr\Big(\min_{0\leq t\leq T} h_t \leq r\Big) \;\leq\; \exp\Big(-\frac{(D-r)^2}{2Tc^2}\Big).$$

(b) Drift. *If $\mu > 0$, $\mu T > D$, and $\mathbb{E}[h_{t+1} - h_t \mid x_{0:t}] \leq -\mu$ whenever $h_t > 0$, then*

$$\Pr\Big(\min_{0\leq t\leq T} h_t > 0\Big) \;\leq\; \exp\Big(-\frac{(\mu T - D)^2}{2Tc^2}\Big).$$

Proof. For part (a), define the stopping time $\tau = \inf\{t \geq 0 : h_t \leq r\}$ and the stopped process $M_t = D - h_{t\wedge\tau}$. Then $M_0 = 0$, the conditional increment satisfies $\mathbb{E}[M_{t+1} - M_t \mid x_{0:t}] \leq 0$ for every t and equals zero after τ, and $|M_{t+1} - M_t| \leq c$ gives a predictable interval of width $2c$. Because the event $\{\min_{t\leq T} h_t \leq r\}$ coincides with $\{M_T \geq D - r\}$, applying Azuma–Hoeffding at time T yields the bound. For part (b), let $\tau = \inf\{t \geq 0 : h_t \leq 0\}$ and $Y_t = h_{t\wedge\tau} + \mu(t \wedge \tau)$. Conditioned on $x_{0:t}$, the increment $Y_{t+1} - Y_t$ has non-positive expectation and lies inside $[-c + \mu, c + \mu]$ of width $2c$ before τ and at 0 after τ, with $\mu \leq c$ following automatically from $|h_{t+1} - h_t| \leq c$. On the failure event $\{\min_{t\leq T} h_t > 0\}$ we have $\tau > T$ and $h_T > 0$, so $Y_T - Y_0 = h_T + \mu T - D > \mu T - D > 0$. Applying Azuma–Hoeffding at time T completes the proof. □

Scaling intuition. To picture what Proposition 1 means in closed loop, suppose each five-block decision travels s hops and ends closer to the goal with probability p. Here p measures closed-loop decision accuracy, which is separate from the offline one-hop score $\rho(h)$. The expected hop change per decision is $\mathbb{E}[h_{t+1} - h_t] = (1 - 2p)\, s$. When the planner merely moves at chance with $p = \frac{1}{2}$, case (a) applies and reaching a goal at distance D requires $\Omega(D^2)$ decisions. Entering the linear $O(D)$ regime of case (b) requires $p > \frac{1}{2} + \frac{D}{2sT}$, which equals roughly 0.71 on PointMaze-Medium at d=200 where D=34, T=16, and s=5. These bounds explain the shift from quadratic wandering to linear progress rather than yielding tight constants, and they do not model contact execution such as initiating a grasp on Cube.

Table A4: **Closed-loop drift on Two-Room and Push-T.** Each decision runs for 25 steps. Closer reports the percentage of decisions ending closer to the goal in true-state reference hops with 95% confidence intervals over episodes, and drift reports the mean hop change per decision.

Planner	Succ. (%)	Decisions	Closer (%)	Drift
Two-Room, d=75				
LeWM	32	705	33 [29, 37]	−0.08
LeWM + min-over-plan	61	533	46 [41, 51]	−0.60
Graph Anchors	**94**	474	**74** [71, 78]	**−1.54**
Push-T, d=150				
LeWM	4	1106	31 [28, 34]	−0.03
LeWM + min-over-plan	6	1050	37 [34, 40]	−0.14
Graph Anchors	31	979	53 [50, 56]	−0.67
Graph Anchors (λ=3)	**46**	897	**54** [51, 58]	**−0.77**

Verification on real rollouts. Figure A7a and Figure A7b check these conditions directly on traced PointMaze-Medium rollouts at d=200 across three seeds of 50 pairs each, measured in true-position reference hops. Whenever the agent sits more than 10 hops from the goal, three sharp differences stand out:

- *Closer decisions.* Goal-aimed LeWM planners end only 0.42–0.51 of decisions closer to the goal, whereas Graph Anchors end 0.67–0.83 closer.
- *Mean drift.* Graph Anchors hold a steady negative drift between −2.2 and −3.7 hops per decision across all distance bins. LeWM sits at zero drift through the maze interior, averaging −0.05 hops at 21–30 hops and +0.19 at 16–20 hops. Its slight −0.68 mean beyond 30 hops comes from bouncing off outer maze corners rather than directed navigation.
- *Stalls and recovery.* Immediately after a non-improving decision, Graph Anchors still drift at −1.3 hops while LeWM stays flat at +0.02. Overall, anchors stall far less often, recording 199 stalls compared to 673 for LeWM, and recover far more frequently at 92% versus 56%.

Table A4 repeats this drift measurement on Two-Room at d=75 across three seeds and on Push-T at d=150 across two seeds, building reference graphs over true agent and block states. Once again, goal-aimed planning ends closer only 0.33–0.46 of the time on Two-Room and 0.31–0.37 on Push-T, showing almost zero net drift. Graph Anchors raise those shares to 0.74 and 0.53 with strong negative drift. On Push-T, measuring graph-free block-pose error confirms the exact same ordering. The share of decisions that reduce true block error is 28% for LeWM and 31% for LeWM with min-over-plan, jumping to 46% for Graph Anchors and 47% with λ=3. Across episodes, the corresponding 95% confidence intervals are 24–31%, 28–35%, 41–50%, and 43–51%.

It is worth noting that Lemma 1 is a sufficient condition rather than a literal picture of every decision. In practice, an executed move often stops partway before reaching its active anchor. Measured in straight-line node steps of one median five-step move, PointMaze-Medium anchors begin 4.4 node steps away and come within 2.1 at closest approach. Across decisions, the agent reaches within 1 and 2 node steps on 21% and 49% of moves on PointMaze-Medium, 60% and 78% on Two-Room, and 24% and 55% on Push-T. Reference-graph hops make this approach look slightly smaller because fine-grained reference edges are shorter than a full recorded move. For example, on Two-Room the median closest approach is 0.7 node steps compared to 5 reference hops, and on PointMaze-Medium

the agent ends closer to its anchor on 60–65% of decisions in straight-line distance versus 23–26% in reference hops. What Proposition 1 actually requires, however, is steady net drift toward the final goal, which Figure A7a, Figure A7b, and Table A4 verify directly.

Object progress and failure prediction. Even though graph nodes store whole-frame latents with no object factorization, the latent space still represents the manipulated object cleanly at short range. Linear probes on held-out Push-T episodes recover block position with R^2=0.98 and block angle with R^2=0.92. Likewise, cross-episode neighbour edges shift the block by only 1.2 px at the median, compared to 120 px for random frame pairs. Over 289 held-out Push-T goals shown in Figure A7c, shortest-path graph hops order state pairs by true object distance accurately 0.68 of the time, compared to 0.54 for direct latent distance. Graph hops stay ahead across every distance bin, scoring 0.78 near the goal, 0.55 at mid-range, and 0.53 far away, against 0.80, 0.50, and 0.53 for latent distance.

Finally, because the graph-distance test runs completely offline before planning begins, it predicts closed-loop difficulty across environments in advance. As shown in Figure 9 and Table 4, goal graph distance correlates strongly with LeWM success at Spearman -0.90 across 20 settings, and it also predicts where anchors gain the most over min-over-plan at Spearman 0.82 across 19 settings.

B. Graph Anchors: Algorithm and Implementation

Graph Anchors turns the diagnosis of Section 4.1 and Section A into a simple planning method without changing or retraining the world model. As detailed in Algorithm A1, the method builds a sparse latent graph once from training trajectories, selects an anchor a few moves ahead along the shortest path to the goal at each replanning step, and scores candidate CEM rollouts by their closest approach to that nearby anchor.

