Hidden Failure Modes in Latent World-Model Planning from Offline Data

Kanpat Vesessook [*1] **Kevin Yang** [*1]

Abstract

Latent world models offer an appealing route from offline datasets to online decision-making: learn compact predictive dynamics from logged trajectories, then adapt online by planning in latent space. We revisit LeWorldModel-style joint-embedding predictive architectures under closed-loop receding-horizon MPC, where a planner optimizes horizon H but executes only a prefix $K < H$ before replanning. Across standard tasks, we find that terminal-at-H scoring can misdiagnose model quality because the score is evaluated at a time index the controller never directly executes: prefix-terminal scoring repairs PushT and TwoRoom, while running costs substantially repair the broader standard-task set. On deceptive navigation, however, aligned scalar latent costs remain insufficient: the best scalar/value-bootstrapped latent objective reaches 26.5%, while a waypoint/local-actuator interface reaches 78.8%–92.7% on held-out TwoRoom Far Door, with nearest retrieval remaining a strong control. The main lesson is that offline-to-online world-model evaluations can fail at the planning interface even when the learned representation is usable. These results suggest that offline-to-online latent planning benchmarks should report not only learned model quality, but also replanning interval, scoring time index, and controllability interface.

1. Introduction

Offline world models promise to turn fixed datasets into adaptable decision rules (Byravan et al., 2020; Hafner et al., 2023; Kidambi et al., 2020; Yu et al., 2020). Given logged trajectories, one trains a predictive model once, then performs online adaptation by replanning from the current observation. This makes model-based planning attractive in settings where new interaction is limited, costly, or unsafe, including robotics, scientific design, and healthcare. It also aligns closely with the scope of decision-making from offline datasets to online adaptation: the model is trained offline, but deployment depends on closed-loop online choices.

We study this question through LeWorldModel (LeWM), a recent joint-embedding predictive architecture that learns latent dynamics from pixels and plans by optimizing action sequences with the cross-entropy method (Maes et al., 2026; Rubinstein, 1999). LeWM is appealing because it makes the world-modeling component simple: a compact latent predictor is trained offline and reused for online goal-conditioned planning. However, the planning protocol contains a subtle but consequential choice. Let H be the imagined planning horizon and K be the number of actions executed before replanning. If $K = H$, a terminal latent cost at step H is naturally aligned with execution. If $K < H$, the planner optimizes a terminal state that it will not directly execute before feedback intervenes. We use *closed-loop* or *receding-horizon* MPC to refer to this standard execution protocol: optimize a horizon-H sequence, execute only a prefix, observe the new state, and replan.

This paper asks what closed-loop MPC actually measures in LeWM-style latent planning. Our answer is diagnostic rather than celebratory. Published-style terminal scoring can fail for a simple time-index reason, while a deceptive navigation task exposes a deeper mismatch between latent proximity and controllable progress. We use LeWM as a concrete case study, but the failure modes concern offline-trained latent models more generally when they are evaluated through online receding-horizon planning. These distinctions matter for offline-to-online evaluation because a single success number may otherwise mix objective mismatch, proposal mismatch, and actuator failure.

Contributions. We make two empirical findings and one set of protocol recommendations. First, we separate the planning horizon H from the execution interval K and show that terminal-at-H scoring is a poor objective for this closed-loop MPC regime: prefix-terminal-at-K repairs PushT and TwoRoom, while running costs largely repair the broader representative standard-task set we evaluated. Second, we

[*]Equal contribution [1]Brown University. Correspondence to: Kanpat Vesessook <kanpat_vesessook@brown.edu>.

Proceedings of the 43^{rd} International Conference on Machine Learning DEMO Workshop, Seoul, South Korea. PMLR 306, 2026.

