World Model Control by Trajectory Reachability Models

Anonymous authors
Paper under double-blind review

Abstract

Latent world models can learn representations that contain information needed for control, while the downstream controller may still rank candidate actions poorly when it relies on terminal latent distance alone. We study this failure in a fixed encoder and introduce a trajectory reachability model (TRM), a small temporal pairwise cost trained from logged trajectories and used to rank predicted endpoints of a candidate action sequence against a goal. In the TwoRoom evaluation on 100 episodes from a high distance range, the original controller with LeWorldModel reaches 7.0% mean success. Temporal TRM trained after excluding all evaluation episodes reaches 96.7%, shuffled label controls stay at 0.0%, and TRM also improves PLDM from 32.7% to 84.0%. TopoNav, a separate pixel navigation task, shows the same link between repaired ranking and closed loop control. The selection audit with shared candidates (SASC) and rowspace interventions localize the ranking failure to underweighted latent directions carrying the action decision. The XY rowspace contributes less than 1% of terminal latent MSE but carries most of the information needed for control. Coverage and boundary tests define the scope. Balanced doorway coverage restores a 0.0% coverage failure to 100.0% success, while an unseen wall orientation and contact rich PushT expose layout, dynamics, and recovery limits.

1 Introduction

Latent world models promise to learn compact representations from pixels, roll them forward under candidate action sequences, and choose actions by optimizing in latent space. This promise underlies model based reinforcement learning and visual planning systems, ranging from probabilistic dynamics models with trajectory sampling to latent imagination methods and JEPA based world models (Chua et al., 2018; Hafner et al., 2020; Zhou et al., 2025; Maes et al., 2026b).

In latent MPC, a CEM controller rolls out candidate action sequences in the learned predictor and ranks their endpoints by a terminal cost, often Euclidean distance to the goal latent.

The common terminal cost formulation assumes that Euclidean proximity in the learned latent space is a good proxy for task progress. This requires the controller's cost to weight the control variables so that terminal cost tracks the gap between the current and target states in the real world. If these variables occupy a small subspace, Euclidean latent distance can be dominated by residual directions encoding nuisance variation or local discrimination (Gelada et al., 2019; Zhang et al., 2021a). In that case, the controller's terminal cost underweights the information needed to choose the correct action sequence.

We investigate this problem in LeWorldModel (LeWM), an end to end joint embedding predictive architecture trained from pixels (Maes et al., 2026b). The high distance TwoRoom evaluation exposes a sharp failure under the raw latent cost. The same evaluation episodes are solvable by the same CEM controller with a cost aligned with reachability, and LeWM latent states encode XY position almost perfectly under a linear readout. Together, these observations identify the terminal cost as the failure point.

We call our intervention a *trajectory reachability model* (TRM). It learns a scalar pairwise distance surrogate from logged trajectory structure and uses it as a terminal cost on predicted endpoints against goal latent states. The world model remains fixed.

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

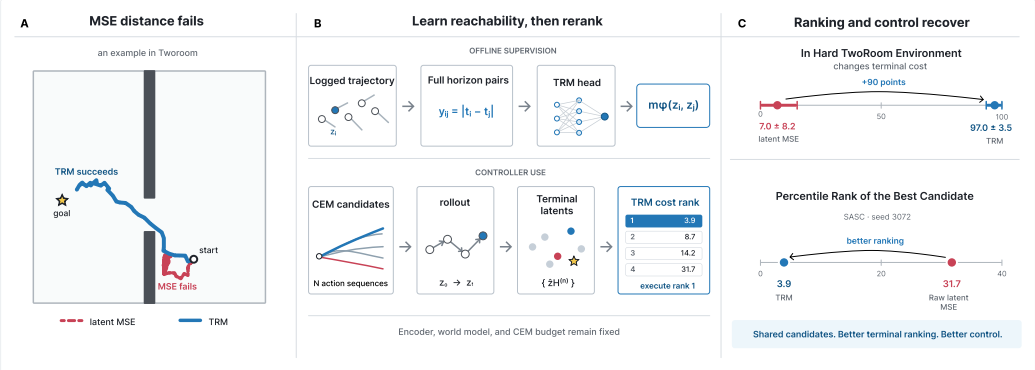

Figure 1: Overview of the terminal ranking failure and the TRM repair. (A) Raw latent MSE drives CEM toward an unsuccessful plan, whereas TRM follows the doorway route. (B) Temporal pairs train the TRM head, which changes only terminal ranking under a fixed world model and controller. (C) TRM trained on the complete cache raises success on the high distance evaluation from $7.0 \pm 8.2\%$ to $97.0 \pm 3.5\%$ and improves the seed 3072 SASC best rank percentile from 31.7 to 3.9.

The central design principle is that the terminal cost must be trained at the temporal scale where it will be used. A local temporal head can be accurate on short pairs yet fail to rank endpoints from long horizon candidates. We use balanced trajectory pair sampling across the full horizon in TwoRoom and evaluate both replacement and hybrid costs. The experiments first expose a controller failure, then change only the terminal cost, localize why the original terminal MSE cost underweights useful state, and test where the repair stops helping.

This paper makes four contributions.

1. **Terminal ranking failure.** We identify a mismatch between accurately encoded task state and the terminal cost used for control, and study it through the minimal TRM intervention.
2. **Trajectory supervision across the full horizon.** We show that sampling is part of the method, not an implementation detail. Broad trajectory coverage and temporal deltas across the full horizon are critical in ablation.
3. **Evidence from candidate ranks.** We use SASC and rowspace interventions to test whether temporal TRM works by correcting candidate rank.
4. **Navigation benchmarks and boundary conditions.** The repair recurs in LeWM, PLDM, and TopoNav. Doorway coverage, unseen layouts, and PushT establish its empirical scope.

2 Related Work

World models and latent space control. Model based control commonly trains a dynamics model and optimizes candidate action sequences through it, from data efficient probabilistic models (Deisenroth & Rasmussen, 2011; Chua et al., 2018; Janner et al., 2019) to visual foresight systems (Ebert et al., 2018) and learned latent dynamics (Watter et al., 2015; Hafner et al., 2019). Latent world models move this optimization into a compact representation, either to learn policies through imagined rollouts (Ha & Schmidhuber, 2018; Hafner et al., 2020; Hansen et al., 2022; 2024) or to rank goal conditioned action sequences in learned visual features (Zhou et al., 2025; Maes et al., 2026b). We examine the assumption that terminal Euclidean distance in the learned latent space is a reliable cost (Lambert et al., 2020).

Goal reaching and reachability. Goal conditioned RL has long recognized that representation distance can be a poor proxy for reachability. Work on universal value functions (Schaul et al., 2015), hindsight replay (Andrychowicz et al., 2017), hierarchical goals (Nachum et al., 2018), imagined visual goals (Nair et al., 2018), and coverage driven goal sampling (Pong et al., 2020; Eysenbach et al., 2021; 2022; Ma et al., 2023) shows that a goal representation alone does not tell the controller

which actions can reach the goal. Replay buffer graph search, topological memory, and latent landmark world models use graph or value structure for long horizon goal reaching (Savinov et al., 2018; Eysenbach et al., 2019; Zhang et al., 2021b). Tian et al. (2021) learn a functional distance through offline Q-learning and use it to score model-predicted outcomes in visual MPC. Our study examines terminal ranking in frozen latent models through shared-candidate and rowspace interventions. REPlan separates compact representation learning from a reachability module estimating temporal distance (Qian et al., 2023). Recent work learns temporal distance aware representations, temporal metrics related to successor structure, or quasimetric goal reaching distances (Dayan, 1993; Bae et al., 2025; Park et al., 2024b;a; Myers et al., 2024; 2025). We use the same reachability idea to define the controller terminal cost and then audit whether it changes candidate ranks.

Controller interfaces. Sampling based controllers such as CEM (Rubinstein, 1999; Pinneri et al., 2021) and path integral control (Williams et al., 2017) repeatedly score many candidate action sequences and then act on the best ranked candidates. This makes the interface between the learned representation and the scalar terminal cost part of the control problem. Recent work improves adjacent interfaces by closing the train test gap for gradient based world model control (Parthasarathy et al., 2025) or by learning priors that change imagination sampling (Wang et al., 2026). TRM keeps the sampler and the fixed world model unchanged and changes only the scalar endpoint cost used by the controller.

Concurrent views on plannability in joint embedding world models. The RC-AUX concurrent preprint and our work both point to a similar problem in joint embedding world models. It also examines downstream planning with predictive representations (Li et al., 2026; Maes et al., 2026b). RC-AUX keeps the LEWM backbone architecture unchanged but changes training or continuation training by adding multi horizon open loop prediction, budget conditioned reachability supervision with temporal hard negatives, and optionally a reachability aware controller. In contrast, TRM freezes the trained encoder and predictor, then learns a continuous pairwise ranking signal for candidate action sequences and endpoints.

3 Latent Terminal Cost Planning with Fixed World Models

Latent MPC **with terminal costs.** We denote the observation at time t by o_t. Let an encoder map observations to a latent state, $\mathbf{z}_t = f_\theta(o_t)$, and let $F_\theta^{(H)}$ denote the deterministic H-step latent rollout operator

$$\hat{\mathbf{z}}_{t+H} = F_\theta^{(H)}(\mathbf{z}_t, \mathbf{a}_{0:H-1}). \tag{1}$$

Given a goal observation o_g with latent $\mathbf{z}_g = f_\theta(o_g)$, the usual terminal objective is to minimize

$$c_{\mathrm{lat}}(\mathbf{a}_{0:H-1}) = \|\hat{\mathbf{z}}_{t+H} - \mathbf{z}_g\|_2^2 . \tag{2}$$

A CEM controller (Rubinstein, 1999) samples candidate action sequences, uses Eq. 2 to update its search distribution, executes the configured receding-horizon prefix of the final plan, and replans. This objective is justified only when lower terminal latent distance implies higher probability of reaching the goal.

Terminal cost failure. We test for a terminal cost failure when the latent state contains information needed for the task, the controller's terminal cost ranks candidate action sequences poorly with respect to task progress, and replacing or restricting that cost improves control while the checkpoint, evaluation set, and controller remain fixed.

4 Method: Trajectory Reachability Model (TRM)

Design goal. TRM supplies a nonnegative pairwise terminal cost for latent controllers with a fixed world model and CEM. The encoder, dynamics, candidate sampler, and CEM update rule remain unchanged. The inductive assumption is that trajectory connectivity provides a useful ordering of candidate endpoints at the controller's horizon.

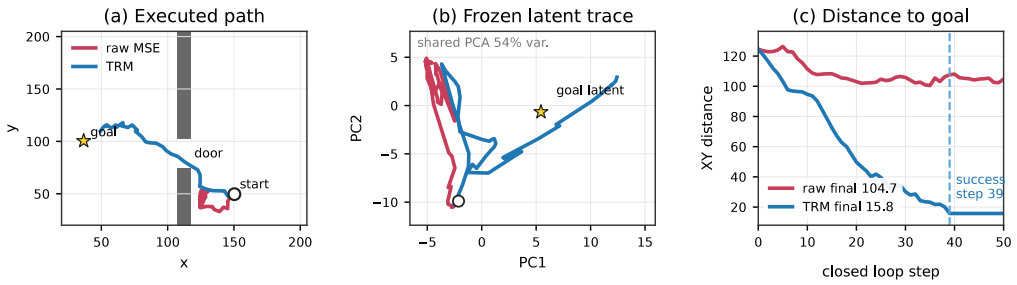

Figure 2: Representative trace from the high distance TwoRoom evaluation under a fixed start, goal, checkpoint, and controller budget. Raw latent MSE stops near the wall, whereas temporal TRM follows the doorway route and succeeds. The latent panel is a shared PCA visualization.

Pairwise reachability head. To avoid relying on specific oracle information, TRM trains a small pairwise head $m_\phi(\mathbf{z}_i, \mathbf{z}_j)$ on encoded state pairs. The default feature map is

$$[\mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_i - \mathbf{z}_j, |\mathbf{z}_i - \mathbf{z}_j|]. \tag{3}$$

In the main TwoRoom method experiments, we supervise the distance head using temporal labels from pairs within the same episode,

$$y_{ij} = |t_i - t_j|, \tag{4}$$

which serve as a symmetric proxy for reachability along logged trajectories. Temporal separation captures the accumulated transition gap between two states under the logged dynamics. Smaller gaps indicate that the states can be connected by fewer feasible actions, while larger gaps usually imply greater changes in configuration needed for the task. This proxy can fail when the logging distribution misses regions that are critical for connectivity. In the coverage stress test, we measure that failure directly. For PushT and for some stress test baselines, we also train the same pairwise architecture on task state or oracle geodesic labels to test whether stronger supervision can express missing ranks.