B.1. Algorithm and Walkthrough

Building the latent graph. We encode every fifth frame, corresponding to one action block, of every training episode with the frozen encoder and link consecutive frames by directed temporal edges as shown in Figure A8a. To stitch routes across different trajectories, we add bidirectional neighbour edges from each node to its k=10 nearest latent neighbours whose squared latent distance falls below τ. As illustrated in Figure A8b, τ is set automatically to the empirical median squared latent length of a single recorded action block. Because τ comes directly from the model's own typical one-step jump size, no environment-specific distance threshold needs to be tuned. On PointMaze-Medium this builds 199k nodes, 194k temporal edges, and 2.0 M neighbour edges. Figure A8c and Figure A8d show that 99.9% of neighbour edges connect distinct episodes at physical separations far smaller than one five-step move. Test episodes are always removed before evaluation so no anchor can ever replay a test trajectory.

Selecting and following anchors. At each replanning step, the current latent z_t and goal latent z_g map to their nearest graph nodes s and g via a single GPU matrix product. One reversed Dijkstra search from g is run once and cached for the entire episode. Each replanning decision therefore only walks K successors forward along the shortest path $\pi(s, g)$, falling back directly to z_g once the goal is within K moves or unreachable, as written in Part 2 of Algorithm A1. Figure A9 traces this closed loop on real PointMaze-Medium episodes at d=200 on seed 0. Anchors stay six hops ahead around walls, turning a difficult 34-hop navigation problem into a sequence of reliable 6-hop subproblems.

Algorithm A1 Graph Anchors (complete specification). Part 1 runs once per environment; Parts 2–3 run at every replanning step.

Input: training episodes $\mathcal{D}$, frozen encoder f and predictor p, goal latent z_g; k=10 neighbours, span K (nav: 6; contact: 2), switch cost λ (nav: 1; contact: 3), horizon H (nav: 10; contact: 2)

Part 1: Build sparse graph (once)

$\mathcal{V} \leftarrow$ every fifth frame of every episode; $z_i \leftarrow f(o_i)$ ▷ nodes

$\mathcal{E}_{\mathrm{T}} \leftarrow \{(i,j) : e(j){=}e(i),\ t(j){=}t(i){+}5\}$ ▷ temporal

$\tau \leftarrow \mathrm{median}_{(i,j)\in\mathcal{E}_{\mathrm{T}}} \|z_i - z_j\|^2$ ▷ 1-block median

$\mathcal{E}_{\mathrm{N}} \leftarrow \{(i,j),(j,i) : j \in \mathrm{NN}_k(i),\ \|z_i - z_j\|^2 < \tau\}$ ▷ k-NN

remove test episodes; $\mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E}_{\mathrm{T}} \cup \mathcal{E}_{\mathrm{N}})$

Part 2: Select anchor (current latent z_t)

$s \leftarrow \arg\min_i \|z_i - z_t\|$; $g \leftarrow \arg\min_i \|z_i - z_g\|$

Dijkstra from g on reversed $\mathcal{G}$, cached per goal (temporal cost 1, neighbour cost λ)

if g is reachable from s and $> K$ moves away **then**

  $z_a \leftarrow$ node K moves along $\pi(s,g)$ ▷ λ>1: temporal only

else

  $z_a \leftarrow z_g$

Part 3: Plan via unchanged CEM

for 30 iterations **do**

  sample 300 sequences of H action blocks; roll out $\hat{z}^{(n)}_{1:H}$ with p

  score $d_a^{(n)} = \min_h \|\hat{z}_h^{(n)} - z_a\|^2$ ▷ closest approach, Equation (8)

  refit Gaussian to the 30 lowest-scoring sequences

execute best plan (nav: 5 blocks; contact: to closest approach), replan

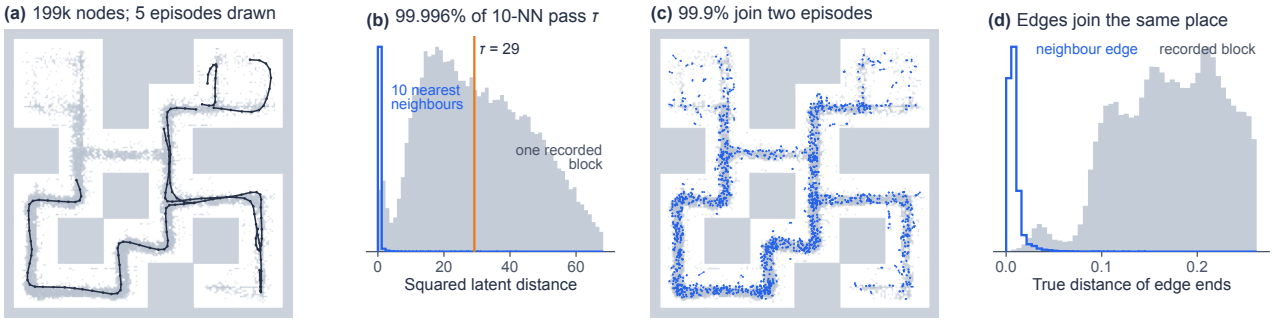

Figure A8: **Building the latent graph** on real PointMaze-Medium data, where grey dots mark node positions. Panel (a) encodes every fifth training frame as a node and links consecutive frames within an episode by directed temporal edges. Panel (b) sets the threshold τ to the median squared latent distance of one recorded block in grey, so each node's 10 nearest latent neighbours in blue fall far below τ. Panel (c) draws 1,500 neighbour edges at their true positions, showing that 99.9% bridge distinct episodes and enable route stitching. Panel (d) confirms that neighbour edges connect states physically much closer than one five-step move.

Closest-approach scoring and goal handoff. CEM keeps LeWM's exact optimizer with 300 samples, 30 elites, and 30 iterations, changing only the target latent, the scored step, and the horizon. Rather than scoring only the final step of a rollout, Figure A10a scores each rollout by its minimum distance to z_a across steps $h \in \{2,\ldots,H{+}1\}$. Intuitively, this encourages the agent to pass smoothly *through* the stepping stone instead of braking to a stop at it. In navigation we plan over H=10 blocks and execute 5 blocks before replanning. In the contact setting we use tighter anchors K=2, switch cost λ=3, and

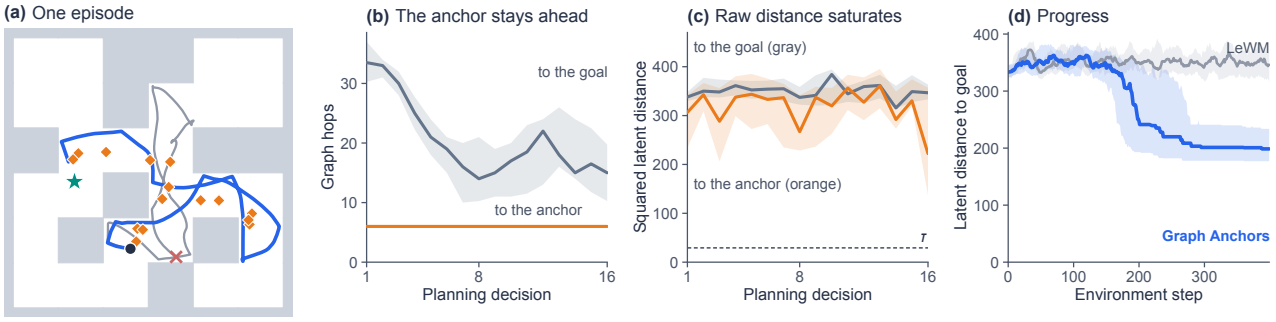

Figure A9: **Anchors in closed loop** on PointMaze-Medium at d=200 over 50 identical start and goal pairs, showing medians and interquartile ranges. Panel (a) plots the longest-path episode solved by Graph Anchors in blue with anchor diamonds and missed by LeWM in grey with its final cross. Panel (b) compares remaining graph hops to the goal in grey against hops to the active anchor in orange, illustrating that every planning step stays a 6-hop problem. Panel (c) shows that raw squared latent distance flattens out beyond a few hops, confirming that local *ranking* from Figure 2a and Table A2, rather than raw magnitude, drives progress. Panel (d) shows latent distance to the goal falling steadily with anchors while stagnating without them.