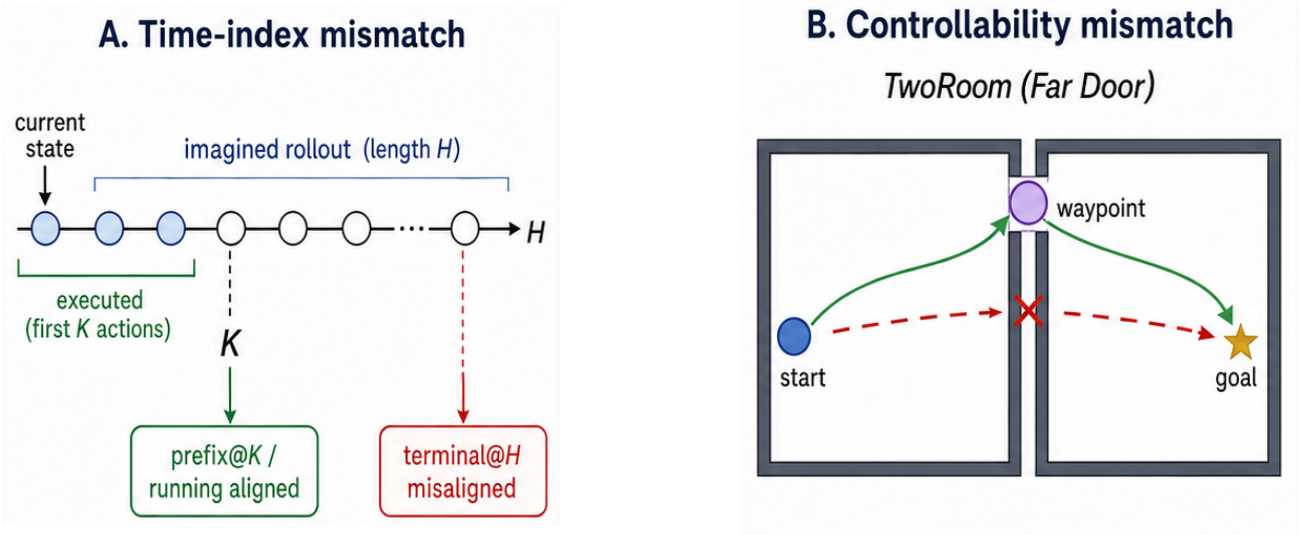

Figure 1. Overview of the two main diagnostics exposed by evaluating latent world-model planning under closed-loop receding-horizon MPC: objective time-index mismatch and latent-controllability mismatch.

introduce a deceptive TwoRoom variant that requires a controllable detour and show that scalar latent objectives are not enough; a waypoint/local-actuator interface succeeds on held-out TwoRoom Far Door, but retrieval remains a strong control. Finally, we distill these findings into reporting recommendations for offline-to-online latent planning benchmarks. We do not claim that stage costs or receding-horizon execution are new MPC machinery; our claim is that LeWM-style offline-to-online evaluations can change qualitatively when these standard distinctions are made explicit. As a protocol caveat, Appendix B.3 identifies a reset-state artifact in PushT Gate and treats it as a sanity check rather than primary evidence for the two main findings.

Related work. Our study sits between model-based offline RL, latent world-model planning, and offline-to-online adaptation. Offline model-based RL learns dynamics from fixed datasets and uses those models for planning or policy improvement (Kidambi et al., 2020; Yu et al., 2020; Janner et al., 2019), while recent latent world models and joint-embedding predictors make visual control feasible by planning in compact representation spaces (Byravan et al., 2020; Hafner et al., 2023; Hansen et al., 2022; Maes et al., 2026). Offline-to-online methods emphasize how policies trained from logged data behave when deployed with limited interaction (Nair et al., 2020; Kumar et al., 2020), and trajectory/waypoint-style planners use dataset structure as an action prior (Janner et al., 2022). We focus on an evaluation-level failure mode: under closed-loop MPC, the same offline-trained model can appear weak or strong depending on the replanning interval, objective time index, and local controllability interface.

2. Setup: Planning from Offline Trajectories

We evaluate LeWM-style planners trained from offline demonstrations. At test time, the planner receives a current observation and goal observation, optimizes a length-H action sequence in latent imagination, executes the first K action chunks, observes the new state, and replans. The key regime is closed-loop receding-horizon execution, with $K < H$. This is offline-to-online adaptation in the evaluation sense: the model is fixed after offline training, but action selection adapts online through repeated replanning from states induced by prior executed prefixes. Unless otherwise stated, the main deceptive-navigation experiments use $H = 15$ and $K = 3$.