Pair sampling at the whole planning horizon. The sampling rule is part of the method. We sample an episode, choose a temporal separation over the full episode horizon, choose a valid start time, and train on pairs in random order. Balanced sampling over the full horizon prevents the head from being dominated by local neighbors. It also exposes the head to the temporal scale at which the terminal cost will rank CEM candidates. Table 4 gives the key comparison. Broad random coverage over full episodes reaches 90.0%, balanced pairs over full episodes reach 97.5%, and a short horizon head with maximum $\Delta = 50$ reaches only 35.0% even with 100,000 training pairs.

Replacement and hybrid terminal costs. At control time, CEM proposes candidates, the fixed world model rolls each candidate to a predicted terminal latent state $\hat{\mathbf{z}}^{(i)}_{t+H}$, and TRM computes the temporal terminal cost against the goal latent state.

$$c_{\text{TRM}}(\mathbf{a}^{(i)}_{0:H-1}) = m_\phi\left(\hat{\mathbf{z}}^{(i)}_{t+H}, \mathbf{z}_g\right). \tag{5}$$

In settings dominated by reachability, we use c_{TRM} as a replacement terminal cost. In settings where raw latent distance may preserve useful local physical information, we use the standardized hybrid cost below.

$$c_{\text{hyb}} = \text{std}(c_{\text{lat}}) + \lambda\, \text{std}(m_\phi). \tag{6}$$

Here std standardizes costs within the current candidate set, $\text{std}(c_i) = (c_i - \mu_c)/(\sigma_c + \epsilon)$. The PushT sweeps λ in the supplement, whereas TwoRoom primarily uses replacement costs. TRM is a scalar endpoint terminal cost used by CEM for action selection. Table 1 summarizes the evaluation tasks and the evidence used for each.

5 Experimental Protocol

TwoRoom evaluation sets. TwoRoom contains two rooms separated by a wall and connected by a doorway. We use balanced evaluation sets and distance controlled evaluation sets for start and

Table 1: Evaluation suite used in the paper. Each row keeps the controller budget fixed within a comparison and changes only the terminal cost or the diagnostic setting described in the text.

Task	Role	Data source	Seeds	Primary evidence
TWOROOM	Main navigation task	Public LeWorldModel cache	3072/3073/ 3074	Success, SASC, rowspace audits
TopoNav	Non-TWOROOM navigation diagnostic	Generated grid trajectories	3072/3073/ 3074	Success and exact graph distance SASC
PUSHT	Contact rich boundary task	Public PushT expert data	3072/3073/ 3074	Success, selected distance, SASC

goal pairs. The distance controlled sets draw cases from the same Euclidean distance range in the same room and across the wall, preventing topology from being confounded with longer Euclidean distance. The main high distance evaluation contains 50 episodes in the same room and 50 episodes across the wall. Among the 50 goals across the wall, 47 require the doorway, and success is measured over 100 closed loop episodes per seed.

Models. The primary subject is LEWM with three seeds. We also evaluate a PLDM world model baseline from the stable world model ecosystem (Maes et al., 2026a) using evaluation sets drawn from the same complete cache to test whether the same terminal cost change (TRM) helps another fixed model.

Evaluation and reporting. Unless otherwise stated, comparisons use the same checkpoint family, HDF5 cache, evaluation set, random seed, evaluation budget, and CEM settings. For the high distance TWOROOM evaluation, this means 300 candidate samples, 10 CEM iterations, top-k 30, and evaluation budget 50. We report success percentages over fixed evaluation sets and means across seeds. The evidence chain combines closed loop success, audits on matched candidate sets, negative controls, and targeted stress tests.

Selection audit with shared candidates. For candidates $\{\mathbf{a}^{(i)}_{0:H-1}\}_{i=1}^{N}$, let $c(i)$ be the controller cost and $c^\star(i)$ a task-space diagnostic cost for the same candidate. The selection audit with shared candidates (SASC) holds the checkpoint, evaluation set, sampler, and CEM budget fixed and changes only the selector. In TWOROOM, $c^\star$ is computed by the known task-geometry rollout; TopoNav uses exact grid dynamics, and PUSHT uses simulator-executed outcomes. We calculate Spearman correlation within each candidate set and average over episodes or seeds as stated in the captions. Larger positive values indicate better agreement with the diagnostic cost. We also report the rank assigned to the oracle best candidate and, where applicable, selected task-space distance. Additional details are in Appendix B.1.

Implementation of the pairwise head. Pairwise heads use two hidden layers with 256 units each, SiLU nonlinearities, and a Softplus scalar output. They are trained with AdamW (learning rate 10^{-3}, weight decay 10^{-4}), batch size 1024, and Smooth-L1 loss on distances scaled by 224. This scale matches the TWOROOM image and coordinate scale and is used only to condition the regression loss. The primary temporal heads use 100,000 balanced training pairs and validation pairs disjoint from training, sampled from the same complete cache protocol. Shuffled controls permute only training labels while preserving architecture, data volume, and evaluation.

Data splits and leakage control. TRM uses logged trajectory pairs, not evaluation successes, closed loop rollouts, SASC candidate outcomes, or labels for test pairs. For the primary leakage audit on the high distance TWOROOM evaluation, all 200 evaluation start and goal rows and their 193 source episodes are excluded from the endpoints used to train and validate temporal pairs, giving row overlap 0 and episode overlap 0. This episode exclusion audit preserves the repair ($96.7\pm2.1\%$ success and $\rho_{\text{geo}} = 0.577$) while the corresponding shuffled head remains at 0.0%. The row trained

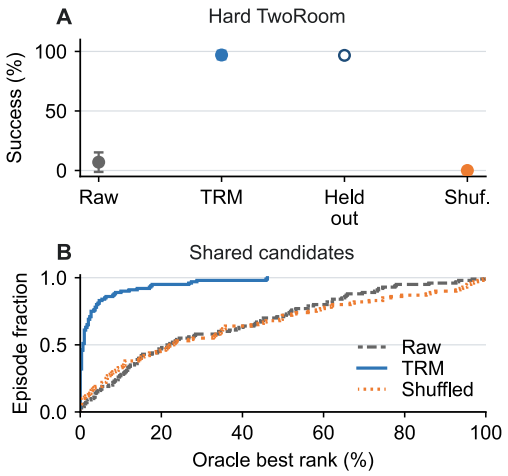

Figure 3: Control and shared candidate ranking. (A) Success on the high distance TwoRoom evaluation across three LeWM seeds. (B) Seed 3072 SASC rank distributions on fixed candidates from the same evaluation, where left is better.

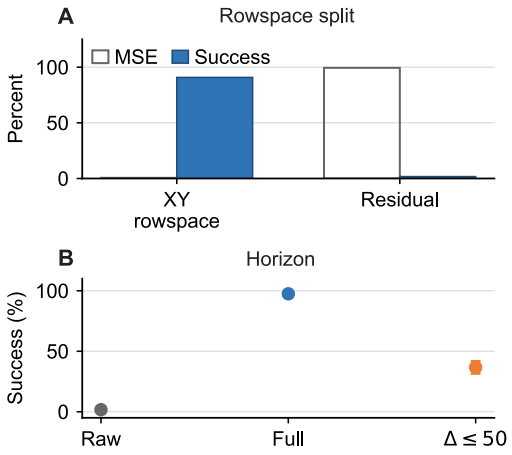

Figure 4: Mechanism and horizon evidence. (A) Latent MSE share and control success for XY rowspace and residual costs. (B) Success in the distance controlled evaluation with an action budget of 50 under raw, full episode, and short horizon costs.

on the complete cache in Table 2 is retained as a reference under the same cache setting. The result with episode exclusion is the primary reported TwoRoom result. For TopoNav, the exact start and goal pairs used for evaluation are also excluded from the training and validation pairs for both TRM and shuffled controls.

Coverage, baseline, and layout checks. We compare standardized latent, linear, diagonal Mahalanobis, pairwise MLP, and ranking-loss baselines under the same checkpoint, evaluation set, controller budget, and candidate pools. The pairwise MLP and ranking-loss stress tests use oracle geodesic supervision. Layout variants move the doorway along the original vertical wall or rotate the wall; observations and rollout environments reflect the changed geometry. The encoder, dynamics, policy checkpoint, controller, and budgets are fixed.

PushT. For PushT, we use the official LeWM checkpoint converted from the public artifact and the complete expert training dataset. We evaluate goal offsets 25 and 50 with an action budget of 50, and offset 75 with a budget of 75. Each protocol uses 50 evaluation episodes per reported seed. We use the same SASC metrics to relate candidate ranking to selected final distance and closed loop success.

6 Results

6.1 TRM Repairs the High Distance TwoRoom Evaluation

Table 2 tests whether changing only the terminal selector repairs the high distance TwoRoom evaluation. Raw LeWM latent MSE reaches 7.0% mean success. Temporal TRM reaches 96.7% with episode exclusion and 97.0% with the complete cache. Wrong-room and wall-stuck failures nearly disappear. Shuffled heads with matched architecture and data volume remain at 0.0%, linking the gain to temporal supervision.

Across three PLDM seeds on the same high distance evaluation set, the temporal recipe improves success from 32.7% to 84.0%. A separate seven-checkpoint LeWM screening comparison records four gains, two ties, and one decrease in success (Appendix F). Each checkpoint uses its own trained head and a common 50-episode screening set.

Table 2: Results on the high distance TWOROOM evaluation. Success is mean±sample std over three seeds, and the remaining columns are mean percentages. The episode exclusion row removes all evaluation start and goal episodes from pair training and validation.

Cost	Succ.	Same	Cross	Wrong	Stuck
Raw latent MSE	7.0±8.2	10.0	4.0	45.3	10.3
TRM, complete cache	97.0±3.5	95.3	98.7	0.0	0.3
TRM, episode exclusion	96.7±2.1	98.7	94.7	2.0	0.3
Shuffled temporal head	0.0±0.0	0.0	0.0	46.0	8.7
PLDM raw latent MSE	32.7±3.2	4.0	61.3	12.7	14.0
PLDM temporal TRM	84.0±9.5	79.3	88.7	3.0	2.0
PLDM shuffled head	0.0±0.0	0.0	0.0	50.0	2.7

Table 3: Seed 3072 SASC audit on the high distance evaluation with fixed candidates. The ρ columns are Spearman correlations with oracle terminal quality. Best rank pct. is the percentile assigned to the oracle best candidate, where lower is better.

Candidate cost	ρ Euclid.	ρ geodesic	Best rank pct.
Raw latent MSE	0.021	0.018	31.71
Decoded XY	0.876	0.860	0.59
TRM temporal labels	0.720	0.729	3.86
TRM shuffled labels	0.105	0.119	34.00

Figure 3 summarizes success and candidate ranks. On the seed 3072 high distance audit, TRM raises geodesic Spearman from 0.018 to 0.729 and moves the oracle best candidate from rank percentile 31.71 to 3.86. Shuffled-label ranks remain poor (Table 3).

Candidate availability and ranking. Simulator replay compares selectors on 300 shared candidates at each of 1,200 replanning states. Mean regret over finite outcomes is 22.67/13.02 pixels for raw/TRM scoring on raw-generated pools and 13.32/14.69 on TRM-generated pools. Appendix G reports oracle selection and the remaining-failure breakdown.

6.2 BROAD TEMPORAL COVERAGE AND TEMPORAL STRUCTURE DRIVE THE REPAIR

We next ask whether broad temporal supervision across the planning-relevant range matters beyond merely adding a learned scalar head. Table 4 and Figure 4B give the distance controlled comparison with an action budget of 50 across three seeds for the key horizon choices. Raw latent MSE reaches only 1.7% mean success, while balanced temporal supervision over full episodes with 100,000 pairs reaches 97.5%. The broader coverage and pair count sweeps are single seed diagnostics reported in the supplementary material.

Matched shuffled-label controls reach 0.0% success on the high distance evaluation across all three LEWM seeds and all three PLDM seeds. For LEWM seed 3072, shuffling lowers success from 99.0% to 0.0% and disrupts candidate ranks (Table 3), supporting the role of temporal structure.

6.3 COVERAGE, BASELINE, AND LAYOUT CHECKS

We test by varying the task, terminal cost class, logged pair coverage, and layout under fixed controller budgets. TopoNav uses a frozen encoder, exact frozen dynamics, and a fixed route biased candidate proposal shared by all selectors. Metric baselines use the same high distance evaluation set, controller budget, and matched candidate pools within each seed. The coverage check varies doorway examples in the logged pairs, while layout checks move the doorway or rotate the wall without retraining the fixed world model.

These results define the useful region. TopoNav shows that a distinct visual navigation topology reproduces the same pattern. In TopoNav, raw terminal latent cost succeeds on 15.0% of episodes,

Table 4: Horizon comparison on the distance controlled evaluation with an action budget of 50 across three LEWM seeds. Broader coverage and pair count sweeps are single seed diagnostics in the supplementary material.