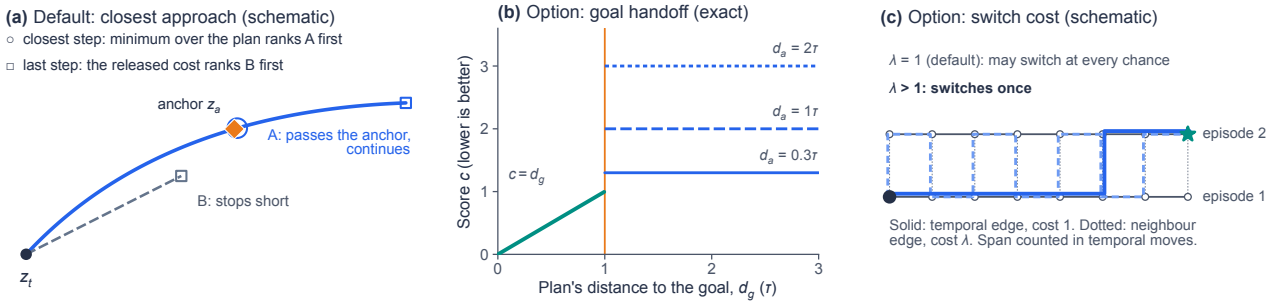

Figure A10: **Scoring plans and preserving trajectory continuity.** Panel (a) illustrates closest-approach scoring from Equation (8), which favors a rollout passing through the anchor toward the goal in A over one stopping at the anchor in B. Panel (b) shows optional gated goal handoff from Section C.2, where any plan coming within τ of z_g once $d_G(s,g) \leq 12$ is scored by d_g and beats every anchor-only plan scored by $d_a + \tau$. Panel (c) shows how charging an episode-switch cost $\lambda > 1$ on neighbour edges keeps shortest paths on continuous recorded transitions.

horizon H=2, executing up to the block of closest approach. Figure A10b also illustrates an optional gated goal handoff evaluated in Section C.2. Whenever $d_G(s,g) \leq 12$, any plan entering within τ of z_g is scored directly by d_g, while all other plans are scored by $d_a + \tau$.

B.2. Implementation Details and Graph Quality

Evaluation setup and checkpoints. All evaluations use the released `stable-worldmodel` framework [25] with 224×224 images, 5-step action blocks, a step budget of $2d$, and each environment's native success criteria. Every random seed fixes the exact same 50 start and goal pairs and identical CEM noise across methods. Push-T, Two-Room, Cube, and Reacher use the released LeWM checkpoints and datasets, while PointMaze-Medium uses the public D4RL LeWM checkpoint of [19] verified for safe class loading. Because PointMaze-Large has no released checkpoint, we train LeWM with its

Table A5: **Latent graphs across environments.** Node counts, temporal edges, neighbour edges, the automatic 1-block median threshold τ, and fp16 node storage in MB for 192-D latents.

Environment	Nodes	Temporal	Neighbour	τ	MB
Push-T	460k	0.44M	6.21M	25.1	169
Two-Room	181k	0.17M	2.10M	295.7	66
PointMaze-M	199k	0.19M	2.32M	29.2	73
OGBench-Cube	400k	0.39M	4.81M	26.9	146
Reacher	400k	0.39M	4.57M	76.2	146

default script on OGBench `pointmaze-large-navigate` for 10 epochs of 3,000 batches on one GPU over 1,000 episodes of 1,000 steps. As summarized in Table A5, this yields 200 k nodes, 199 k temporal edges, and 2.3 M neighbour edges. For the unseen-dataset graph in Figure 8, we render OGBench `pointmaze-large-stitch` with the exact same environment class, verify pixel and encoding agreement against stored training frames, and encode those frames with the frozen model.

Hyperparameters, timing, and offline diagnostics. All environments share identical graph construction rules with k=10 neighbours, 5-step nodes, and the empirical 1-block median threshold τ. On PointMaze-Medium, building the graph takes 51 s once, and one CPU Dijkstra search takes 84 to 93 ms across three timed runs. In interleaved timed runs at d=100 across 10 pairs and 3 repeats, one planning decision averages 0.8 to 1.5 s for LeWM, 2.6 s for LeWM with min-over-plan, and 1.8 s for Graph Anchors. Because solved episodes finish early, wall-clock episode time mixes solve rate and compute cost, so exact per-step GPU overhead is isolated separately in Table A17b. For offline diagnostics, Figure 2a and Figure A3 rebuild identical CPU graphs from stored node latents, testing 300 goals with up to 4 states per hop up to h=40 and 200 pairs per distance in Figure A3.

Edge fidelity and Push-T route fragmentation. To verify that neighbour edges do not create illegal shortcuts across walls or teleports, we count how many neighbour edges span a larger physical distance than the 95th percentile of a real five-step move. These false-shortcut rates are tiny on every benchmark, measuring 0.0% on PointMaze-Medium, 0.0% on Two-Room, and 0.2% on Push-T, while more than 94% of neighbour edges bridge distinct episodes. Checking all 994,834 unique neighbour edges on PointMaze-Medium finds zero wall crossings, as reported in Table A17b.

On Push-T, however, accurate local edges alone are not enough. When neighbour edges carry the same weight λ=1 as temporal edges, unweighted shortest paths jump repeatedly between different recorded episodes. Across full d=150 routes in Table A17a, 53.4% of edges are cross-episode switches rather than recorded temporal transitions, and that share rises to 64% inside the first six hops where anchors are selected, averaging 8.2 switches per 6-hop anchor over 400 paths. Jumping between demos mid-push breaks continuous block contact. Raising the neighbour edge weight to λ=3, illustrated in Figure A10c, cuts switches per anchor to 4.6, and down to 3.1 at λ=10, while keeping every node reachable. This observation directly motivates the contact setting evaluated in Section C.2.

C. Extended Results and Ablations

C.1. All Distances, Retraining Sensitivity, and Paired Tests

Table A6 reports every environment and goal distance for all world-model planners and offline policies, with every individual seed run recorded in Table A18.

Table A6: **Success (%) at every goal distance** for world-model planners and offline policies on identical pairs and 2d budgets, with stitched budgets of 200 on Two-Room and 400 on the mazes. Entries report mean ± std over 50-pair evaluations, where † marks a single evaluation and bold marks the best score per column. Exact paired McNemar tests appear in Section C.1 and the full per-evaluation results appear in Table A18.

Push-T	d=25	d=50	d=75	d=100	d=150
LeWM [25]	89 ±1	41 ±13	21 ±4	17 ±1	5 ±4
LeWM + min-over-plan	**91** ±3	55 ±10	33 ±7	17 ±7	9 ±1
GCIVL [33]	2 ±3	2 ±0	0 ±0	0 ±0	0 ±0
GCIQL [33]	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0
HILP [32]	4 ±3	0 ±0	0 ±0	0 ±0	0 ±0
Graph Anchors (ours)	82 ±6	**65** ±1	**50** ±0	**40** ±3	**30** ±3

Two-Room	d=25	d=50	d=75	stitched
LeWM [25]	88 ±4	54 ±5	31 ±3	61 ±6
LeWM + min-over-plan	**99** ±1	81 ±5	62 ±10	82 ±3
GCIVL [33]	**99** ±2	90 ±0	81 ±2	86 ±6
GCIQL [33]	61 ±1	51 ±11	37 ±4	46 ±0
HILP [32]	98 ±0	**95** ±1	87 ±3	89 ±1
Graph Anchors (ours)	96 ±4	93 ±1	**93** ±1	**99** ±1

PointMaze-Medium	d=50	d=100	d=150	d=200	stitched
LeWM [25]	50 ±12	25 ±4	13 ±3	13 ±2	39 ±10
LeWM + min-over-plan	70 ±5	33 ±6	17 ±4	12 ±5	41 ±4
GCIVL [33]	30†	4†	2†	0†	16 ±0
GCIQL [33]	34 ±5	4 ±2	0 ±0	1 ±1	12 ±3
HILP [32]	32 ±4	3 ±2	1 ±1	1 ±1	18 ±0
Graph Anchors (ours)	**91** ±2	**89** ±4	**88** ±6	**85** ±6	**95** ±4