Let $\hat{z}_t$ denote the predicted latent state at imagined step t and z_g the goal embedding. The terminal objective scores candidates as

$$c_{\text{term}} = \|\hat{z}_H - z_g\|_2^2. \tag{1}$$

We compare two time-aligned alternatives:

$$c_{\text{prefix}} = \|\hat{z}_K - z_g\|_2^2, \tag{2}$$

and

$$c_{\text{run}} = \sum_{t=1}^{H} w_t \|\hat{z}_t - z_g\|_2^2, \qquad w_t \propto \gamma^{t-1}. \tag{3}$$

Unless otherwise stated, running-cost results use $\gamma = 0.95$. These are standard MPC-style distinctions between terminal and stage costs (Mayne et al., 2000); here they are used as diagnostics for latent world-model planning under offline-to-online evaluation rather than proposed as new control machinery.

Table 1. Protocol summary. Success is the primary metric throughout; uncertainty is reported as fold standard deviation for held-out waypoint folds and as Wilson intervals in the appendix.

Experiment	Control protocol	Evaluation set
Standard objectives	$H = 15$, $K = 3$	$N = 200$ row-sampled starts; Cube $N = 100$
Interval sweep	$H = 15$; $K \in \{1, 3, 5, H\}$	$N = 200$ row-sampled starts
TwoRoom Far Door	$H = 15$, $K = 3$	$N = 200$ scalar; 5 held-out folds

Table 2. Representative standard-task success rates at $H = 15$, $K = 3$. Prefix-terminal evaluates the terminal latent distance at the executed step K rather than the imagined horizon H. Dashes denote prefix-terminal settings not evaluated for the released Reacher/Cube checkpoints; running-cost controls were run for all listed tasks. PushT standard is distinct from PushT Gate in Appendix B.3.

Task	terminal@H	prefix@K	running
PushT (std.)	17.0	94.5	86.5
TwoRoom	52.0	83.5	88.0
Reacher	20.5	–	100.0
Cube	54.0	–	67.0

Table 3. TwoRoom Far Door results at $H = 15$, $K = 3$. The scalar row is the best scalar/value-bootstrapped latent objective from Table 6; its dash indicates that no fold standard deviation is available for this single $N = 200$ scalar evaluation. Waypoint rows use the local actuator and report five-fold held-out results with same-episode leakage removed; nearest retrieval selects from the train-fold demonstration library, not from the evaluation episode.

Interface / selector	Mean success	Std.
Best scalar/value latent	26.5	–
Waypoint + direct final goal	78.8	2.5
Waypoint + nearest retrieval	92.7	0.9
Waypoint + raw LeWM latent	90.3	4.4
Waypoint + learned control metric	88.5	3.0

Deceptive navigation. We use TwoRoom Far Door to test whether aligned costs imply genuine lookahead. TwoRoom Far Door requires moving away from the Euclidean goal direction to pass through a far doorway. The dataset contains 400 successful expert episodes after filtering.

3. Time-Index Alignment Repairs Standard MPC Failures

Figure 2 and Table 2 summarize representative standard-task results. Under closed-loop MPC, terminal-at-H is consistently weak. On the tasks where prefix-terminal was evaluated, prefix-terminal recovers much of the loss; running costs recover much of the loss across the listed standard tasks. This shows that a large part of the apparent planning failure is a scoring-time mismatch rather than a learned-dynamics failure. A low terminal@H score under closed-loop MPC should therefore not be interpreted as model failure until prefix@K or running-cost controls are checked.

The result changes the interpretation of LeWM-style evaluation. When $K = H$, terminal costs are aligned with the action sequence actually executed. When $K < H$, terminal@H can prefer plans whose early actions only make sense if the unexecuted suffix is also carried out. The prefix@K comparator is therefore an important diagnostic: it tests whether the failure is simply that the terminal objective is evaluated at the wrong time. The interval sweep in Figure 2 makes the same point without changing the model or optimizer: terminal scoring improves as K approaches H, while running cost is strongest in the frequent-feedback regimes.

This repair is substantial but not a complete solution. The next section shows that deceptive navigation can remain hard even after replacing terminal@H with prefix or running scores.