Sampling	Temporal horizon	Seed 3072	Seed 3073	Seed 3074	Mean±std
Raw latent MSE	None	2.5	2.5	0.0	1.7±1.4
Balanced temporal	Full episode, 100k pairs	97.5	100.0	95.0	97.5±2.5
Balanced temporal	$\Delta \leq 50$, 100k pairs	35.0	42.5	32.5	36.7±5.2

Table 5: Metric baselines on matched high distance candidate pools, averaged over three seeds. Geodesic learned rows use oracle supervision; lower Rank and Dist. are better.

Selector	Succ.	ρ_{geo}	Rank	Dist.
Raw latent MSE	7.0	0.007	38.95	139.72
Standardized L2	3.3	0.007	40.34	147.59
Linear	15.3	0.354	17.81	88.75
Diag. Mahalanobis	21.3	0.279	18.93	99.37
Decoded XY geodesic	93.3	0.857	0.56	10.90
Temporal TRM	97.0	0.678	3.79	7.85
Geodesic pair MLP	99.3	0.908	0.45	7.61
Geodesic ranking	95.7	0.839	0.70	9.22
Shuffled temporal	0.0	0.053	44.10	138.86
Oracle geodesic	100.0	1.000	0.00	5.74

temporal TRM reaches 87.5% with $\rho_{\text{geo}} = 0.848$, and shuffled labels fall to 1.1%. The doorway coverage result is a recovery test. Adding doorway coverage moves the previous 0.0% coverage failure to 100.0% success, moves the rank assigned to the candidate with the best oracle cost from 34.62 to 0.49, and reduces final distance from 124.36 to 9.61. The layout result with unseen geometry is a boundary test. Moving the doorway along the same vertical wall improves over raw latent planning, but rotating the wall orientation collapses both ranking and success. Full stress test tables are in the supplementary material.

6.4 MECHANISM AND BOUNDARY EVIDENCE

The mechanism evidence indicates that raw latent MSE fails because it underweights state directions that matter for decisions. A linear XY probe trained on 20,000 states from the complete cache achieves test RMSE around 1.8 pixels and $R^2 = 0.998$ for all three LEWM seeds, so the position needed for the TWOROOM route decision is present. Yet raw latent MSE still collapses on the distance controlled evaluation with an action budget of 50, where LEWM reaches only 1.7% mean success. A decoded XY terminal cost raises success on this evaluation to 94.2%, using only predicted terminal latents and a learned readout rather than oracle state at evaluation time.

The stronger test is a rowspace intervention. Let the XY probe be $W\mathbf{z} + b$ and define $P_W = W^\top (WW^\top)^\dagger W$. On sampled candidate terminal cost, the XY rowspace accounts for only 0.5–0.7% of total latent MSE. Yet planning with rowspace only latent MSE reaches 90.8% mean success, while planning with the orthogonal residual remains at 1.7%. Figure 4A summarizes this asymmetry. The controller fails not because the relevant coordinate is outside the latent, but because raw Euclidean distance gives it negligible influence. Other controls give the same conclusion. An oracle task state cost solves the high distance evaluation, while increasing raw latent search to 1000 samples, 20 CEM iterations, and top-k 100 remains at 2.5%.

PUSHT tests TRM style task state heads as auxiliary ranking signals. At goal offset 25, raw latent planning is already near saturated at 88.0%, and learned replacement or hybrid costs do not signif-

Table 6: Supporting evaluations under fixed controller budgets. Full settings are in the supplement.

Check	Main signal	Seeds	Appendix evidence
Episode exclusion	96.7% success	3	Table 22
TopoNav	87.5% success	3	Table 24
Doorway coverage	Rank 34.62 to 0.49	1	Table 28
PUSHT boundary	Distance improves more than success	3	Appendix C.15

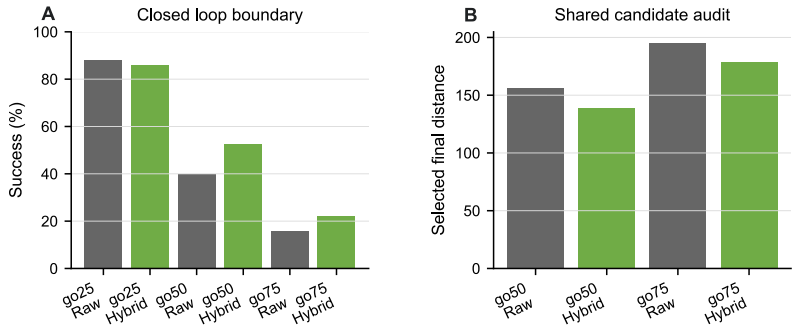

Figure 5: PUSHT boundary evidence. Closed loop success improves modestly, while SASC shows larger reductions in selected final distance.

icantly improve it. At the harder goal offsets 50 and 75, task state heads strongly improve SASC ranks and selected final distance, but closed loop success improves only modestly. The offset 50 comparison is raw 40.0% versus task state hybrid 52.7%, and the offset 75 comparison is raw 16.0% versus task state hybrid 22.0%. Figure 5 summarizes this boundary. The learned task state cost is informative, while continuous contact manipulation still depends on rollout fidelity, contact execution, action search, and recovery. Detailed mechanism tables and PUSHT sweeps are in the supplementary material.

7 SCOPE AND LIMITATIONS

The strongest control gains are established in visual navigation. In TWOROOM and TopoNav, temporal replacement works when candidate pools contain viable routes and logged trajectories cover connectivity-critical regions; rare doorway coverage fails, while balanced coverage restores ranking and success. PUSHT is more continuous: raw latent cost retains useful physical signal, and learned task state heads work best as hybrid auxiliary costs.

The layout and coverage stress tests are single checkpoint probes. Door position shifts transfer partially, horizontal wall orientation fails, and PUSHT improves candidate ranking more clearly than closed loop success. The within episode temporal label is a symmetric scalar proxy rather than a directed, budget-conditioned goal reaching value.

8 CONCLUSION

We identify a terminal ranking failure in fixed latent controllers and repair candidate selection with a temporal cost trained across the planning-relevant temporal range. The method recovers strong navigation performance when viable candidates and connectivity coverage are present. Shared candidate audits and rowspace interventions localize the failure to underweighted directions relevant to the decision. Layout and contact tests set its limits.

AI USE STATEMENT

Generative AI tools assisted literature search, code and workflow preparation, manuscript editing, and LaTeX preparation. The authors are responsible for verifying AI-assisted content and for the final claims, experiments, code, and artifacts.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/abs/1707.01495.

Junik Bae, Kwanyoung Park, and Youngwoon Lee. TLDR: Unsupervised goal-conditioned RL via temporal distance-aware representations. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2183–2204. PMLR, 2025. URL https://proceedings.mlr.press/v270/bae25a.html.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. URL https://arxiv.org/abs/2303.04137.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems*, 2018. URL https://arxiv.org/abs/1805.12114.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993. doi: 10.1162/neco.1993.5.4.613.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. *Proceedings of the 28th International Conference on Machine Learning*, 2011. URL https://icml.cc/2011/papers/323_icmlpaper.pdf.

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

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019. URL https://arxiv.org/abs/1906.05253.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-Learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tc5qisoB-C.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/e7663e974c4ee7a2b475a4775201ce1f-Abstract-Conference.html.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2170–2179. PMLR, 2019. URL https://proceedings.mlr.press/v97/gelada19a.html.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. URL https://arxiv.org/abs/1803.10122.

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*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1912.01603.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019. URL https://arxiv.org/abs/1906.08253.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 761–770. PMLR, 2020. URL https://proceedings.mlr.press/v120/lambert20a.html.

Wenyuan Li, Guang Li, Keisuke Maeda, Takahiro Ogawa, and Miki Haseyama. Predictive but not plannable: RC-aux for latent world models. *arXiv preprint arXiv:2605.07278*, 2026. URL https://arxiv.org/abs/2605.07278.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations*, 2023. URL https://iclr.cc/virtual/2023/poster/11127.

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

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

Vivek Myers, Chongyi Zheng, Anca Dragan, Sergey Levine, and Benjamin Eysenbach. Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 37076–37096. PMLR, 2024. URL https://proceedings.mlr.press/v235/myers24a.html.

Vivek Myers, Bill Chunyuan Zheng, Benjamin Eysenbach, and Sergey Levine. Offline goal-conditioned reinforcement learning with quasimetric representations. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1c5956164472d6d8123d574aa75cd063-Abstract-Conference.html.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, 2018. URL https://arxiv.org/abs/1805.08296.

Ashvin Nair, Vitchyr H. Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, 2018. URL https://arxiv.org/abs/1807.04742.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with Hilbert representations. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 39737–39761. PMLR, 2024a. URL https://proceedings.mlr.press/v235/park24g.html.

Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: Scalable unsupervised RL with metric-aware abstraction. In *International Conference on Learning Representations*, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/516593a423838642a2eb4e9c5b9c7f44-Abstract-Conference.html.

Arjun Parthasarathy, Nimit Kalra, Rohun Agrawal, Yann LeCun, Oumayma Bounou, Pavel Izmailov, and Micah Goldblum. Closing the train-test gap in world models for gradient-based planning. *arXiv preprint arXiv:2512.09929*, 2025. URL https://arxiv.org/abs/2512.09929.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1049–1065. PMLR, 2021. URL https://proceedings.mlr.press/v155/pinneri21a.html.

Vitchyr H. Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7783–7792. PMLR, 2020. URL https://proceedings.mlr.press/v119/pong20a.html.

Zhifeng Qian, Mingyu You, Hongjun Zhou, Xuanhui Xu, and Bin He. Goal-conditioned reinforcement learning with disentanglement-based reachability planning. *arXiv preprint arXiv:2307.10846*, 2023. URL https://arxiv.org/abs/2307.10846.

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

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SygwwGbRW.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320. PMLR, 2015. URL https://proceedings.mlr.press/v37/schaul15.html.

Stephen Tian, Suraj Nair, Frederik Ebert, Sudeep Dasari, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Model-based visual planning with self-supervised functional distances. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2012.15373.

Yuhai Wang, Jiawei Xia, Rongxuan Zhou, Xiao Hu, Yongliang Shi, Jing Du, and Yang Ye. PRISM: PRior-guided imagination sampling in world models. *arXiv preprint arXiv:2606.07974*, 2026. URL https://arxiv.org/abs/2606.07974.

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/a1afc58c6ca9540d057299ec3016d726-Abstract.html.

Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40 (2):344–357, 2017. doi: 10.2514/1.G001921.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=-2FCwDKRREu.

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*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12611–12620. PMLR, 2021b. URL https://proceedings.mlr.press/v139/zhang21x.html.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

A TRM Method Details

This supplement provides reproducibility details and supporting results for the main paper.

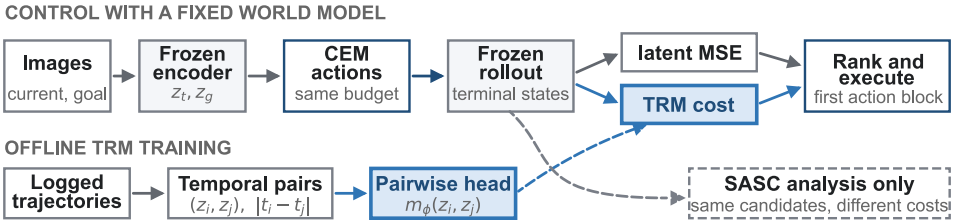

Figure 6: Control and analysis interfaces for TRM. The encoder, latent rollout model, candidate generation rule, and controller budget remain fixed. Logged temporal pairs train the TRM head. During control, the head changes only the terminal cost. During SASC, every terminal cost scores the same candidate action sequences in a fixed candidate pool.

A.1 Training Objective and Pair Sampling

Given a fixed world model encoder f_θ and latent rollout model F_θ, we train a pairwise scalar head $m_\phi(\mathbf{z}_i, \mathbf{z}_j)$ over encoded latent states. The default feature map is

$$\psi(\mathbf{z}_i, \mathbf{z}_j) = [\mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_i - \mathbf{z}_j, |\mathbf{z}_i - \mathbf{z}_j|]\,. \tag{7}$$

The head is an MLP with two hidden layers.

$$m_\phi(\mathbf{z}_i, \mathbf{z}_j) = \mathrm{Softplus}(\mathrm{MLP}_\phi(\psi(\mathbf{z}_i, \mathbf{z}_j)))\,, \tag{8}$$

where MLP_ϕ has two hidden layers with 256 units each and SiLU nonlinearities. The Softplus output enforces nonnegative predicted distance. We train with Smooth-L1 loss on scaled labels.

$$\min_\phi \; \mathbb{E}_{(i,j)}\, \ell_{\mathrm{Huber}}\left(m_\phi(\mathbf{z}_i, \mathbf{z}_j),\; y_{ij}/s\right), \tag{9}$$

with scale $s = 224$, the TwoRoom image and coordinate scale used only for loss conditioning.

For temporal sampling in TwoRoom, we draw an episode e of length L_e and choose Δ uniformly from $[1, L_e - 1]$. The short horizon ablation instead uses $[1, \min(L_e - 1, \Delta_{\max})]$. We then choose t uniformly from $[0, L_e - \Delta - 1]$ and place rows (e, t) and $(e, t + \Delta)$ in random order. The label is $y_{ij} = \Delta$. Balanced sampling allocates comparable numbers of training pairs to different ranges of temporal separation.