PointMaze-Large	d=50	d=100	d=200	d=400	stitched
LeWM [25]	14 ±3	15 ±7	13 ±4	8 ±0	5 ±1
LeWM + min-over-plan	44 ±8	28 ±3	23 ±4	26 ±6	17 ±1
GCIVL [33]	9 ±1	8 ±3	9 ±1	1 ±1	3 ±1
GCIQL [33]	9 ±1	8 ±3	9 ±1	3 ±1	3 ±1
HILP [32]	12 ±0	13 ±4	16 ±3	9 ±1	10 ±6
Graph Anchors (ours)	**62** ±6	**63** ±7	**73** ±4	**87** ±1	**66** ±6

Retraining sensitivity on PointMaze-Large. Because our original PointMaze-Large LeWM checkpoint from Table 1 was no longer stored on disk, we trained a second model from scratch using the exact same code, data, and hyperparameters. As shown in Table A7, absolute success shifts modestly between training runs. For Graph Anchors in the navigation setting, d=400 success moves from 87% to 76% and stitched success moves from 66% to 45%. Importantly, the relative separation stays wide, with Graph Anchors remaining far ahead of LeWM, LeWM with min-over-plan, and every training-free alternative from Section 5.5 at every distance. The PointMaze-Large contact setting in Table 1 ran on this second

Table A7: **Two training runs of PointMaze-Large LeWM.** Success (%) averaged over seeds 0 and 42 with 50 pairs per seed, and over seeds 0, 42, and 1 for the contact setting, using identical code, data, and test pairs.

Method	d=50	100	200	400	stitched
Original model (Table 1)					
LeWM	14	15	13	8	5
LeWM + min-over-plan	44	28	23	26	17
AP (our re-implementation)	17	11	16	12	6
Graph Anchors, navigation setting	62	63	73	87	66
Second training run					
LeWM	14	11	10	4	8
LeWM + min-over-plan	38	25	24	20	21
AP (our re-implementation)	31	16	17	18	9
Graph Anchors, navigation setting	47	43	58	76	45

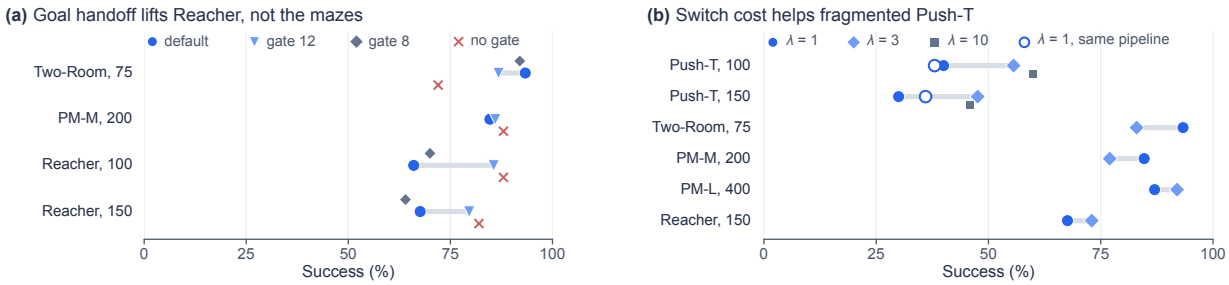

Figure A11: **Options of Graph Anchors** measured by mean success (%) across seeds. Panel (a) tests goal handoff from Figure A10b gated at 8 or 12 graph hops or left ungated. Gated handoff removes anchor detours on short-range Reacher, whereas ungated handoff misfires through the wall on Two-Room. Panel (b) compares episode-switch cost λ=3 against the default at the longest distance. Open rings show the Push-T default re-run in the switch-cost pipeline at 38 and 36 for an exact like-for-like check against the older-pipeline default of 40 and 30.

checkpoint. As expected on a 400-step maze, taking small 2-move anchors exhausts the step budget before reaching the goal, scoring 13% at d=400 and 9% on stitched pairs.

Exact paired statistical tests. Because every seed fixes the exact same 50 start and goal pairs across methods, we can test gains cleanly on paired episodes. For each pair of methods, we pool all shared seeds and apply an exact two-sided McNemar test to the episodes where one method succeeds and the other fails. Against both LeWM and LeWM with min-over-plan, every improvement of Graph Anchors at $d \geq 50$ and on stitched goals is statistically significant at $p < 10^{-2}$, and most reach $p < 10^{-6}$. The sole non-significant cell is Two-Room at d=25 versus min-over-plan, where both methods are already near ceiling. Against GCBC, gains at d=200 and on stitched goals are significant while d=100 is not. Likewise, building the graph from random play does not differ significantly from the expert graph, and feeding held graph waypoints to GCBC makes it significantly worse at d=200.



Figure A12: **Push-T in the contact setting** on episode 21 at d=150 and seed 0 with identical start and goal. The green T marks the goal, and the blue outline marks the active anchor two moves ahead on a recorded push. LeWM leaves the block off target, while Graph Anchors guide the block smoothly onto the goal by step 195 of 300, solving 29 of 50 episodes compared to 1 of 50 for LeWM.

C.2. Contact Setting, Switch Cost, and Goal Handoff

Goal handoff. When a task's goals already lie within the reliable range of the visual cost, stopping at intermediate anchors adds unnecessary detours. On Reacher, Figure A3 showed that graph distances stay at or below 10 hops even at d=150. Enabling our goal handoff from Figure A10b, which scores any rollout coming within τ of z_g directly by d_g and all others by $d_a + \tau$, eliminates those detours. As shown in Figure A11a, Reacher success rises from 66% to 86% at d=100 and from 68% to 80% at d=150, outperforming min-over-plan by 12 and 2 points across five runs per setting. On Cube at d=150 across two cluster seeds, handoff changes little, shifting from 29% to 31% compared to 19% for LeWM and 24% for min-over-plan, because failing to initiate a grasp dominates every planner.

Importantly, handoff must be gated by graph distance so it cannot fire prematurely through walls. On Two-Room at d=75 on seed 0, leaving handoff ungated causes the visual cost to latch onto goals behind the dividing wall and drops success to 72%. Gating handoff at 12 graph hops restores Two-Room to 87% and preserves 86% on PointMaze-Medium at d=200. Tightening the gate to 8 hops still preserves Two-Room at 92%, but lowers Reacher to 70% at d=100 and 64% at d=150. Because handoff fires on nearly every step once the gate opens, we present it as an optional short-range switch rather than altering the default method.

Episode-switch cost and contact configurations. On Push-T, charging λ=3 on cross-episode neighbour edges and counting span strictly in temporal moves keeps routes on continuous recorded pushes, as visualized in Figure A12. Even with the navigation horizon H=10, adding λ=3 lifts d=150 success from 30% to 48% in Figure A11b. Combining λ=3 with tighter anchors K=2, horizon H=2, and closest-approach execution in our full contact setting lifts Push-T d=150 all the way to 71% in Table A8. Both ingredients matter together. Placing the anchor closer at K=3 without λ=3 stays stuck at 30%, whereas adding λ=3 jumps to 72% across three seeds on the exact same machine and pairs used for AP. On Cube, gains remain modest from 26% to 39%, staying below AP's 53%, because Cube episodes require initiating a grasp rather than maintaining a continuous push.

Table A8: **Contact-setting configurations at d=150.** Success (%) averaged across seeds 0, 42, and 1 on the same machine and identical pairs, where † marks seed 0 only.