4. Deceptive Navigation Requires a Controllability Interface

On TwoRoom Far Door, aligned scalar objectives do not solve the task. With $H = 15$, $K = 3$, terminal@H reaches 7.0%, prefix-terminal reaches 20.5%, and running cost reaches 14.5%. Value bootstrapping improves the best baseline to 26.5% at one weight, but remains far below the waypoint/local-actuator results below. The failure is not only the time index. The planner needs to prefer prefixes that move toward a doorway bottleneck even when those prefixes are not immediately close to the final goal in latent Euclidean distance. This points to a proposal and local-control-support problem, not merely to a different scalar shaping term.

We therefore tested a hierarchical waypoint/local-actuator interface. The planner chooses an intermediate waypoint from a train-fold demonstration library, excluding the evaluation fold, then a learned local controller executes a short K-step action prefix toward that waypoint. This changes the interface from "sample raw action sequences and score their terminal latent states" to "choose a plausible local edge and execute it." Table 3 reports the best scalar baseline alongside five-fold held-out waypoint results with same-episode leakage removed.

This is the clearest repair, but its interpretation is deliberately

Figure 2. Replanning-interval sweep at $H = 15$. Terminal-at-H degrades when only a prefix of the plan is executed before replanning, while running latent cost remains strong under partial replanning. Raw values appear in Table 4.

narrow. The direct-final-goal row is not the raw scalar CEM planner; it uses the same learned local actuator with the final goal as the waypoint, which is why it is far stronger than scalar latent objectives. Raw LeWM latents are competitive subgoal selectors inside the hierarchical interface, but they do not dominate nearest retrieval. We also tested a learned graph-value scorer that predicts graph distance over the train-fold waypoint library. These graph-value experiments use a separate matched ablation protocol, so comparisons should be made within that block rather than against the primary raw-latent row in Table 3. In that matched five-fold ablation, the best graph-value variant reached $91.5\% \pm 2.3$ success versus a rerun raw-latent baseline of $87.3\% \pm 3.2$, while nearest retrieval remained slightly stronger at $92.7\% \pm 0.9$. Under a longer-goal stress test, graph-value improved from $67.2\% \pm 3.8$ to $70.7\% \pm 5.3$. These results strengthen the diagnosis without changing the headline: learned topological scoring helps, but the main repair is still the hierarchical controllability interface rather than another unconstrained scalar latent cost. We therefore treat this as evidence about the planning interface, not as a LeWM scoring improvement. The stronger contribution is the diagnosis: deceptive navigation exposes a missing controllability/proposal interface in latent world-model planning. The successful repair is not another scalar cost; it is an interface that restricts planning to locally executable edges.

5. Reporting Recommendations for Offline-to-Online MPC

The main results suggest three reporting recommendations for latent world-model planning from offline datasets. First, papers should report both H and K; evaluating only $K = H$ does not test closed-loop replanning. Second, scoring objectives should be indexed to the execution protocol, and prefix-terminal@K should be a baseline whenever $K < H$. Third, deceptive or detour-style tasks should include controls that distinguish latent goal similarity from local controllability, such as nearest-retrieval controls drawn from train-fold libraries or action-replay edges. For contact-rich tasks evaluated from saved dataset rows, Appendix B.3 adds one further sanity check: if expert actions cannot be replayed from reset states, the benchmark is not measuring the intended decision problem.

These recommendations are especially relevant for offline-to-online adaptation. The world model is trained offline, but online planning repeatedly queries states induced by its own previous choices. If the objective or proposal interface is mismatched, performance can collapse even when the learned latent model appears plausible.

Scope. We do not claim that running costs are novel MPC machinery or that waypoint retrieval is a final algorithmic solution. The PushT Gate analysis in the appendix is a reset-fidelity sanity check, not primary method evidence. The contribution is narrower: under fixed offline-trained world models, small choices in the online evaluation protocol can determine whether a benchmark measures model quality, objective alignment, or local controllability.

6. Conclusion

Closed-loop MPC exposes hidden failure modes in LeWM-style latent planning. Standard tasks reveal a time-index mismatch: terminal@H is misaligned when only $K < H$ actions are executed. Deceptive navigation reveals a controllability mismatch: latent proximity alone is not a reliable local control metric, and waypoint/local-actuator interfaces are a stronger planning interface. Together, these findings support a more careful benchmark protocol for offline-to-online world-model planning and motivate future methods that learn controllable latent edges rather than only scalar latent goal distances. For offline-to-online decision-making, the relevant unit of evaluation is not only the learned model, but the closed-loop interface through which that model is queried.