Full episode sampling includes larger temporal separations than the short horizon setting. Table 30 changes trajectory coverage, pair count, and maximum temporal separation separately.

The label is symmetric, while the default feature map does not impose exact symmetry. Random input order during pair construction is therefore an augmentation rather than a constraint. We report the difference between the two input orders below and also evaluate an explicitly symmetric variant,

$$m_{\mathrm{sym}}(\mathbf{z}_i, \mathbf{z}_j) = \tfrac{1}{2}\left[h_\phi(\mathbf{z}_i, \mathbf{z}_j) + h_\phi(\mathbf{z}_j, \mathbf{z}_i)\right]. \tag{10}$$

We use "pairwise cost" for the default head and reserve "symmetric cost" for this averaged form.

Table 7: TRM head implementation details.

Component	Setting
Input features	$[\mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_i - \mathbf{z}_j, \lvert\mathbf{z}_i - \mathbf{z}_j\rvert]$
Architecture	MLP, two 256-unit hidden layers
Nonlinearity	SiLU hidden layers, Softplus scalar output
Training update	AdamW
Learning rate	10^{-3}
Weight decay	10^{-4}
Batch size	1024 pairs
Loss	Smooth-L1 on scaled distances
Distance scale	224
Main train pairs	100,000
Validation pairs	Validation pair split recorded per run
Shuffled control	Same architecture/data, permuted training labels only

A.2 Controller Use

At evaluation time, TRM replaces or augments only the terminal cost. For each CEM replan, the controller proposes candidate action sequences $\mathbf{a}^{(i)}_{0:H-1}$. The fixed world model rolls each sequence forward to a predicted terminal latent state $\hat{\mathbf{z}}^{(i)}_{t+H}$. The goal image is encoded once as $\mathbf{z}_g$. The replacement TRM cost is

$$c_{\mathrm{TRM}}(\mathbf{a}^{(i)}_{0:H-1}) = m_\phi\left(\hat{\mathbf{z}}^{(i)}_{t+H}, \mathbf{z}_g\right). \tag{11}$$

For the hybrid terminal cost, we standardize the raw latent and learned cost values over the candidate action sequences in the current CEM iteration.

$$c^{(i)}_{\mathrm{hyb}} = \frac{c^{(i)}_{\mathrm{lat}} - \mu_{\mathrm{lat}}}{\sigma_{\mathrm{lat}} + \epsilon} + \lambda \frac{c^{(i)}_{\mathrm{TRM}} - \mu_{\mathrm{TRM}}}{\sigma_{\mathrm{TRM}} + \epsilon}. \tag{12}$$

This standardization uses the current candidate pool. It prevents either component from dominating the terminal cost only because its numerical scale is larger.

We use *head* for the learned function m_ϕ and *terminal cost* for the scalar rule that assigns a cost to each candidate action sequence. CEM uses these costs to refit its search distribution and returns the final mean plan. The environment executes the receding-horizon prefix specified by the task protocol.

B Diagnostic Protocols

B.1 Selection Audit with Shared Candidates

For each episode, SASC fixes one candidate pool, and every terminal cost scores every candidate action sequence in that pool. For candidate action sequence i, $c(i)$ is the cost assigned by the tested terminal cost and $c^\star(i)$ is a task-space diagnostic cost for the same candidate. In TwoRoom, the diagnostic endpoint is obtained by rolling the candidate through the known TwoRoom geometry; TopoNav uses exact grid dynamics; and PushT uses simulator-executed final task state distance as defined in Section C.15. SASC measures whether changing only the terminal cost changes which candidate action sequence the controller selects from the fixed candidate pool. To compute candidate order Spearman, we rank the candidate action sequences by $c(i)$ and $c^\star(i)$ within the episode, use average ranks for ties, and compute the Pearson correlation between the two rank vectors. Both quantities are costs, so a positive value means that the tested terminal cost tends to assign lower costs to candidate action sequences with lower diagnostic costs. We first compute each value within an episode and then average over the episodes or seeds stated in the table caption. The audit reports four quantities.

Table 8: Complete TWOROOM controller configuration. The CEM seed and evaluation seed are recorded for each run. The controller uses standardized actions internally, and the environment enforces physical action bounds.

Controller item	Fixed setting for TWOROOM
Action space	Two continuous velocity components. The environment clips each component to $[-1, 1]$ and multiplies it by the sampled agent speed. The released evaluation uses the fixed speed recorded in the environment configuration.
Planning horizon	5 model steps with an action block of 5, giving 25 environment actions per candidate action sequence.
Replanning	Receding horizon 5. The environment therefore executes all 25 actions from the final CEM mean plan before the controller replans. A 50-step evaluation makes two controller calls.
CEM initialization	Zero mean and unit standard deviation in standardized action coordinates. The first candidate in every iteration is the current mean.
CEM update	300 candidates, 10 iterations, 30 elites. The next mean and standard deviation are the unweighted sample mean and standard deviation of the elites.
Smoothing and variance floor	No smoothing coefficient and no explicit variance floor.
Warm start	The implementation flag is enabled, but it is inactive here because receding horizon equals planning horizon and no unused suffix remains.
Clipping	CEM does not clip actions. The environment applies action clipping after inverse standardization.
Success and termination	TWOROOM success at a Euclidean goal distance below 16 pixels. Evaluation continues for the fixed 50-step reporting budget while retaining the first success step.
World model rollout	Frozen encoder and dynamics. The terminal cost uses the final predicted latent after 5 model steps.
SASC candidate pool	300 Gaussian candidate action sequences per episode with candidate seed 123. The candidate pool is generated separately from the final CEM iteration, and every terminal cost scores the same candidate action sequences.

Table 9: Pseudocode for training and using TRM.

Train TRM
1. Freeze the world model encoder f_θ and dynamics F_θ.
2. Sample state pairs from logged trajectories using the temporal horizon rule above.
3. Encode both observations, $\mathbf{z}_i = f_\theta(o_i)$ and $\mathbf{z}_j = f_\theta(o_j)$.
4. Build pair features $[\mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_i - \mathbf{z}_j, |\mathbf{z}_i - \mathbf{z}_j|]$.
5. Train m_ϕ to predict scaled temporal distance with Smooth-L1 loss.

Use TRM **in latent** CEM **planning**
1. Encode the current observation and goal observation.
2. Sample candidate action sequences with the same CEM controller.
3. Roll each candidate forward through the fixed world model.
4. Use m_ϕ to compute the terminal cost of each candidate action sequence from its predicted terminal latent state.
5. CEM uses the replacement or hybrid costs to update its distribution and returns the final mean plan.
6. The environment executes the configured receding-horizon prefix of that plan.

- **Candidate order Spearman**. This is the Spearman correlation between $c(i)$ and $c^\star(i)$ over the candidate action sequences. A higher value means that the tested terminal cost orders the sequences more like the diagnostic task cost.
- **Oracle best rank**. This is the percentile rank that c assigns to the candidate action sequence $\arg\min_i c^\star(i)$. A lower value means that the terminal cost places the candidate action sequence with the lowest diagnostic cost nearer the top.
- **Selected diagnostic distance**. This is the task-space distance assigned to the candidate selected by the terminal cost under the task-specific diagnostic rollout above. A lower value means that the changed ranking selects a candidate with a better diagnostic outcome.
- **Top-k diagnostic cost**. When reported, these are the diagnostic costs of the candidate action sequences that the terminal cost places in its top k positions.

This audit separates two failure cases. The candidate pool may contain no candidate action sequence with a low diagnostic cost. Alternatively, the candidate pool may contain such a sequence while the tested terminal cost assigns it a poor rank. The audit also identifies settings in which the terminal cost improves the ranking but execution, contact, or rollout error still prevents success.

Table 10: SASC audit on the high distance evaluation set for LEWM seed 3072. The audit measures candidate ranking, and lower oracle best rank percentile is better.

Terminal cost	ρ Euc.	ρ Geo.	Best rank
Raw latent MSE	0.021	0.018	31.71
Decoded XY	0.876	0.860	0.59
TRM temporal labels	0.720	0.729	3.86
TRM shuffled labels	0.105	0.119	34.00

B.2 Subspace Projection Audit

The XY rowspace intervention uses the linear probe matrix W to define

$$P_W = W^\top (WW^\top)^\dagger W. \tag{13}$$

For each candidate action sequence, let $d = \hat{\mathbf{z}}_{t+H} - \mathbf{z}_g$ be its predicted terminal difference. The XY rowspace terminal cost is $\|P_W d\|_2^2$, and the residual terminal cost is $\|(I - P_W)d\|_2^2$. On the distance controlled set with a budget of 50 actions, the XY rowspace terminal cost reaches 97.5%, 92.5%, and 82.5% success for seeds 3072, 3073, and 3074. The residual terminal cost reaches 2.5%, 2.5%, and 0.0% success for the same seeds.

Table 11: Per seed subspace intervention on the distance controlled set with a budget of 50 actions.

Seed	Latent MSE	Decoded XY	XY rowspace	Residual only	Rowspace MSE share
3072	2.5	95.0	97.5	2.5	0.006
3073	2.5	95.0	92.5	2.5	0.0069
3074	0.0	92.5	82.5	0.0	0.005

B.3 Direct Attribution of the TRM Terminal Cost

The intervention above shows that the XY rowspace can support control, but it does not show how TRM uses that rowspace. For each candidate action sequence, the frozen world model predicts a terminal latent state $\hat{\mathbf{z}}_{t+H}$. We write its difference from the goal latent as $d = \hat{\mathbf{z}}_{t+H} - \mathbf{z}_g$. We then construct two diagnostic terminal latent states. The first is $\mathbf{z}_g + P_W d$, which retains only the part of the candidate difference in the XY rowspace. The second is $\mathbf{z}_g + (I - P_W)d$, which retains only the residual part. The same trained head evaluates each constructed terminal latent state against $\mathbf{z}_g$ and assigns the resulting cost to the same candidate action sequence. The candidate action sequences,

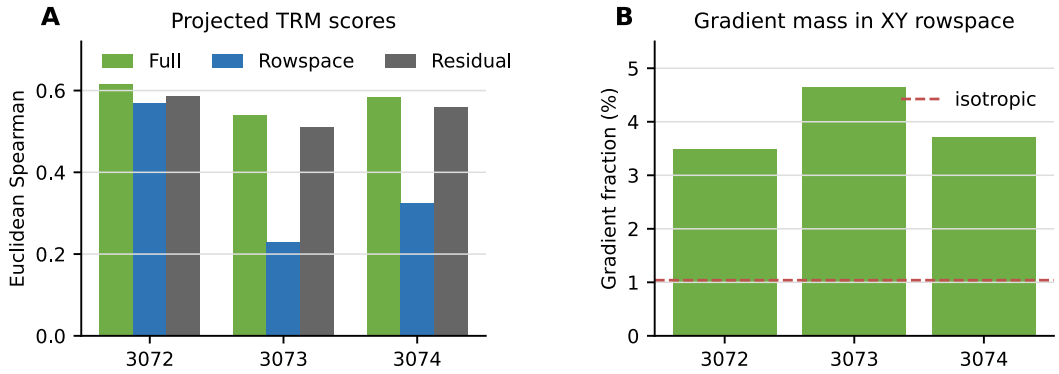

Figure 7: Direct TRM attribution visualized from Table 12. (A) Terminal costs computed from the full difference, XY rowspace difference, and residual difference each preserve some Euclidean ordering. (B) A larger fraction of the squared terminal cost gradient lies in the XY rowspace than the isotropic baseline predicts.

predicted terminal latent states, and head parameters remain fixed. We also differentiate the original TRM terminal cost with respect to d and report

$$r_W = \frac{\|P_W \nabla_d m_\phi(\mathbf{z}_g + d, \mathbf{z}_g)\|_2^2}{\|\nabla_d m_\phi(\mathbf{z}_g + d, \mathbf{z}_g)\|_2^2}. \tag{14}$$

The audit uses heads trained with episode exclusion. Every head scores the same 300 candidate action sequences in the high distance candidate pool for each episode.

Table 12: Direct TRM attribution on the high distance candidate pools. ρ is Spearman correlation with the oracle Euclidean terminal cost. Rank is the percentile assigned to the candidate action sequence with the lowest oracle Euclidean cost, where lower is better. The XY rowspace has dimension 2 in a 192 dimensional latent, so an isotropic gradient baseline is 1.04%. The observed gradient fraction is 3.4 to 4.5 times this baseline.