Configuration	Push-T	Cube
Graph Anchors, K=6 (default)	30	26
K=3, closest-approach execution	30	38†
K=3 + switch cost	**72**	26†
K=2 + switch cost (contact setting)	71	**39**
AP [23] (same machine)	65	53

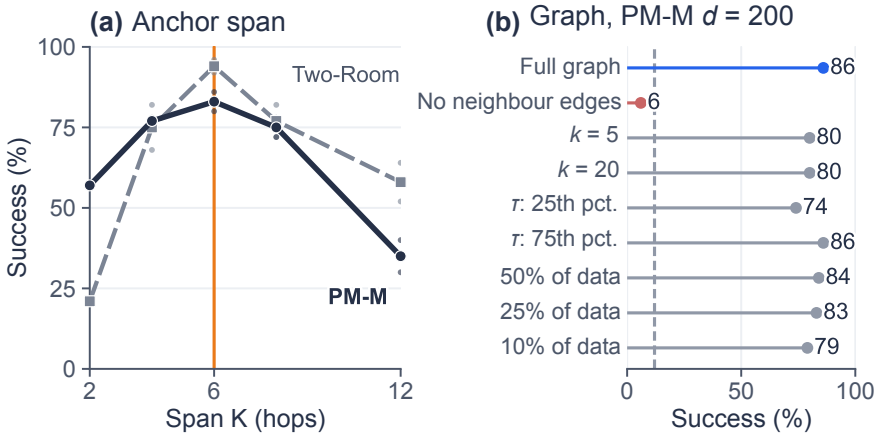

Figure A13: **Anchor span and graph construction ablations.** Panel (a) sweeps anchor span K across environments, averaged over seeds 0 and 42. Panel (b) ablates graph construction on PointMaze-Medium at d=200 on seed 0, with the no-neighbour-edges and data-fraction points averaged over seeds 0 and 42. The dashed line marks LeWM with 3× CEM samples.

C.3. Ablations, Scaling, and Robustness Stress Tests

Target vs. search and stronger CEM. Table A9a separates the benefit of changing the target from the benefit of changing the search rule. On PointMaze-Medium, changing the target accounts for nearly the entire gain. Aiming LeWM's unchanged search at our anchors lifts success from 12% to 85%, whereas aiming our closest-approach search at the distant goal stays at 9%. On Two-Room, both changes contribute, lifting success from 66% to 94%. Scaling up search toward the goal cannot substitute for anchors either. In Table A9b, doubling the horizon to H=20 and raising CEM samples from 300 to 1,000 leaves goal-aimed planning at 28% on PointMaze-Medium and 77% on Two-Room, far behind default Graph Anchors in the same runs at 89% and 94%.

Span, horizon, and execution length. Figure A13a and Table A10a confirm that anchor span K and planning horizon H couple just as the range diagnosis predicts. Longer imagined rollouts can chase farther stepping stones, so the optimal span shifts from K=4 at H=5 to K=6 at H=10 and H=15 on both mazes. Across the entire table, our default K=6 stays within 6 points of the optimum. Placing anchors far out at K=12 overwhelms a short 5-block plan on PointMaze-Medium at 11%, but recovers to 65% once the horizon stretches to H=15. Conversely, overly close anchors at K=2 stall under fixed 5-block execution, dropping from 66% at H=5 to 45% at H=15 on PointMaze-Medium and to just 8%

Table A9: **Separating target choice from search strength.** Entries report success (%). Panel (a) runs a 2×2 factorial crossing target and search rule as in Figure 7a over seeds 0, 42, and 1 on a separate machine. Panel (b) scales planning horizon H and CEM samples N while aiming at the distant goal, compared against default Graph Anchors in the same runs over seeds 0 and 42.

(a) Target crossed with search

Target	Search	PM-M, 200	Two-Room, 75
goal	H=5, last step	12	32
goal	H=10, min over plan	9	61
anchor	H=5, last step	85	66
anchor	H=10, min over plan	87	94

(b) Stronger search aimed at the goal

Planner	H	N	PM-M, 200	Two-Room, 75
LeWM + min-over-plan	10	300	6	58
LeWM + min-over-plan	10	1000	14	56
LeWM + min-over-plan	20	300	22	76
LeWM + min-over-plan	20	1000	28	77
Graph Anchors	10	300	**89**	**94**

Table A10: **Span–horizon coupling and execution length for Graph Anchors.** Entries report success (%). Panel (a) crosses anchor span K with planning horizon H averaged over seeds 0 and 42, with bold marking the best span per row. Panel (b) varies plan length and execution rule across span K on seed 0, while the default plan-10 execute-5 row appears in panel (a). The closest-approach rows for $K\in\{1,2,6\}$ and the 10-block K=1 run were evaluated on a second machine.

(a) Span K crossed with horizon H

	K=2	4	6	8	12
PointMaze-Medium, d=200					
H=5	66	**92**	88	60	11
H=10	57	77	**89**	75	35
H=15	45	82	**87**	71	65
Two-Room, d=75					
H=5	24	**86**	80	52	39
H=10	21	75	**94**	77	58
H=15	11	78	**95**	85	66

(b) Plan length and execution rule

	K=1	2	3	4	6
PointMaze-Medium, d=200					
plan K, execute K	48	92	92	88	86
plan K, execute to the closest approach	48	92	92	88	94
plan 10, execute to the closest approach	22	60	72	82	80
Two-Room, d=75					
plan K, execute K	4	76	82	96	86
plan K, execute to the closest approach	4	92	100	100	94
plan 10, execute to the closest approach	10	92	100	100	92

on Push-T at d=50.

Table A10b isolates how execution length solves this short-span stall. Matching plan length to K blocks and stopping execution right at closest approach lets short anchors succeed cleanly, reaching 92 to 94% on PointMaze-Medium for $K\in\{2,3,6\}$ and 100% on Two-Room for $K\in\{3,4\}$. That finding directly motivates the short-horizon closest-approach execution in our contact setting with K=2 and H=2. By contrast, cutting off a 10-block plan early at closest approach hurts PointMaze-Medium because min-over-plan gives CEM no reason to reach the anchor early in a long rollout. Finally, the graph ablation in Figure A13b shows that neighbour edges are essential. Dropping neighbour edges collapses PointMaze-Medium from 86% down to 6%, while varying k across 5, 10, and 20 changes success very little.

Stitching, limited data, and displacement. Table A11 fills in the full numbers behind Figure 8, covering training-data subsampling, larger stitched budgets, mid-episode kidnapping, and Reacher.

Table A11: **Robustness stress tests for Graph Anchors.** This table supplements Figure 8 with the complete training-data subsampling curve, larger stitched budgets, mid-episode kidnapping, and Reacher results. Entries show means and individual runs for seeds 0 and 42, while Reacher lists d=25, 50, 100, and 150 on seed 0.

Stress test	Mean	Runs
Kidnapped mid-episode, PointMaze-M d=200	65	66/64
Kidnapped mid-episode, Two-Room d=75	74	76/72
Stitched goals, PointMaze-M, budget 400	94	98/90
Stitched goals, PointMaze-M, budget 600	99	100/98
Stitched goals, PointMaze-M, budget 800	99	100/98
Reacher, mean over d=25..150	62	60/60/64/64
Data fraction 1%, d=200 / stitched	63 / 86	74/52
Data fraction 2%, d=200 / stitched	80 / 73	86/74
Data fraction 5%, d=200 / stitched	78 / 86	82/74
Data fraction 10%, d=200	79	80/78
Data fraction 25%, d=200	83	76/90
Data fraction 50%, d=200	84	86/82
Data fraction 100%, d=200 / stitched	86 / 94	92/80

Because 99.9% of neighbour edges connect distinct episodes as shown in Figure A8c, the graph assembles valid routes even under extreme data scarcity or shift. Keeping just 1% of training episodes still solves 63% of PointMaze-Medium d=200 goals, climbing steadily to 86% with the full dataset. Pure random-play data reaches 90% on PointMaze-Medium, and encoding an unseen dataset never used during world-model training still reaches 87% on PointMaze-Large at d=400. Mid-episode displacement needs no extra recovery mechanism either. Whenever the agent is abruptly moved, the next replanning step simply re-anchors from the new observation, shifting only from 86% to 65% on PointMaze-Medium and from 93% to 74% on Two-Room.

D. Additional Baselines and Planner Details

D.1. World-Model Baselines and Graph-Guided Policies (GAS, TTGS)

World-model and offline-policy baselines. We train PLDM [38], DINO-WM [45], GCBC, GCIVL, GCIQL [33], and HILP [32] inside the exact same `stable-worldmodel` codebase on identical datasets, test pairs, and budgets. All offline policies act in closed loop from frozen DINOv2 features plus proprioceptive goal state, trained for one epoch of up to 5,000 batches of size 128 per phase. PLDM trains for one epoch of 10,000 batches and plans with LeWM's CEM setup. DINO-WM trains for five epochs of 5,000 batches and plans directly over patch tokens. Because DINO-WM takes 37 to 68 s per decision on an H200 GPU, more than 20× slower than LeWM, we evaluate it on 20 seed-0 pairs at long distances.