References

Byravan, A., Hasenclever, L., Trochim, P., Mirza, M., Ialongo, A., Tassa, Y., Springenberg, J. T., Abdolmaleki, A., Heess, N., Merel, J., and Riedmiller, M. Imagined value gradients: Model-based policy optimization with transferable latent dynamics models. *Conference on Robot Learning*, 2020.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Hansen, N., Su, H., and Wang, X. Temporal difference learning for model predictive control. In *Proceedings of the International Conference on Machine Learning*, 2022.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the International Conference on Machine Learning*, 2022.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. MOReL: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Maes, L., Le Lidec, Q., Scieur, D., LeCun, Y., and Balestriero, R. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. M. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

Nair, A., Gupta, A., Dalal, M., and Levine, S. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

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

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., and Ma, T. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, 2020.

A. Appendix: Evaluation Protocol

All main deceptive-navigation experiments use closed-loop receding-horizon MPC with planning horizon $H = 15$ and execution interval $K = 3$ unless otherwise stated. The planner optimizes a length-H imagined action sequence, executes only the first K actions, observes the new state, and replans. This separation is essential: terminal-at-H evaluates the end of the imagined rollout, while the system only commits to the prefix.

The primary scalar objectives are:

$$c_{\text{term}} = \|\hat{z}_H - z_g\|_2^2, \tag{4}$$

$$c_{\text{prefix}} = \|\hat{z}_K - z_g\|_2^2, \tag{5}$$

$$c_{\text{run}} = \sum_{t=1}^{H} w_t \|\hat{z}_t - z_g\|_2^2, \quad w_t \propto \gamma^{t-1}. \tag{6}$$

For value bootstrapping, the prefix-value objective combines prefix-terminal-at-K with a learned value estimate on the state at step K. For waypoint experiments, a local actuator is trained from offline demonstrations to execute a short prefix toward a selected intermediate waypoint. The held-out TwoRoom experiments train local actuators and build waypoint libraries from episodes outside the evaluation fold.

Implementation details. The planning horizon H and replan interval K count action chunks; each chunk contains five primitive environment steps. Unless otherwise stated, CEM uses 300 action-sequence candidates, 30 optimization iterations, and the best 30 candidates per iteration. The $H = 15$, $K = 3$ evaluations use an environment budget of 75 primitive steps and a maximum episode length of 150 primitive steps. Success is the environment termination signal for each task, evaluated from row-sampled dataset states and future goal states. PushT, TwoRoom, and TwoRoom Far Door use checkpoints trained in this codebase; Reacher and Cube use the official released LeWM checkpoints. Waypoint/local-actuator experiments choose from 128 nearest current-state neighbors, use waypoints 15 primitive environment steps, or 3 action chunks, in the future, and train the local actuator on episodes outside the held-out fold using five episode-id modulo folds. The split actuator is a three-layer MLP with hidden width 512 trained for 200 epochs.

B. Appendix: Complete Deceptive-Task Results

B.1. Deceptive Dataset Generation

Both deceptive datasets are stored as HDF5 rollout datasets generated with the same `World.record_dataset` interface used by the rest of the codebase, including rendered 224×224 observations, actions, and task state fields used for reset and evaluation.

Table 4. $H = 15$ replanning-interval sweep. Each cell is terminal@H / running-cost success. The pattern supports the time-index diagnosis: terminal scoring generally improves as the executed prefix approaches the imagined horizon, while running cost remains strong under frequent replanning.

Task	$K = 1$	$K = 3$	$K = 5$	$K = H$
PushT (std.)	16.0 / 92.0	17.0 / 86.5	13.0 / 88.0	36.0 / 79.0
TwoRoom	45.0 / 90.0	52.0 / 88.0	58.5 / 88.5	77.5 / 87.0
Reacher	19.0 / 100.0	20.5 / 100.0	21.5 / 99.5	41.0 / 73.0

Table 5. Wilson 95% intervals for representative standard-task success rates in Table 2. Values are percentages over $N = 200$ row-sampled evaluation starts, except Cube, which uses $N = 100$. Dashes denote prefix-terminal settings not evaluated for the released Reacher/Cube checkpoints.