Seed	Full ρ	Row ρ	Resid. ρ	Full rank	Row rank	Resid. rank	r_W
3072	0.615	0.569	0.587	9.44	14.90	9.14	3.49%
3073	0.539	0.228	0.511	5.23	43.04	6.15	4.64%
3074	0.583	0.325	0.559	5.59	32.76	5.65	3.70%

The gradient result shows that the TRM terminal cost is more sensitive to changes in the XY probe rowspace than the isotropic baseline predicts. The costs computed from the projected differences also show that TRM does not use only this two dimensional rowspace. Residual features retain useful ordering, and the cost computed only from the rowspace is weaker for two seeds. We therefore interpret TRM as giving more weight to latent directions that carry position while still using information outside those directions.

B.4 Swap Consistency and Feature Groups

The default head is not exactly symmetric. On the high distance candidate pools, the mean absolute difference between $m_\phi(\mathbf{z}_i, \mathbf{z}_j)$ and $m_\phi(\mathbf{z}_j, \mathbf{z}_i)$ is 4.92, 6.13, and 5.35 temporal steps for seeds 3072, 3073, and 3074. These values are 6.4%, 7.7%, and 6.9% of the mean terminal cost magnitude. We next use the same trained head to score the same candidate action sequences while changing only its input features. For each row in Table 13, we retain the named feature blocks and set every other block to zero before the head assigns a cost. The controller then ranks the unchanged candidate action sequences by these costs. We do not retrain the head or select a new checkpoint for this intervention.

Table 13: Feature group intervention on temporal heads trained without the evaluation episodes. ρ is Spearman against oracle Euclidean terminal cost and Rank is the oracle Euclidean best percentile. Absolute differences alone do not preserve the candidate ordering. Signed and absolute differences help seed 3073 but not the other two seeds. The full features work across all three seeds, and averaging the two input orders improves the ranking for every seed.

Input to fitted head	Seed 3072		Seed 3073		Seed 3074	
	ρ	Rank	ρ	Rank	ρ	Rank
$\lvert z_i - z_j \rvert$ only	0.006	27.47	-0.028	36.80	0.074	45.93
$[z_i - z_j, \lvert z_i - z_j \rvert]$ only	0.037	42.45	0.555	8.27	0.086	19.07
Full ordered features	0.615	9.44	0.539	5.23	0.583	5.59
Full features with swap symmetrization	0.738	3.40	0.592	3.33	0.686	3.24

B.5 Temporal Labels and Geodesic Distance

The offline cache contains 10,000 episodes and 632,749 observations. Episode length ranges from 2 to 101 steps, with median 67 and mean 63.3. The cache records one action and one state per environment step. Consecutive displacement has mean 5.92 pixels and median 6.19 pixels. Only 0.037% of steps move less than 0.1 pixels, so pauses are rare. Temporal separation is not assumed to equal a shortest path. It is treated as a supervision signal that depends on the policy, and its relation to task geometry is measured below.

Table 14: Correlation between logged temporal separation and oracle TwoRoom geodesic distance. Pairs use the same episode exclusion as the high distance set audit. The aggregate relation partly reflects horizon and doorway coverage. Correlation decreases when pairs are restricted to narrower horizon bands.

Pair subset	Pairs	Pearson	Spearman
All	100,000	0.855	0.883
Same room	70,976	0.829	0.854
Doorway required	29,024	0.772	0.779
$\Delta \in [1, 10]$	28,067	0.653	0.670
$\Delta \in [11, 25]$	23,777	0.398	0.388
$\Delta \in [26, 50]$	24,346	0.387	0.380
$\Delta \in [51, 75]$	14,774	0.271	0.266
$\Delta \in [76, 100]$	9,036	0.203	0.203

C Experimental Details

C.1 Data Appendix and Public Sources

The experiments use three data sources. TwoRoom is the main navigation setting. It is drawn from the public LeWorldModel and stable world model ecosystem (Maes et al., 2026b;a). The exact cache is `datasets/tworoom.h5`, 10,175,381,504 bytes. Its SHA256 is `ee3218700ddbcc2d0e8c810a357e5ce2d3b610323f44a52bbf91d4296c4a98a7`. It contains 632,749 rows from 10,000 episodes. The smaller default cache with 493 rows is not used. The frozen checkpoint family is `tworoom_aligned_e10_h1_go25_b50`, with latent dimension 192, 512 projections, and seed suffix 3072, 3073, or 3074. We use these checkpoints to build fixed evaluation sets, logged temporal pairs, and SASC candidate pools.

The immutable upstream release snapshots are the TwoRoom dataset revision https://huggingface.co/datasets/quentinll/lewm-tworooms/tree/6903a2de048b13819d812da0b4dd661290bc01e4 and model revision https://huggingface.co/quentinll/lewm-tworooms/tree/77adaae0bc31deab21c93740d1f8bb947cd0bdec. The corresponding PushT snapshots are dataset revision https://huggingface.co/datasets/quentinll/lewm-pusht/tree/655cd446b9929369d7d406001da85c15d1457850 and model revision https://huggingface.co/

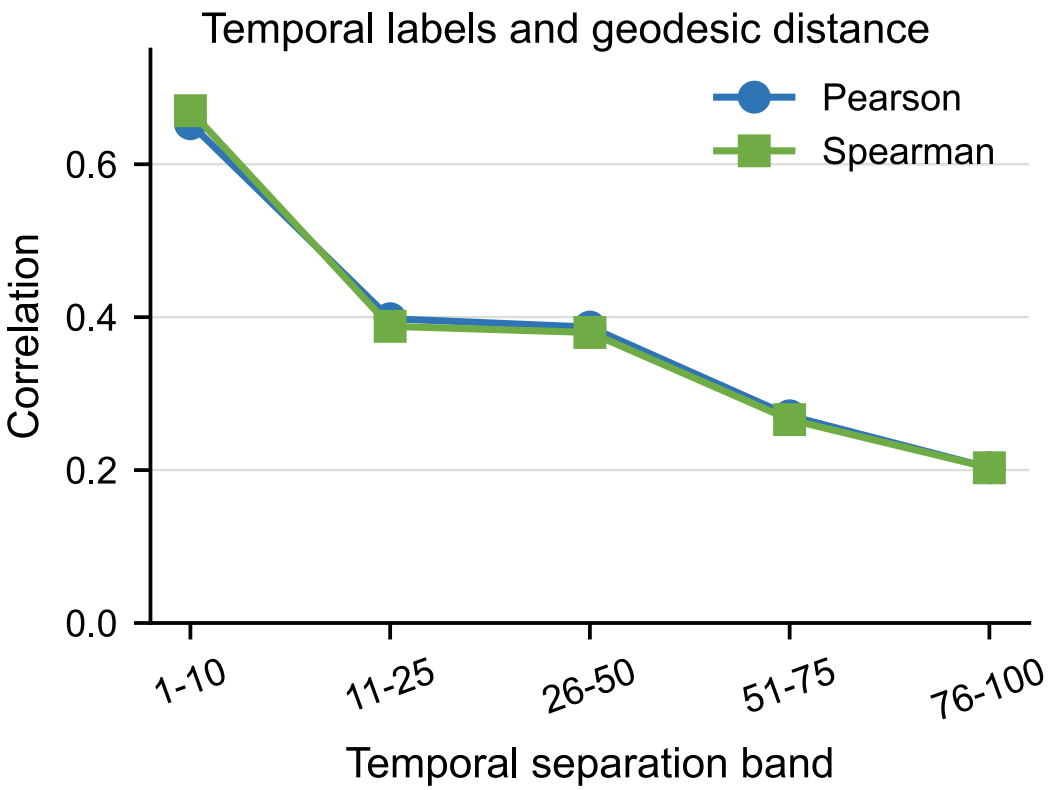

Figure 8: Temporal labels and oracle geodesic distance by separation band. The aggregate correlation is strong across all pairs. Correlation decreases within narrower separation bands.

quentinll/lewm-pusht/tree/22b330c28c27ead4bfd1888615af1340e3fe9052. These revision links identify the upstream source releases; the byte count and SHA256 above identify the exact expanded TwoRoom cache used by our runs.

TopoNav is a synthetic grid navigation diagnostic introduced in this paper. Its saved result directory contains the run configuration, evaluation set, summaries for each seed and across seeds, closed loop episode rows, and SASC candidate rows. Section C.11 gives the exact geometry, rendering, state representation, logged pair construction, and controller settings. The released TopoNav code and generated records use the MIT license.

PushT uses the public PushT task introduced with Diffusion Policy (Chi et al., 2023) through the stable world model release (Maes et al., 2026a). The dataset key is `pusht_expert_train`. It is distributed as `datasets/pusht_expert_train.h5.zst`. The checkpoint key is `pusht/lewm`. The release resolves it to `checkpoints/pusht/lewm_object.ckpt`. The paper adds evaluation sets for goal offsets 25, 50, and 75, pair head training records, hybrid weight sweeps, SASC candidate pools, and rollout calibration summaries.

The accompanying release contains the new evaluation sets, logged pair metadata, TopoNav configuration and outputs, candidate audit files, and table producing summaries. The larger TwoRoom cache and PushT data remain distributed through their public source releases and the accompanying Code and Data Supplement records their exact identifiers. No experiment in the paper requires a private dataset. Public datasets drawn from prior work are cited above.

The new data products introduced by this paper are the TopoNav trajectories, fixed evaluation sets, logged pair indices and labels, SASC candidate audit records, and aggregate tables derived from those records. TopoNav trajectories are generated from the grid and transition rules in Section C.11. The other new records contain indices, model outputs, controller costs, or task metrics derived from the cited public sources. No human participants, personal data, or manual annotation are used. The

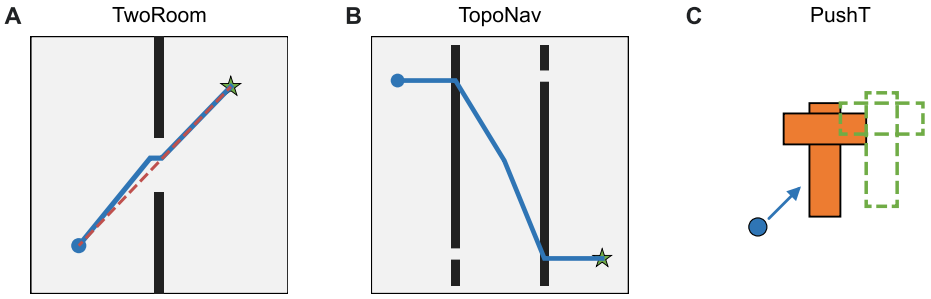

Figure 9: Task suite geometry. TWOROOM tests whether a terminal cost can prefer the doorway route over a blocked straight line. TopoNav uses a separate grid topology with exact graph distances. PUSHT requires object motion through contact, so a better ranking of candidate action sequences need not produce successful closed loop control.

TopoNav code and generated records use the MIT license. The TWOROOM and PUSHT observations and checkpoints retain the licenses of their source releases, and we do not relicense those upstream assets.

We do not retrain the LEWM or PLDM world models. Reproducing our comparisons begins from the cited frozen checkpoints. The source releases define the model architectures and their original training procedures. This supplement specifies the checkpoint family, latent interface, controller settings, and added TRM head that affect the reported experiments.

C.2 TWOROOM EVALUATION SETS AND FAILURE TAXONOMY

The balanced evaluation set contains 40 episodes, with 20 goals in the same room and 20 goals across the wall. The distance controlled evaluation set uses the same counts and draws both groups from the same Euclidean distance range. The high distance evaluation set contains 100 episodes, with 50 goals in the same room and 50 goals across the wall. We subsequently refer to them as the balanced, distance controlled, and high distance sets. Among the 50 goals across the wall, 47 require the doorway.

The balanced and distance controlled sets each contain 40 episodes. For compact table headings, B50 and B150 denote the balanced sets with action budgets of 50 and 150, while D50 and D150 denote the distance controlled sets with the same budgets.

Evaluation set construction computes Euclidean and approximate geodesic distance from the recorded start and goal coordinates. Distance controlled evaluation sets draw equal same room and cross wall counts from the same Euclidean distance range. The high distance set uses geodesic distances from 120 to 205 pixels and requires a margin of 5 to 30 steps relative to the action budget of 50. A pair is doorway required when the endpoints lie on opposite wall sides and the straight segment intersects the solid wall outside every door interval.

We use the following failure taxonomy. *Success* occurs when the environment reaches a goal distance below 16 pixels. *Wrong room* means the final agent location remains on the wrong side of the wall. *Stuck at wall* means the final position lies within one speed unit of the wall and outside a door interval. *Same room not precise* and *crossed door not precise* are near miss categories used in episode reports.

C.3 DATA SPLITS AND LEAKAGE CONTROLS

Table 15 summarizes the split boundary for each experiment family. The central rule is that evaluation outcomes are never used as TRM training labels. Temporal TRM heads see logged state pairs and temporal separations within the same episode. They do not see closed loop success labels, SASC oracle costs, or candidate outcomes. Some baselines intentionally use stronger supervision, such as TWOROOM oracle geodesic distance or PUSHT task state distance. Those rows are marked as diagnostic terminal cost baselines rather than the default temporal TRM recipe.