Because one epoch is much shorter than typical 100-epoch policy defaults, these policy rows act as matched-budget references. Even so, extending training to four or five epochs on PointMaze-Medium in Table A13b leaves GCIVL unchanged, lifts GCBC only at d=50 to 98%, and leaves five-epoch PLDM weak at d=200 at 1% alone and 7% with anchors. When PLDM already has a reliable local model on Two-Room in Table A13a, adding Graph Anchors lifts d=75 success from 50% to 97% and stitched goals from 81% to 99%. On Reacher, whose dataset is uniform random play, offline policies have no

Table A12: **DINO-WM on PointMaze-Large relies on goal coordinates.** Success (%) at d=400 for one trained model evaluated with three goal costs on 20 pairs per run. Graph Anchors uses images only across 50 pairs per seed.

Planner	Goal cost	Success (seeds)
DINO-WM	image + position (default)	100 / 95 / 100
DINO-WM	position only	95
DINO-WM	image only	70
Graph Anchors (LeWM)	image only	**87** (mean)

Table A13: **PLDM anchors and longer baseline training.** Entries report success (%). Panel (a) adds Graph Anchors to one-epoch PLDM across seeds 0 and 42, averaging two trainings on PointMaze-Medium. Panel (b) tests longer training on PointMaze-Medium, with four-epoch GCBC and GCIVL evaluated on seed 0 and on stitched seeds 0 and 42.

(a) Anchors on PLDM

	Two-Room			PointMaze-Medium		
	25	50	75	100	200	St.
PLDM planner	98	75	50	17	3	28
+ min-over-plan	100	85	62	12	4	27
+ graph anchors (ours)	95	98	97	28	4	28

(b) Longer training, PointMaze-Medium

	50	100	150	200	St.
GCBC, 1 epoch (3 trainings)	93	83	54	51	51
GCBC, 4 epochs	98	88	68	58	59
GCIVL, 1 epoch	30	4	2	0	16
GCIVL, 4 epochs	30	2	0	0	12

goal-directed demonstrations to imitate and remain at 0 to 4%, whereas dynamics models still plan effectively, with PLDM reaching 24% at d=25 and 15% at d=150.

Finally, Table A12 explains DINO-WM's strong PointMaze-Large result of 100%. Its default cost includes ground-truth proprioceptive agent coordinates alongside the image. Removing position coordinates drops image-only DINO-WM down to 70%, compared to 95% using position alone and 87% for image-only Graph Anchors.

Why graph-guided policies (GAS and TTGS) fail on visual mazes. A natural question is whether simply adding a graph on top of an offline policy already solves long-horizon visual planning. Prior methods like GAS [4], which builds a graph in HILP temporal-distance space ϕ, and TTGS [30], which builds a test-time graph from learned value distance $\log(V(1-\gamma)+1)/\log\gamma$, take exactly this approach. Examining both their graph construction in Table A14 and their closed-loop execution in Table A15 reveals *three coupled failure modes* that Graph Anchors directly resolves:

1. **Metric coupling and the sharp τ threshold jump.** Both GAS and TTGS connect nodes by thresholding a *learned* value or temporal-distance head at $d_V < \tau$, requiring per-task and per-agent τ tuning. Whenever that learned head saturates on visual mazes, it outputs nearly the same high distance for almost every pair of states, leaving no usable τ window at all. On simpler Two-Room, GCIVL predicts a healthy median distance of 2.9, so both TTGS and GAS form local edges and solve over 82% of goals in Table A14. On PointMaze-Medium and PointMaze-Large, however, GCIVL distances cluster near 24 to 25. Any threshold below that plateau, such as $\tau \leq 16$, leaves TTGS with zero value edges out of more than 142k candidates, disconnecting episodes completely. Raising τ just past the plateau flips straight to **100% of all candidates**, turning an empty graph directly into a dense graph full of false shortcuts that solves only 0% on PointMaze-Medium and

Table A14: **Value-graph saturation and collapse in TTGS and GAS**, averaged over all evaluation logs per environment and τ. On visual mazes, learned value distances saturate near 24 to 25 so TTGS jumps abruptly from 0% to 100% edges with no usable τ window, while GAS collapses to 75 nodes on PointMaze-Large at τ=16. Graph Anchors self-calibrates on 100% of frames.

Env.	Method (τ)	Nodes	Med. d_V	Value edges (%)	Far (%)	St. (%)
Two-Room (d=75)	TTGS (τ=16)	2,981	2.9	99.8%	82	87
	GAS (τ=24)	2,981	2.2	100.0%	86	86
	Ours (1-blk τ)	181k	–	0.0% jumps	**93** (100[c])	**99**
PM-M (d=200)	TTGS (τ=16/24/32)	2,946	24.6	**0.0 / 0.2 / 100%**	0	17
	GAS (τ=24/32)	2,946	4.4	100.0%	19	36
	Ours (1-blk τ)	199k	–	0.0% jumps	**85** (94[c])	**95**
PM-L (d=400)	TTGS (τ=16/24)	2,837	23.9	**0.0 / 100%**	5	3
	GAS (τ=16)	**75**	14.3	80.8%	2	3
	GAS (τ=24/32)	2,837	5.5	100.0%	2	3
	Ours (1-blk τ)	200k	–	0.13% jumps	**87** [76]	**66** [45]

5% on PointMaze-Large. GAS hits a related collapse, shrinking PointMaze-Large from 2,837 nodes down to just **75 nodes** at τ=16 and admitting every edge at $\tau \geq 24$, yet solving at most 19% on PointMaze-Medium and 2% on PointMaze-Large.
Resolution. Graph Anchors decouples the graph from any learned value head and sets τ automatically from the frozen encoder's empirical 1-block median, achieving 0.00% wall crossings on PointMaze-Medium and 0.13% long jumps on PointMaze-Large with zero tuning.

2. **Quadratic $O(N^2)$ neural distance matrices and undirected shortcuts.** TTGS evaluates $V(s_i, s_j)$ over all N^2 node pairs, forcing graphs to stay capped at 2k to 4k random frames. GAS similarly constructs a dense $N \times N$ pairwise matrix with bidirectional ϕ-edges plus forced component-merging bridges, discarding temporal directionality and contact continuity. *Resolution.* Graph Anchors indexes 100% of training frames, from 199k up to 453k nodes, via a sparse 10-NN graph combined with directed temporal edges and switch cost λ.
3. **Double bottleneck: policy execution fails even on a verified graph.** Even when we discard GAS and TTGS's broken value graphs and hand GCBC, GCIVL, and HILP waypoints from our own verified shortcut-free graph, closed-loop execution still fails. As reported in Table A15, holding each waypoint until reached caps PointMaze-Medium d=200 at 21% and PointMaze-Large d=400 at 7%, dropping GCBC by 30 points below its no-graph score. Refreshing waypoints every five policy steps on PointMaze-Medium across three seeds hurts GCBC too, pulling d=200 down from 55% alone to 43% at K=6 and 25% at K=2, since the policy reaches fewer than 1% of K=6 waypoints before refresh. Offline RL policies also require goal-directed trajectories and fail on random play, scoring 0 to 4% on Reacher.
Resolution. Pairing our sparse graph with frozen world-model CEM executes nearby visual targets through imagined physics without policy training, solving PointMaze-Medium at 85% and 94% from random play, PointMaze-Large at 87%, and Reacher at 68%. Although longer training beyond our 1-epoch budget would soften value saturation in GCIVL and HILP, training-free deployment, random-play robustness, and full-dataset sparse scaling remain structural advantages of Graph Anchors.