Task	terminal@H	prefix@K	running
PushT (std.)	17.0 [12.4, 22.8]	94.5 [90.4, 96.9]	86.5 [81.1, 90.6]
TwoRoom	52.0 [45.1, 58.8]	83.5 [77.7, 88.0]	88.0 [82.8, 91.8]
Reacher	20.5 [15.5, 26.6]	–	100.0 [98.1, 100.0]
Cube	54.0 [44.3, 63.4]	–	67.0 [57.3, 75.4]

Deceptive dataset generation. The deceptive datasets are generated from scripted expert policies. For TwoRoom Far Door, the policy first aligns the agent with the low far-door waypoint, crosses the wall through that doorway, and then moves toward the sampled final goal. We record 400 successful episodes with eight parallel environments, seed 42, and an 80-step cap. For PushT Gate, the policy stages the point agent behind the object, pushes the object to a pre-gate waypoint, through the gate opening, and then toward the goal. We generate 900 raw episodes with eight parallel environments, seed 42, and a 150-step cap, then keep the first 400 episodes whose final row is terminated and not truncated. These are expert-demonstration diagnostics rather than standard benchmark releases; the PushT Gate results should be read with the reset-state caveat in Appendix B.3.

TwoRoom Far Door. TwoRoom Far Door uses a custom TwoRoom environment with one vertical wall and a single doorway whose center is sampled uniformly from $y \in [42, 74]$. Initial agent positions are sampled on the left side with $x \in [40, 78]$, while target positions are sampled on the right side with $x \in [146, 184]$; both are placed in the lower band $y \in [150, 188]$. This makes the task deceptive because the straight-line direction to the target points into the wall, while the valid route first moves upward to the far doorway. The expert is a waypoint controller: it aligns the agent vertically with the doorway, crosses through the doorway, then aligns vertically with the final target and moves to it. We generated `tworoom_far_door_v2_expert` with seed 42, 8 parallel environments, 400 episodes, and an 80-step episode limit. The stored dataset includes the