Table 15: Split and leakage control summary. Temporal TRM uses logged temporal separation rather than evaluation outcomes. Oracle geodesic and task state labels train diagnostic terminal costs and are not the default temporal supervision.

Experiment family	Training labels	Evaluation source	Leakage control and interpretation
TWOROOM temporal TRM	Logged pairs from the same episode in the fixed offline cache, labeled by temporal separation	Fixed high distance and distance controlled sets, closed loop rollouts, and SASC candidate pools	No evaluation success, candidate oracle cost, or closed loop outcome is used for training. Individual observations may come from the same cache. The audit below removes every episode that contains an evaluation observation from the high distance set.
TWOROOM audit after excluding evaluation episodes	Logged pairs from the same episode after excluding every source episode that contains a start or goal from the high distance set	Same high distance set, checkpoints, controller budget, and SASC candidate pools	All 200 evaluation start and goal rows and their 193 source episodes are excluded. Measured row and episode overlap are both zero.
TWOROOM shuffled	Same state pair exposure and architecture as temporal TRM with permuted labels	Same evaluation sets, checkpoints, controller budget, and candidate pools as temporal TRM	Controls for architecture, optimization, and state exposure. This comparison tests whether the repair depends on temporal label structure rather than only on fitting a scalar function.
TWOROOM geodesic baselines	Logged/cache state pairs labeled by oracle geodesic distance	Same high distance set and SASC candidate pools	These diagnostic baselines test whether a stronger terminal cost can express the missing ordering. They do not test whether temporal labels alone are sufficient.

Experiment family	Training labels	Evaluation source	Leakage control and interpretation
Coverage diagnostic	Random, mixed, limited doorway coverage, or balanced doorway coverage logged pair sampling under the same checkpoint	Same high distance set and controller budget	The evaluation set is fixed while the logged pair coverage changes. Recovery after adding doorway coverage is evidence that the learned terminal cost depends on coverage near the doorway.
Layout transfer	Terminal cost heads trained on the original layout. Evaluation images and the rollout environment are rendered again after changing the wall or door parameters	Evaluation sets and geometry rebuilt for each changed layout	Tests transfer to new visual layouts. Door position shifts transfer partially. A horizontal wall is a boundary case.
TopoNav	Pixel gridworld logged pairs. Exact evaluation start and goal pairs are excluded from train and validation pairs	Fixed TopoNav evaluation set, exact frozen dynamics, and the candidate distributions specified in Section C.11	Exact evaluation pairs are excluded from training. Every terminal cost scores the same sampled pool within each SASC evaluation unit.
PUSHT	Task state pair labels for learned diagnostic heads	Threshold crossing success and SASC final distance audits on fixed candidate pools	Boundary case. Ranking and selected distance improvements are interpreted separately from contact execution and recovery.

C.4 LINEAR XY PROBE DETAILS

The XY probe uses 20,000 rows sampled without replacement from the public cache. We randomly assign 16,000 sampled rows to training and 4,000 to testing. Ridge regression uses $\alpha = 1$. Because rows rather than complete episodes are split, this probe shows that position is decodable from test observations but does not measure generalization to new episodes. The temporal head audit in Section C.10 excludes complete evaluation episodes.

Table 16: Linear XY probe results. RMSE is in pixels. Rows are split after sampling, and the saved probe file records the exact row indices.

Seed	Test RMSE	Test R^2	Train/Test rows
3072	1.776	0.99818	16k/4k
3073	1.870	0.99802	16k/4k
3074	1.855	0.99801	16k/4k

C.5 Raw Latent Controller Budget

Table 17: Raw latent CEM budget comparison for LeWM seed 3072 on the distance controlled set with a budget of 50 actions, using 40 episodes. Same room and cross wall success are 5.0% and 0.0% with the default budget. They are 0.0% and 5.0% with the larger budget. Aggregate success is 2.5% for both budgets.

Setting	Samples	Iter.	Elite	Success
Default	300	10	30	2.5
Larger budget	1000	20	100	2.5

C.6 Implementation and Reproduction Scope

The accompanying Code and Data Supplement contains the executable workflows and saved records used to reproduce the reported data products. This PDF records the scientific protocol, while the package README and reproduction guide provide exact entry points, commands, file locations, and completion checks. The separation avoids duplicating an operational manual without removing information needed for reproduction.

The workflow resolves the cited cache and frozen checkpoint for TwoRoom or PushT, or generates TopoNav from the geometry and transition rules in Section C.11. It then applies the recorded episode or start and goal exclusions before pair sampling, encodes observations with the frozen encoder, saves the sampled pair indices, trains each pairwise head for the fixed epoch budget, and retains the checkpoint with the lowest validation loss. No closed loop result is used for head selection.

Evaluation loads the fixed manifests, checkpoints, controller settings, and candidate seeds recorded for each run. Closed loop scripts save one row per episode. SASC scripts save scores and ranks for individual candidates from a shared pool within each evaluation unit. Aggregation reads these records directly and stops on missing or malformed rows rather than imputing results. Each run record distinguishes the task, frozen checkpoint, pair sampling seed, head initialization seed, evaluation seed, candidate seed, evaluation set, terminal cost, controller settings, and input artifacts. The package manifest and result map connect these records to every reported table and figure.

C.7 Final Settings and Checkpoint Selection

Table 18 collects the final settings used by the reported experiments. Pairwise heads retain the checkpoint with the lowest validation loss from the fixed training budget, and controller budgets remain fixed within each comparison. Task-specific settings not shared across all experiments are given in their corresponding subsections.

C.8 Compute Infrastructure

The main TwoRoom and TopoNav runs were executed on Server A. Server A runs Linux 5.4.0 on x86_64, with two NVIDIA RTX 6000 Ada GPUs of 49,140 MiB each, NVIDIA driver 570.133.07, CUDA 12.8, two AMD EPYC 7763 sockets for 256 logical CPUs, and 503 GiB system memory. The recorded Python environment uses Python 3.10.20, PyTorch 2.10.0+cu128, NumPy 1.24.4, SciPy 1.15.3, scikit-learn 1.7.2, and h5py 3.16.0. Each output directory stores an environment snapshot with the Python executable, environment variables, and package freeze.

Supporting exploratory and boundary runs were also executed on Server B. Server B runs Linux 5.4.0 on x86_64, with two Tesla V100 PCIe GPUs of 32,768 MiB each, NVIDIA driver 550.54.15, CUDA 12.4, two Intel Xeon Gold 6248 sockets for 80 logical CPUs, and 503 GiB system memory. These supporting runs use the same code artifact and saved run records. The paper tables report only results whose run records identify the checkpoint, seed, controller budget, and input artifact paths.

Recorded runtime and compute. Table 19 summarizes stages whose timing records include `elapsed_sec`. The totals cover timed computation only; queue time, environment setup, cache

Table 18: Final settings used in the reported experiments. Task-specific controller and evaluation settings are listed in the corresponding protocol sections.

Setting	Reported value
TWOROOM pair head	Two 256-unit hidden layers, SiLU, Softplus output
Optimizer	AdamW, learning rate 10^{-3}, weight decay 10^{-4}
Batch size	1024 pairs
Main TWOROOM training set	100,000 balanced temporal pairs over the full episode range
Validation set	10,000 independently sampled pairs; episode exclusions applied when specified
TWOROOM controller	300 candidates, 10 CEM iterations, 30 elites
High-distance evaluation	50 executed actions
Main seeds	3072, 3073, 3074
Checkpoint selection	Lowest validation loss within the fixed training budget

download, and stages without recorded elapsed time are not included. Each timed process used one GPU, so total GPU hours are computed by summing wall time and dividing by 3,600.

Table 19: Recorded runtimes and single GPU compute. The number of runs is the number of separately timed seed and terminal cost records in each row. The table includes only stages with recorded elapsed times.

Timed process	Runs	Wall time per run	Total GPU h	Timing record
TWOROOM temporal head, episode exclusion	3	6.98–7.39 min	0.362	Head `elapsed_sec`
TWOROOM shuffled head, episode exclusion	3	7.04–7.23 min	0.355	Head `elapsed_sec`
TWOROOM ranking audit, fixed pools	9	7.81–8.05 s	0.020	Ranking `elapsed_sec`
PUSHT offset 50, raw latent	3	5.80–6.12 min	0.300	Evaluation `elapsed_sec`
PUSHT offset 50, learned hybrid, $w = 1$	3	10.76–11.34 min	0.557	Evaluation `elapsed_sec`
PUSHT offset 50, shuffled hybrid, $w = 1$	3	11.37–12.13 min	0.594	Evaluation `elapsed_sec`
PUSHT offset 75, raw latent	3	8.40–8.47 min	0.422	Evaluation `elapsed_sec`
PUSHT offset 75, learned hybrid, $w = 1$	3	15.81–17.35 min	0.819	Evaluation `elapsed_sec`
PUSHT offset 75, shuffled hybrid, $w = 1$	3	15.90–17.15 min	0.817	Evaluation `elapsed_sec`

C.9 REPORTING UNITS, REPEATS, AND TERMINATION

A closed loop episode is the unit used for success and the TWOROOM failure taxonomy. For seed k with n_k episodes, we compute the success percentage as

$$s_k = \frac{100}{n_k} \sum_{e=1}^{n_k} \mathbb{1}[\text{episode } e \text{ succeeds}]. \tag{15}$$

When a table reports mean±sample standard deviation over $K = 3$ seeds, it uses

$$\bar{s} = \frac{1}{K}\sum_{k=1}^{K} s_k, \qquad \mathrm{sd}(s) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}(s_k - \bar{s})^2}. \tag{16}$$

Candidate action sequences are not treated as independent experimental runs. SASC computes candidate ranks within each episode. It then averages the values over the episodes or seeds named in the table caption. The temporal label analysis treats each sampled pair as one observation and reports the number of pairs used.

For the detailed high distance set SASC audit, we form 95% percentile intervals from 100,000 bootstrap resamples paired by episode with deterministic analysis seed 20260726. Each resample draws 100 episodes with replacement and retains all terminal cost measurements from the same episode, so it never treats the 300 candidate action sequences as independent observations. For TopoNav success, a hierarchical paired bootstrap first resamples the three head seeds and then resamples the 120 paired episodes inside each selected head seed. Table 20 reports these intervals.

Table 20: Means and 95% paired bootstrap intervals. High distance evaluation resamples episodes. TopoNav uses the hierarchical procedure described in the text, which resamples head seeds and then paired episodes. Lower rank and selected geodesic are better.

Audit quantity	Raw latent MSE	Temporal TRM	Paired difference, TRM minus raw
High distance evaluation SASC ρ_{geo}	0.018 $[-0.053, 0.088]$	0.729 $[0.695, 0.759]$	0.711 $[0.644, 0.779]$
High distance evaluation oracle best rank, %	31.71 $[26.49, 37.11]$	3.86 $[2.37, 5.62]$	-27.85 $[-33.63, -22.16]$
High distance evaluation selected geodesic	168.81 $[163.02, 174.64]$	134.79 $[130.41, 139.40]$	-34.02 $[-39.40, -28.69]$
TopoNav success, %	15.0 $[11.39, 18.89]$	87.5 $[82.50, 92.22]$	72.5 $[66.39, 78.33]$

As a formal check across seeds, we also apply an exact two-sided paired sign-flip permutation test to the three success differences across checkpoints. The high distance set differences after episode exclusion are 83, 91, and 95 percentage points; the TopoNav differences are 69.17, 71.67, and 76.67 percentage points. Both tests give $p = 0.25$ because only $2^3 = 8$ sign assignments are available and 0.25 is the smallest attainable two-sided value with three consistently signed pairs. We therefore report the tests as a transparent small-K check, not as evidence of conventional significance. The primary evidence remains the paired effect sizes, intervals, fixed evaluation sets, negative controls, and candidate pool audits.

The pairwise heads train for a fixed number of epochs and retain the epoch with the lowest validation loss. No closed loop success value is used to stop training or select a checkpoint. A TwoRoom episode succeeds when the environment crosses the distance threshold. The evaluation records the first success step and continues through the fixed budget of 50 executed actions. An episode that does not cross the threshold within that budget is a failure, and its final state determines the failure category. A TopoNav episode stops when the agent reaches the exact goal cell or after 90 executed actions. Failure to reach the goal within 90 executed actions counts as failure. For PushT, threshold success records any threshold crossing during the episode. Final state success instead applies the terminal thresholds given in the PushT Details subsection. Goal offsets 25 and 50 use 50 executed actions, and goal offset 75 uses 75 executed actions.

C.10 TwoRoom Audit after Excluding Evaluation Episodes

Table 22 tests whether episode overlap between training and evaluation explains the high distance set result. Before sampling training or validation pairs, we exclude all 193 source episodes that contain any of the 200 evaluation start and goal rows. The encoder, dynamics, high distance set, controller, CEM budget, and candidate pools remain fixed. Only the episodes available for temporal pair training change.