Table A15: **Offline policies paired with graphs on the mazes.** Success (%) using held waypoints from our verified latent graph and GAS-style subgoals taking the best $\tau \in \{2, 4, 8\}$ per cell, averaged over seeds 0, 42, and 1, with stitched goals over seeds 0 and 42. Bold marks the best score per column. Even on our verified latent graph, offline policies remain far below Graph Anchors.

	PointMaze-Medium					PointMaze-Large				
Method	50	100	150	200	st.	50	100	200	400	st.
GCBC + graph waypoints	84	70	39	21	41	17	15	10	6	4
GCIVL + graph waypoints	34	8	3	0	18	8	7	9	2	1
HILP + graph waypoints	31	3	1	0	18	8	9	8	7	4
GAS-style (HILP)	31	3	1	0	18	9	10	10	10	8
GCBC, no graph	**93**	83	54	51	51	33	25	18	18	17
HILP, no graph	32	3	1	1	18	12	13	16	9	10
Graph Anchors (ours)	91	**89**	**88**	**85**	**95**	**62**	**63**	**73**	**87**	**66**

D.2. Implementation Fidelity, Push-T Scaling, and Full Metrics for AP, NGC, Metro-WM

This section provides the full re-implementation details, reproduction check, Push-T distance scaling curve, and complete quantitative tables supporting the analysis in Section 5.5.

Shared evaluation setup and re-implementation details. We re-implemented AP [23], NGC [41], and Metro-WM [20] inside our shared evaluation framework on identical 5-step training nodes, test pairs, seeds, 2d budgets, and test-episode exclusions:

- *AP* retrieves the single-episode pair (z_s, z_{s+h}) whose standardized key $(z_s, z_{s+h}, z_{s+h} - z_s)$ best matches $(z_t, z_g, z_g - z_t)$. It then targets the next block z_{s+1} using a 1-block terminal-cost CEM plan and decrements h from the given true step distance d.
- *NGC* replaces Euclidean goal distance with shortest-path length over an undirected mutual 15-NN graph of 5,000 random training snapshots, attached via 4 nearest nodes on each side, while keeping LeWM's 5-block terminal-cost CEM.
- *Metro-WM-style* assigns temporal edges their 5-frame gap and links each node to its 4 nearest cross-episode nodes within a radius calibrated at the 0.25-quantile of intra-episode latent gaps. It selects the farthest reachable node within budget H frames and executes an h-block terminal-cost plan in full, sweeping (h, H) per task as in [20]. Our implementation uses 5-step vertex strides and falls back to z_g if disconnected.

Table A16 verifies that all three re-implementations closely match their published results. Figure A14 extends the Push-T comparison all the way to d=200, where Graph Anchors' unchanged contact setting stays flat at 71% while AP drops to 56%, Metro-WM to 27%, and NGC to 4%. Table A17 lists the complete route-continuity statistics, RTX 3090 Ti memory and cost-latency benchmarks, maze shortcut rates, PointMaze-Large 4-way scores, and AP stress tests summarized in Section 5.5.

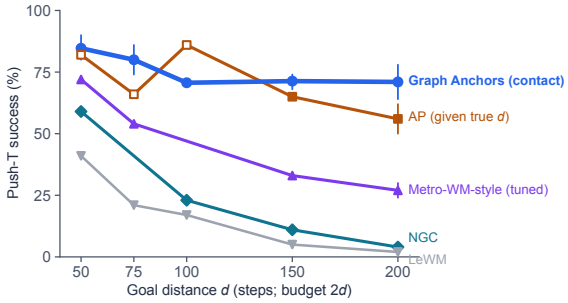

Figure A14: **Push-T success across goal distance** for all four training-free planners on identical frozen LeWM checkpoints, graph nodes, and 50-pair seeds with budget $2d$. Graph Anchors uses its single unchanged contact setting (K=2, λ=3) at every distance, AP receives the true step count d, and Metro-WM uses its best Push-T setting (3:20). Filled markers show multi-seed means with range bars; hollow markers show one seed.

Table A16: **Re-implementation reproduction check vs. published papers.** Across every overlapping setting reported in the original papers, all three re-implementations agree within 9 percentage points despite our tighter $2d$ step budget.

Setting	Reported	Ours	Runs
AP, Push-T, 50 steps	82.0	82	1
AP, Push-T, 100 steps	77.3	86	1
AP, Push-T, 140 / 150 steps	69.5	66	1
NGC, Push-T, 50 steps	63.1	59	2
NGC, Push-T, 100 steps	27.9	23	2
NGC, Two-Room, 75 steps	94.6	99	2
Metro-WM, Two-Room, 75 steps	100.0	99	3
Metro-WM, Push-T, 75 steps	62.7	54	3

Table A17: **Route quality, compute scaling, and stress tests of AP, NGC, and Metro-WM.** Panel (a) measures Push-T d=150 route continuity across 200 held-out routes on 453k nodes, where the 95th percentile of a real 1-block move is 34.5 px. Panel (b) benchmarks GPU memory and per-decision CEM cost latency on one RTX 3090 Ti over 50 environments, 300 samples, and 30 iterations, alongside maze shortcut jumps exceeding the 95th percentile of a real 5-step move, equal to 0.252 on full PointMaze-Medium, 0.250 at 1%, and 1.020 on PointMaze-Large. Panel (c) reports the PointMaze-Large 4-way comparison on `lewm_pml` seed 0 with original-checkpoint scores in brackets, together with AP stress tests where ^{c} marks the contact setting. On Reacher, NGC at 75% and LeWM with min-over-plan at 77% exceed both AP and Graph Anchors.

(a) Push-T d=150 route continuity (200 held-out routes, 453k training nodes)

Method	Recorded temporal (%)	Switches / 6 moves (med / mean)	Cross-jump >15 px (%)	Block jump med / p90 (px)	Angle jump p90	Push-T d=150 (%)
NGC (N=5,000)	0.0	all cross-ep.	65.9	23.4 / 60.3	87.5°	11
Metro-WM (H=20, 25)	45.9	6.0 / 7.03	12.1	2.8 / 17.5	10.9°	33
Graph Anchors (λ=1)	46.6	5.0 / 6.12	21.9	6.7 / 22.3	15.1°	30
Graph Anchors (λ=3, contact)	**82.0**	**1.0 / 1.68**	18.5	5.0 / 20.1	13.6°	71

(b) Memory, in-CEM cost latency & maze shortcut jumps

Method	Nodes	Mem.	Cost / dec.	PM-M full/1%	PM-L >p95 (max)
NGC (5k)	5k (2.5%)	95.4 MB	1,729.9 ms	0.41 / 1.37%	**9.51%** (22.8)
Metro (H=20)	199k (100%)	CSR	plain L2	0.00 / 1.83%	0.07% (5.05)
Metro (H=25)	199k (100%)	CSR	plain L2	0.00 / 1.88%	–
Ours (full)	**199k (100%)**	**72.9 MB**	**64.6 ms**	**0.00 / 0.00%**	**0.13%** (p95 0.11)

(c) PointMaze-Large 4-way & AP stress tests (%)

Setting	Competitor(s)	Ours	Δ
PM-L d=400 (4-way, `lewm_pml`)	NGC 24 / AP 18 / Met. 6	**76** [87]	+52/+70
PM-L stitched (4-way, `lewm_pml`)	NGC 20 / AP 9 / Met. 6	**45** [66]	+25/+39
Push-T d=150, AP wrong d (0.5/2/3×)	AP 16 / 2 / 0	**71**^{c}	+55/+71
Kidnapped PM-M, d=200	AP 22	**65**	+43
Random-data PM-M, d=200	AP 56	**94**	+38
Reacher, d=150	AP 34	**68**	+34
PM-M d=200, AP wrong d (0.5/3×)	AP 29 / 11	**85** (94^c)	+56/+83
Two-Room d=75, AP wrong d (0.25/3×)	AP 36 / 2	**93** (100^c)	+57/+98

E. Full Per-Evaluation Results, Qualitative Rollouts, and Reproducibility

Rollouts and failure analysis. Figure A15 and Figure A16 show LeWM wandering near the start while Graph Anchors steer smoothly around maze walls. Inspecting every traced PointMaze-Medium failure on seed 0, covering 4 of 50 episodes at d=200 and 1 of 50 stitched episodes, confirms that anchors stayed active with zero stalls. Specifically, 4 of 4 failed d=200 episodes came within 0.5 maze cells of an anchor and ended 1.4 to 2.2 units from the goal. Remaining failures therefore come from slow progress along valid routes rather than wrong paths or traps.