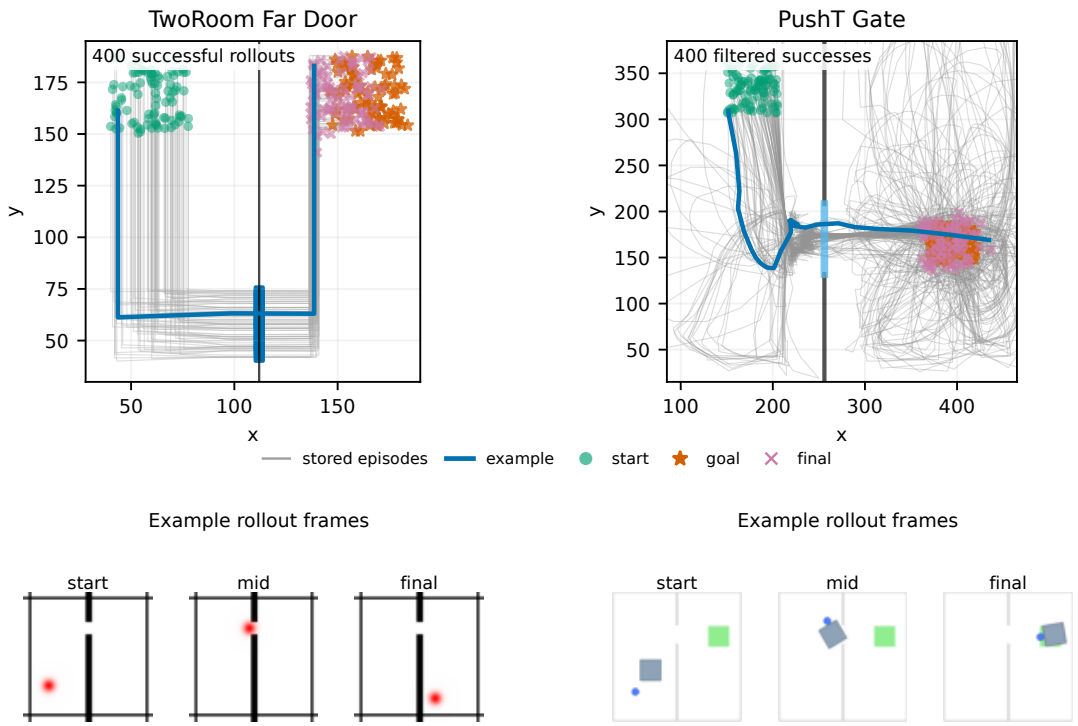

Figure 3. Deceptive dataset geometry and representative rollouts. Top: stored expert trajectories with one example rollout highlighted in blue. TwoRoom Far Door demonstrations route through a low far doorway before moving to high-y goals; PushT Gate demonstrations push the object from the left side of the workspace through the gate opening toward right-side goals. Bottom: rendered start, midpoint, and final frames for the highlighted rollouts.

doorway coordinate `door_center_y`, which is restored during evaluation before resetting the saved agent and goal states.

PushT Gate. PushT Gate uses a PushT variant with a vertical barrier at $x = 256$ and a narrow gate centered at $y = 170$ with half-height 38. At reset, the agent starts to the left/lower side of the barrier, the block starts left of the barrier, and the block goal is sampled on the right side near the gate. The expert first drives the block to a staging point just before the gate, then through the gate, then to the final goal; the agent tracks a pushing point behind the desired block motion. For `pusht_gate_v2_expert`, we generated 900 raw expert rollouts with seed 42, 8 parallel environments, and a 150-step episode limit, then filtered to the first 400 successful episodes. As discussed below, this dataset is useful as a reset-fidelity diagnostic because the saved state contains agent position, block pose, and agent velocity, but not all simulator state needed to replay arbitrary contact-rich nonzero reset rows.

B.2. TwoRoom Far Door

TwoRoom Far Door contains 400 successful expert demonstrations after cleaning. The task is deceptive because the agent must initially move toward a far doorway rather than directly toward the final goal.

Table 6. TwoRoom Far Door scalar and value-bootstrapped results, $H = 15$, $K = 3$, $N = 200$.

Objective	Success (%)
terminal@H	7.0
prefix-terminal@K	20.5
running, $\gamma = 0.95$	14.5
prefix-value, $\alpha = 0.25$	22.0
prefix-value, $\alpha = 0.50$	26.5
prefix-value, $\alpha = 1.00$	20.5

The waypoint interface changes the bottleneck from raw action-sequence proposal to locally executable edge selection. The result is a real improvement over scalar latent costs, but it is not a clean win for learned latent scoring alone: nearest retrieval waypoint selection remains the strongest held-out control. This is why the main-paper claim is framed as a controllability-interface diagnosis rather than as a fully solved LeWM scoring method.

The matched ablation supports the same conclusion as the primary waypoint table. Learning graph distance gives a useful topological signal, especially relative to raw latent waypoint scoring in the same protocol, but it does not remove

Table 7. TwoRoom Far Door five-fold held-out waypoint results. Values are success rates over five held-out episode folds with same-episode leakage removed; nearest retrieval selects from the train-fold demonstration library, not from the evaluation episode.

Waypoint selector	Fold successes	Mean	Std.
Final-goal waypoint	83.0, 76.0, 76.5, 79.5, 79.0	78.8	2.5
Nearest retrieval waypoint	92.0, 92.5, 94.0, 93.5, 91.5	92.7	0.9
Raw LeWM latent waypoint	95.0, 87.0, 83.5, 92.0, 94.0	90.3	4.4
Learned control metric	92.0, 85.5, 84.5, 91.5, 89.0	88.5	3.0

Table 8. TwoRoom Far Door graph-value ablations. The graph-value scorer is trained on the train-fold waypoint graph; c_w is the current-state weighting used when scoring candidate waypoints. The go25 and go40 labels denote the short- and long-goal-offset settings used for this ablation, with go40 serving as the longer-goal stress setting. Raw latent values in this table are rerun matched baselines for this ablation protocol, so comparisons should be made within each setting.