Table 21: Experimental units and repeats used by the main results and supporting analyses. The table separates repeated performance results from single checkpoint diagnostics and identifies the scope of each uncertainty statement.

Experiment	Unit and repeats	Seed scope	Reported uncertainty and role
High distance TWOROOM evaluation	100 episodes for each of three frozen checkpoints	3072, 3073, 3074	Success uses mean±sample standard deviation; the paired checkpoint differences also receive the exact sign-flip test above.
Distance controlled horizon test, budget 50	40 episodes for each of three frozen checkpoints	3072, 3073, 3074	Success uses mean±sample standard deviation.
Detailed SASC audit on the high distance set	100 episodes with 300 candidate action sequences per episode	Checkpoint 3072, candidate seed 123	The 95% bootstrap intervals above are paired by episode.
TopoNav	120 fixed start and goal pairs for each of three heads	Base seed 608 and head seeds 3072, 3073, 3074	Hierarchical paired 95% bootstrap intervals and an exact sign-flip test across seeds are reported above.
Coverage and layout checks	100 episodes from the high distance set for each reported setting	Checkpoint 3072	These diagnostics use one checkpoint and do not establish variation across checkpoints.
PUSHT, goal offset 25	50 episodes	Evaluation seed 3072	This is a single seed boundary result.
PUSHT, goal offsets 50 and 75	50 episodes for each of three evaluation seeds	3072, 3073, 3074	Tables report per seed values or means. No formal significance test is claimed.

Table 22: High distance evaluation audit with episode exclusion. Success is the mean±sample standard deviation over seeds 3072, 3073, and 3074. The SASC columns are means over the same seeds. Lower Rank and Dist. are better.

Terminal cost	Succ.	ρ_{geo}	Rank	Dist.
Raw latent MSE	7.0 ± 8.2	0.007	38.95	139.72
TRM, episodes excluded	96.7 ± 2.1	0.577	5.53	10.68
Shuffled, episodes excluded	0.0 ± 0.0	0.003	56.63	150.55

Table 23: Exclusion check for the audit in Table 22. Each run excludes the 200 evaluation start and goal rows from the high distance set and their 193 source episodes before sampling training and validation pairs. Row and episode overlap are measured after sampling.

Seed	Head	Rows	Row ov.	Ep. ov.	Val. RMSE
3072	temporal	164783	0	0	13.50
3072	shuffled	164783	0	0	26.47
3073	temporal	164895	0	0	13.43
3073	shuffled	164895	0	0	26.52
3074	temporal	164675	0	0	13.38
3074	shuffled	164675	0	0	26.24

C.11 TopoNav Diagnostic

TopoNav uses zero indexed grid coordinates (x, y) on a 29×29 map. The outer boundary is solid. Two vertical walls occupy $x = 9$ and $x = 19$ for interior rows, with single cell openings at $(9, 4)$ and $(19, 24)$. The map has 677 free states. The action set is stay, up, down, left, and right. A move into a wall leaves the state unchanged. Each cell is rendered as a 5×5 RGB block, producing a 145×145 observation. Free space has intensity $(0.92, 0.92, 0.92)$, walls use $(0.08, 0.08, 0.08)$, the goal uses $(0.10, 0.75, 0.22)$, and the agent uses $(0.10, 0.32, 0.95)$.

TopoNav uses a deterministic feature map with 10 dimensions as its frozen representation. For each state, the map concatenates normalized (x, y), two sine and cosine features with weight 0.25, two second frequency features with weight 0.12, and a room identity one hot vector with weight 0.04. Raw latent L2 reflects differences in features derived from position. Graph distance is not encoded. Logged trajectories follow graph shortest paths between uniformly sampled free cells. The training run uses 2,500 trajectories, 60,000 training pairs, and 12,000 validation pairs. Of the sampled pairs, 85% use trajectory endpoints and 15% use two random positions from the same trajectory. Pair order is randomized, and the label is the absolute step separation. The 120 evaluation pairs have graph distance of at least 18, a graph to Euclidean distance ratio of at least 1.45, and Manhattan separation of at least 12. Their exact unordered start and goal pairs are excluded from both training and validation.

The reported run uses base seed 608 and head seeds 3072, 3073, and 3074. Each head has two hidden layers of width 128 and trains for 24 epochs. We retain the checkpoint with the lowest validation loss. The controller uses horizon 12, 256 candidate action sequences, 5 CEM iterations, 32 retained candidate action sequences, categorical prior 0.2, and momentum 0.2.

In each CEM iteration, the controller overwrites 13 positions in the 256-sequence candidate pool: one candidate containing only stay actions and 12 route candidates. The first route candidate follows the shortest graph route and uses stay actions after reaching the goal. Each of the other 11 begins with that same length-12 route. Independently at every horizon position, a Bernoulli variable with probability 0.12 decides whether to draw a replacement. Conditional on replacement, the new action is uniform over all five actions,

$$P(a' = a \mid \text{replace}) = \tfrac{1}{5}, \qquad a \in \mathcal{A}, \tag{17}$$

where $\mathcal{A}$ is the set of five actions given above. The draw may equal the original action. Therefore the marginal probability of an actual action change at any position is $0.12 \times 4/5 = 0.096$. The number of replacement draws in one perturbed route is $\mathrm{Binomial}(12, 0.12)$ with expectation 1.44, while the number of positions whose action actually changes is $\mathrm{Binomial}(12, 0.096)$ with expectation 1.152. The remaining 243 candidate action sequences are sampled from the current CEM categorical distribution. Route injection is repeated in every CEM iteration after ordinary sampling and before scoring.

Every terminal cost scores all 256 candidate action sequences. The controller selects the candidate action sequence with the lowest cost. The environment executes its first action, after which the controller replans. The episode stops when the agent reaches the exact goal cell or after 90 executed actions.

The TopoNav SASC pool uses a separate, fully specified distribution and does not use the 0.12 route replacement rule. It first draws a 256×12 action array independently and uniformly from the five actions, then overwrites sequence 0 with stay actions and sequence 1 with the exact shortest route padded by stay actions. The remaining 254 sequences contain independent uniform actions at every position. For head seed s, the candidate seed is $608 + 31s + 3$. This seed is independent of the terminal cost, so raw latent MSE, temporal TRM, shuffled TRM, and the oracle score the same candidate action sequences within each evaluation unit. The constant offset preserves the pools used by the raw and temporal rows while extending exact candidate sharing to all controls.

C.12 Task Execution Visualizations

Figure 11 shows the route constraint in TwoRoom and the contact requirement in PushT. The drawings are schematic. All quantitative results come from the fixed evaluation sets and SASC diagnostics.

Table 24: TopoNav results averaged over head seeds 3072, 3073, and 3074. Success is the percentage of closed loop episodes that reach the exact goal cell. ρ_{geo} and Rank compare each terminal cost with exact graph distance on candidate pools shared by all four scores. Final geo. is the graph distance from the state reached after the last executed action to the goal.

Terminal cost	Succ.	ρ_{geo}	Rank	Final geo.
Raw latent MSE	15.0	0.496	45.03	36.18
Temporal TRM	87.5	0.848	1.29	3.01
Shuffled TRM	1.1	0.060	49.19	42.35
Oracle geodesic	95.6	1.000	0.00	0.08

Table 25: Closed loop success percentages on the balanced and distance controlled TWOROOM evaluation sets. B50 and B150 use the balanced set with action budgets of 50 and 150; D50 and D150 use the distance controlled set with the same budgets.

Checkpoint	Seed	B50	B150	D50	D150
LEWM np512	3072	55.0	60.0	2.5	17.5
LEWM np512	3073	57.5	60.0	2.5	7.5
LEWM np512	3074	52.5	57.5	0.0	5.0
PLDM full e10	3072	65.0	75.0	15.0	27.5
PLDM full e10	3073	67.5	77.5	17.5	27.5
PLDM full e10	3074	67.5	77.5	25.0	32.5

Table 26: High distance TWOROOM evaluation outcomes for each checkpoint seed and terminal cost. Success is measured over all 100 episodes. Same and Cross are success percentages within the 50 same room and 50 cross wall episodes. Wrong room and Stuck wall are percentages of all 100 episodes.

Seed	Terminal cost	Success	Same	Cross	Wrong room	Stuck wall
3072	LEWM latent MSE	16.0	26.0	6.0	44.0	14.0
3073	LEWM latent MSE	5.0	4.0	6.0	45.0	12.0
3074	LEWM latent MSE	0.0	0.0	0.0	47.0	5.0
3072	LEWM TRM	99.0	100.0	98.0	0.0	1.0
3073	LEWM TRM	99.0	100.0	98.0	0.0	0.0
3074	LEWM TRM	93.0	86.0	100.0	0.0	0.0
3072	LEWM shuffled TRM	0.0	0.0	0.0	39.0	15.0
3073	LEWM shuffled TRM	0.0	0.0	0.0	49.0	11.0
3074	LEWM shuffled TRM	0.0	0.0	0.0	50.0	0.0
3072	PLDM latent MSE	29.0	4.0	54.0	17.0	2.0
3072	PLDM TRM	74.0	58.0	90.0	1.0	2.0
3072	PLDM shuffled TRM	0.0	0.0	0.0	50.0	0.0
3073	PLDM latent MSE	34.0	0.0	68.0	12.0	15.0
3073	PLDM TRM	85.0	90.0	80.0	8.0	2.0
3073	PLDM shuffled TRM	0.0	0.0	0.0	50.0	0.0
3074	PLDM latent MSE	35.0	8.0	62.0	9.0	25.0
3074	PLDM TRM	93.0	90.0	96.0	0.0	2.0
3074	PLDM shuffled TRM	0.0	0.0	0.0	50.0	8.0

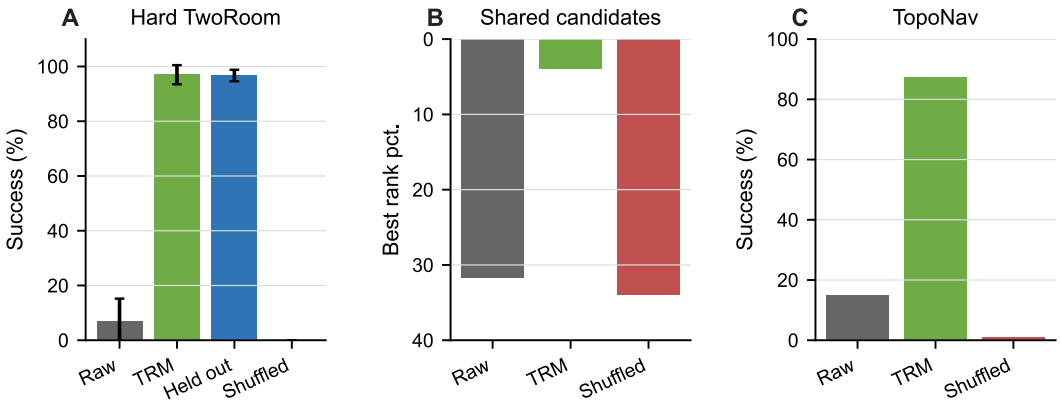

Figure 10: TopoNav context within the main evidence summary. Panel C reports 87.5% closed loop success for temporal TRM, compared with 15.0% for raw latent MSE and 1.1% for shuffled TRM.



Figure 11: Task execution schematics. In TWOROOM, the red straight route is shorter in Euclidean distance but is blocked by the wall, so success requires the blue route through the doorway. In PUSHT, the object is T shaped and must move through contact. SASC can therefore show better candidate ranking and selected distance even when contact, rollout error, and recovery limit closed loop success.

C.13 COVERAGE, BASELINE, AND LAYOUT CHECKS

The coverage experiment trains a geodesic pair head for one checkpoint seed. It is not the default temporal TRM experiment. All rows use checkpoint seed 3072, the same high distance set, the same controller budget, and the same SASC candidate pool. The limited doorway coverage setting samples 25,000 states that are at least 95 pixels from the nearest door. The balanced doorway coverage setting samples 12,500 states within 70 pixels of a door and 12,500 states farther away. Its 50,000 training pairs contain equal numbers of pairs with and without an endpoint near a door, and 50% of the pairs cross the wall. Mixed sampling uses 50% cross wall pairs without constraining distance to a door. Validation uses 10,000 pairs sampled independently from the training pairs.

The layout study uses checkpoint seed 3072 and 100 episodes for each layout. The controller budget remains fixed. Table 29 reports raw latent MSE, a geodesic pair head, and their hybrid for every tested layout. The geodesic pair head is trained on the original vertical wall layout and is not selected using outcomes from the changed layouts. Moving the doorway along the vertical wall gives partial transfer. Rotating the wall causes all three terminal costs to fail. This experiment does not test whether temporal TRM transfers to an unseen layout.