Reproducibility and video archive. Our project page hosts 29 rollout and diagnostic videos covering the range diagnosis, 3D latent PCA, side-by-side LeWM and anchor rollouts across environments, 4-way Push-T d=200 comparisons, and real-time active-anchor camera overlays (Figures A12 and A16). Repeated runs on the same machine match identically, cross-machine runs stay within a few points so every main-table comparison pairs same-machine runs, and Table A18 records every individual 50-pair evaluation behind Tables 1, A6 and A13.

Table A18: **Complete per-evaluation results** behind Tables 1, A6 and A13, excluding DINO-WM which ran one 20-pair evaluation per cell of Table 1. Each entry lists success (%) for individual 50-pair runs separated by "/", while "–" means not run. LeWM, LeWM with min-over-plan, and Graph Anchors list seeds 0, 42, and 1, except Push-T $d \geq 50$ and PointMaze-Large use seeds 0 and 42 and PointMaze-Medium stitched LeWM baselines use seeds 0 and 1. GCBC on PointMaze-Medium lists trainings A and B on seed 0 followed by training C on seeds 0, 42, and 1, with stitched runs adding seed 42 for training A. Other offline policies use seeds 0, 42, and 1 or 0 and 42. PLDM on PointMaze-Medium lists the first training on seeds 0 and 42 followed by the second training on seeds 0 and 42, with d=50, d=150, and stitched min-over-plan using the second training. Cube and Reacher use seed 0.

(a) PointMaze-Medium (stitched: budget 400)

Method	d=50	d=100	d=150	d=200	stitched
LeWM	38/50/62	24/22/30	10/12/16	16/12/12	46/32
LeWM + min	66/68/76	30/40/30	12/20/18	14/6/16	44/38
Graph Anchors	**90/90/94**	**84/92/90**	**88/94/82**	**92/80/82**	**98/90/96**
GCBC	92/94	84/80/82/86/82	50/58	52/38/54/56/54	48/46/56/52/46/60/46
GCIVL	30	4	2	0	16/16
GCIQL	32/40/30	6/4/2	0/0/0	2/0/0	14/10
HILP	28/36/32	4/4/0	0/2/0	2/0/0	18/18
PLDM	26/34	20/14/16/18	4/4	6/0/6/0	30/24/32/26
PLDM + min	50/52	12/8/16/14	2/6	8/4/2/4	26/28
PLDM + anchors	52/66	30/24/30/28	8/10	4/2/4/4	30/30/30/24

(b) Two-Room (stitched: budget 200)

Method	d=25	d=50	d=75	stitched
LeWM	88/84/92	52/50/60	28/32/34	56/68/60
LeWM + min	98/98/100	76/86/82	50/68/68	84/78/84
Graph Anchors	**92/96/100**	**92/94/94**	**94/92/94**	**100/98/98**
GCBC	100	98	100	98/100
GCIVL	100/100/96	90/90/90	84/80/80	82/90
GCIQL	62/60/62	62/40/52	38/32/40	46/46
HILP	98/98/98	96/96/94	84/86/90	88/90
PLDM	100/96	76/74	46/54	80/82
PLDM + min	100/100	88/82	62/62	–
PLDM + anchors	94/96	96/100	98/96	100/98

(c) PointMaze-Large (stitched: budget 400)

Method	d=50	d=100	d=200	d=400	stitched
LeWM	16/12	20/10	10/16	8/8	4/6
LeWM + min	50/38	30/26	20/26	30/22	18/16
Graph Anchors	**66/58**	**58/68**	**76/70**	**88/86**	**70/62**
GCBC	24/42	22/28	16/20	20/16	16/18
GCIVL	8/10	10/6	10/8	2/0	2/4
GCIQL	8/10	10/6	10/8	2/4	2/4
HILP	12/12	16/10	14/18	10/8	6/14

(d) Push-T (navigation setting, contact in Table A8)

Method	d=25	d=50	d=75	d=100	d=150
LeWM	88/90/88	32/50	24/18	18/16	2/8
LeWM + min	94/90/88	48/62	38/28	22/12	8/10
Graph Anchors	**88/82/76**	**64/66**	**50/50**	**38/42**	**28/32**
GCBC	52/28	20/12	12/4	10/12	6/4
GCIVL	4/0	2/2	0/0	0/0	0/0
GCIQL	0/0	0/0	0/0	0/0	0/0
HILP	6/2	0/0	0/0	0/0	0/0

(e) OGBench-Cube (seed 0)

Method	d=25	d=50	d=100	d=150
LeWM	70	64	50	20
LeWM + min	68	60	56	20
Graph Anchors	**70**	**56**	**60**	**20**

(f) Reacher (seed 0)

Method	d=25	d=50	d=100	d=150
LeWM	60	74	62	66
LeWM + min	82	80	78	74
Graph Anchors	**60**	**60**	**64**	**64**



Figure A15: **Rollouts on PointMaze-Medium** for seed 0 on identical start and goal pairs, comparing LeWM in grey with its final cross against Graph Anchors in blue with active anchor diamonds. Episodes follow a fixed selection rule choosing the longest successful paths solved by anchors and missed by LeWM whose starts lie at least 1.5 units apart.

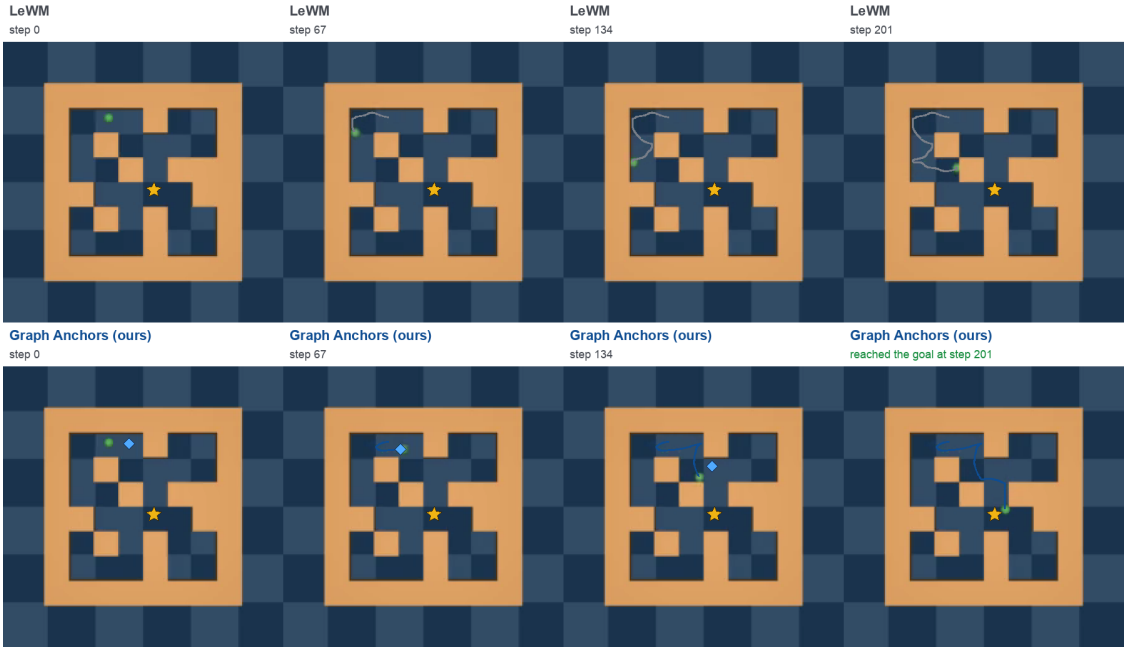

Figure A16: **Planner camera view** on PointMaze-Medium at d=200 and seed 0 across four moments on an identical start and goal pair. The star marks the goal, the diamond marks the active anchor, and the curve traces the path taken so far.