Setting	Selector	Fold successes	Mean	Std.
go25	Nearest retrieval waypoint	92.0, 92.5, 94.0, 93.5, 91.5	92.7	0.9
go25	Raw LeWM latent waypoint	89.5, 84.0, 83.0, 89.0, 91.0	87.3	3.2
go25	Graph value, $c_w = 0.0$	95.0, 91.5, 89.0, 93.0, 89.0	91.5	2.3
go25	Graph value, $c_w = 0.25$	89.5, 92.0, 90.0, 91.0, 90.5	90.6	0.9
go25	Graph value, $c_w = 0.5$	86.0, 91.5, 89.5, 89.5, 90.0	89.3	1.8
go25	Graph value, $c_w = 1.0$	88.5, 90.5, 93.5, 90.0, 87.0	89.9	2.2
go25	Graph value, $c_w = 2.0$	91.0, 89.0, 94.0, 89.5, 88.5	90.4	2.0
go40	Raw LeWM latent waypoint	70.5, 65.5, 61.0, 71.5, 67.5	67.2	3.8
go40	Graph-state distance	64.5, 67.0, 69.5, 73.0, 67.0	68.2	2.9
go40	Graph value, $c_w = 1.0$	64.5, 76.0, 70.0, 77.5, 65.5	70.7	5.3

the need for nearest retrieval controls or local executability checks.

B.3. PushT Gate

PushT Gate also contains 400 successful filtered demonstrations. The task is contact-rich and requires pushing the block through a narrow gate. We report this as an appendix-only protocol caveat rather than as one of the two main method findings. Initial arbitrary-row evaluation appeared uniformly poor, but the replay diagnostic showed that arbitrary nonzero reset states are not physically reproducible. The dataset stores a seven-dimensional state containing agent position, block pose, and agent velocity, but not full contact-relevant simulator state such as block velocity.

Table 9. PushT Gate scalar and value-bootstrapped results under the original arbitrary-row evaluation, $H = 15$, $K = 3$, $N = 200$. These success rates are reported for completeness, but the protocol is confounded by reset-state fidelity.

Objective	Success (%)
terminal@H	3.0
prefix-terminal@K	3.5
running, $\gamma = 0.95$	3.0
prefix-value, $\alpha = 0.25$	4.0
prefix-value, $\alpha = 0.50$	3.5
prefix-value, $\alpha = 1.00$	3.5

The corrected PushT result has two interpretations. First, arbitrary-row evaluation should not be used unless the saved state is sufficient to reconstruct the simulator, including contact-relevant hidden dynamics. Second, even after correcting this artifact, the task remains a method failure: expert replay from the same starts is high, while LeWM-style objectives and waypoint variants remain near floor. A fold-specific graph-value diagnostic did not change this conclusion: nearest waypoint selection reached $8.7\% \pm 1.2$, graph-value selection reached $8.8\% \pm 1.4$, and raw latent selection reached $6.5\% \pm 2.5$ under the same held-out arbitrary-row protocol.

Table 10. PushT Gate expert action replay diagnostic on the held-out split. Replay succeeds from episode starts but collapses from nonzero reset rows.

Start rows	N	Expert replay success (%)
All valid starts	200	21.5
Episode starts only	74	91.9
Nonzero starts	200	16.5

Table 11. PushT Gate step-zero-only evaluation after correcting to held-out episode starts, $N = 74$.

Method	Success (%)
terminal@H	0.0
prefix-terminal@K	0.0
running, $\gamma = 0.95$	0.0
Waypoint direct goal, learned local actuator	13.5
Waypoint nearest, learned local actuator	8.1
Waypoint raw latent, learned local actuator	10.8
Waypoint nearest, selected demo action chunk	10.8
Waypoint raw latent, selected demo action chunk	1.4

C. Appendix: Reporting Checklist

For offline-to-online latent planning experiments, we recommend reporting:

1. The planning horizon H and execution/replan interval K.
2. Whether the objective is terminal-at-H, prefix-terminal-at-K, running cost, value-bootstrapped, or another scoring rule.
3. Whether the action proposal is raw action-sequence sampling, retrieval, learned local-prefix proposal, or waypoint/local-actuator planning.
4. Whether held-out episode splits prevent same-episode waypoint or action-chunk leakage.
5. Whether expert actions can be replayed from reset states, especially in contact-rich environments.