C.14 TEMPORAL HORIZON AND DATA COVERAGE

This ablation varies the number of observation rows available for pair sampling, the number of training pairs, and the maximum temporal separation of a pair. It keeps checkpoint seed 3072, the

Table 27: High distance evaluation terminal cost baselines. Rows are means over LEWM seeds 3072, 3073, and 3074. Within each seed, every terminal cost uses the same evaluation set, controller settings, CEM budget, and candidate pools. Succ. is closed loop success. ρ_{geo} is the SASC Spearman correlation with oracle geodesic cost. Rank is the percentile assigned to the candidate action sequence with the lowest oracle geodesic cost. Dist. is the final task distance after closed loop execution. Lower Rank and Dist. are better.

Terminal cost	Supervision	Succ.	ρ_{geo}	Rank	Dist.
Raw latent MSE	None	7.0	0.007	38.95	139.72
Standardized L2	Latent statistics	3.3	0.007	40.34	147.59
Linear abs diff	Geodesic	15.3	0.354	17.81	88.75
Diag. Mahalanobis	Geodesic	21.3	0.279	18.93	99.37
Decoded XY geodesic	XY probe	93.3	0.857	0.56	10.90
Temporal TRM	Logged temporal	97.0	0.678	3.79	7.85
Pairwise MLP TRM	Geodesic	99.3	0.908	0.45	7.61
Ranking loss TRM	Geodesic	95.7	0.839	0.70	9.22
Shuffled temporal	Shuffled temporal	0.0	0.053	44.10	138.86
Oracle geodesic	Oracle upper bound	100.0	1.000	0.00	5.74

Table 28: Coverage diagnostic trained with geodesic labels. Rank is the percentile assigned to the candidate action sequence with the lowest oracle geodesic cost. This table changes the coverage of the training states and pairs. It does not test whether temporal labels recover from the limited coverage split.

Coverage	Rows	Train pairs	Cross	Near door	Succ.	Rank
Limited doorway coverage	25k	50k	49.8%	0%	0	34.62
Balanced doorway coverage	25k	50k	50.0%	50%	100	0.49
Mixed	25k	50k	50.0%	Unconstrained	100	0.43
Random	30k	100k	50.0%	Unconstrained	100	0.45

Table 29: Results for every terminal cost evaluated on each changed layout. Each row uses 100 episodes and the same controller budget. Succ. is closed loop success. Rank is the percentile assigned to the candidate action sequence with the lowest oracle geodesic cost, and Dist. is the final distance to the goal. Lower Rank and Dist. are better.

Layout	Cost	Succ.	ρ_{geo}	Rank	Dist.
Door 88	Raw	7	0.130	40.46	128.04
Door 88	Geodesic	70	0.777	10.11	26.08
Door 88	Hybrid	20	0.571	13.81	79.60
Door 150	Raw	4	0.016	51.28	109.94
Door 150	Geodesic	44	0.461	27.11	41.11
Door 150	Hybrid	20	0.317	30.95	73.07
Horizontal 88	Raw	0	0.052	47.96	124.53
Horizontal 88	Geodesic	0	0.124	70.32	126.64
Horizontal 88	Hybrid	0	0.108	60.90	123.04

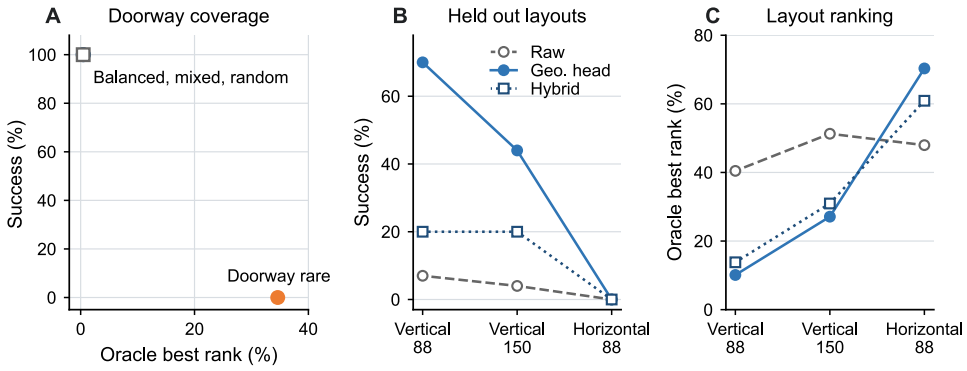

Figure 12: Coverage and layout diagnostics for checkpoint seed 3072. (A) The balanced doorway coverage, mixed, and random settings place the candidate action sequence with the lowest oracle cost near the top and reach 100% success. The limited doorway coverage setting reaches 0% success. (B) A geodesic pair head transfers partly when the doorway moves along the vertical wall. (C) No evaluated terminal cost ranks candidate action sequences reliably after the wall is rotated.

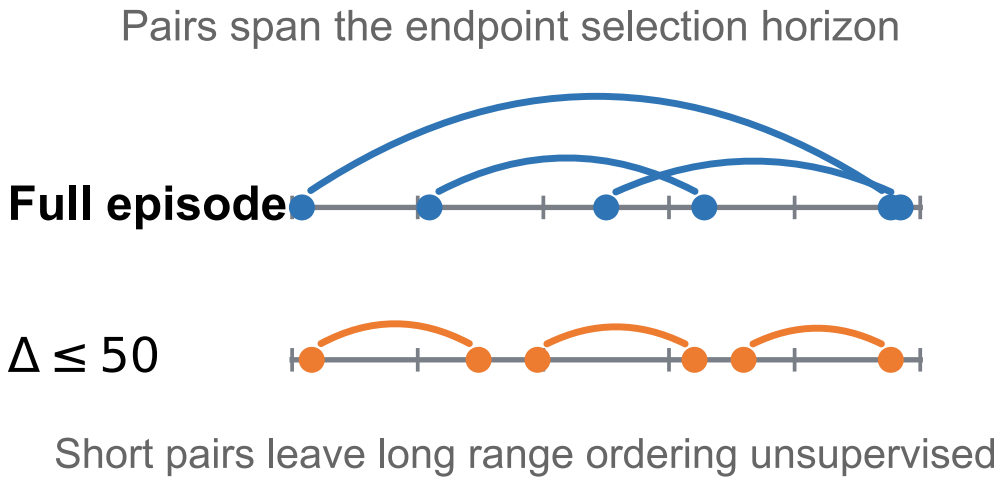

Figure 13: Temporal pair sampling. Full episode sampling includes pairs with temporal separations up to the episode length. The short horizon setting restricts every pair to $\Delta \leq 50$.

distance controlled set with a budget of 50 actions, the controller settings, and the 40 evaluation episodes fixed. With random temporal sampling, increasing the available rows and training pairs from 5,000 rows and 20,000 pairs to 100,000 rows and 100,000 pairs raises success from 17.5% to 90.0%. With 100,000 balanced pairs, full episode sampling reaches 97.5% success, while limiting every pair to $\Delta \leq 50$ reaches 35.0%. These results show that trajectory coverage, training pair count, and temporal separation each affect the learned terminal cost.

C.15 PushT Details

For PushT, the dataset key is `pusht_expert_train`. The checkpoint key is `pusht/lewm`, which resolves to `checkpoints/pusht/lewm_object.ckpt`. For each pair head, we sample 50,000 dataset rows without replacement. The frozen world model encodes the pixels and proprioception in each row. We then draw 200,000 training pairs and 10,000 validation pairs from this fixed set of rows with independent pair seeds. The head has two hidden layers of width 256 and trains

Table 30: TRM coverage and horizon ablations for LEWM seed 3072 on the distance controlled set with a budget of 50 actions. Every row uses 40 evaluation episodes and the same controller settings. With 100,000 balanced training pairs, full episode sampling reaches 97.5% success and the $\Delta \leq 50$ setting reaches 35.0%.

Sampling	Rows	Pairs	Horizon	Succ.
Random	5k	20k	Full episode	17.5
Random	10k	20k	Full episode	25.0
Random	20k	20k	Full episode	35.0
Random	20k	100k	Full episode	42.5
Random	100k	20k	Full episode	62.5
Random	100k	100k	Full episode	90.0
Balanced	–	20k	Full episode	80.0
Balanced	–	100k	Full episode	97.5
Balanced	–	20k	Max $\Delta = 50$	5.0
Balanced	–	100k	Max $\Delta = 50$	35.0

for 20 epochs with batch size 1024. We divide each target by 100 during training and retain the checkpoint with the lowest validation loss.

Closed loop evaluation uses two action dimensions. Each candidate action sequence contains five model steps and five environment actions. The action block also contains five actions, so the environment executes all five actions from the final CEM mean plan before the controller replans. In each iteration, CEM generates 300 candidate action sequences and retains the 30 candidate action sequences with the lowest terminal costs. CEM runs for 30 iterations with unit variance scale. After the final iteration, the controller uses the final CEM mean as an action sequence, and the environment executes its five actions. Goal offsets 25 and 50 use 50 episodes with an action budget of 50. Goal offset 75 uses 50 episodes per seed with an action budget of 75. The ranking audits compare the 64 or 96 candidate action sequences stated in their captions.

Let p_o be object position in pixels, p_e end effector position in pixels, and θ object angle in radians. The pair label is

$$d_{\text{task}}^2 = \|p_o - p_o'\|_2^2 + \|p_e - p_e'\|_2^2 + \left(\frac{180}{\pi} |\operatorname{wrap}(\theta - \theta')|\right)^2 . \tag{18}$$

Both position terms have unit weight. We convert the angle from radians to degrees before combining the three terms. We do not apply additional state normalization. We call the head trained on this distance the task state pair head. The shuffled pair head uses the same architecture and state pairs but permutes the training labels. The replacement terminal cost uses the corresponding pair head to assign a cost to each candidate action sequence. The task state hybrid standardizes the raw latent MSE cost and task state pair cost over the same candidate pool before adding them. The shuffled hybrid applies the same formula with the shuffled pair head. Threshold success records whether the environment reaches its success condition at any time. Final state success is computed from the state reached after the last executed action. It requires the joint four dimensional position error to be below 20 pixels and the wrapped angle error to be below $\pi/9$ radians.

Table 31: PUSHT offset 25 success over 50 episodes with evaluation seed 3072. This is a single seed comparison, so no significance test is reported.

Goal offset 25 terminal cost	Success
Raw latent MSE	88.0
Task state pair head	78.0
Task state hybrid	86.0
Shuffled pair head	64.0
Shuffled hybrid	84.0

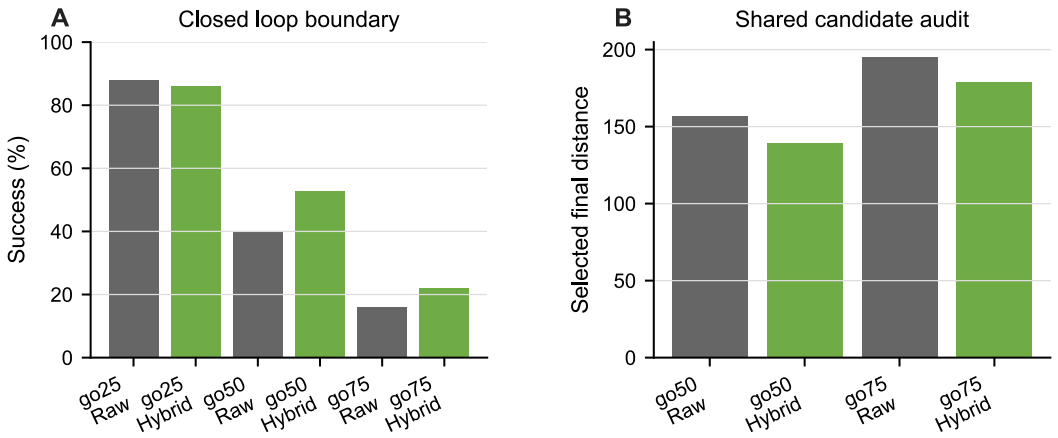

Figure 14: PUSHT diagnostics. Compared with raw latent MSE, the task state hybrid produces a larger reduction in the final task distance of the selected candidate action sequence than an increase in binary success.

For goal offsets 50 and 75, we also report SASC audits. Candidate order Spearman measures how each terminal cost ranks the same candidate action sequences. Selected final distance measures the outcome after the simulator executes the candidate action sequence with the lowest terminal cost.

Table 32: PUSHT offset 50 success for evaluation seeds 3072, 3073, and 3074. The rows report 50 episodes per seed. Tables 35 and 37 report candidate rankings and selected final distances for the same task.

Seed	Raw	Pair	Hybrid	Shuff. pair	Shuff. hybrid
3072	40.0	44.0	60.0	26.0	38.0
3073	38.0	40.0	48.0	28.0	40.0
3074	42.0	42.0	50.0	20.0	50.0

Table 33: PUSHT offset 75 closed loop results, averaged over 50 episodes for each of evaluation seeds 3072, 3073, and 3074. Mean dist. is final task distance, and Final succ. is final state success.

Cost	λ	Succ.	Mean dist.	Final succ.
Raw	–	16.0	219.7	4.7
Task hybrid	0.50	20.7	149.6	9.3
Task hybrid	1.00	22.0	103.7	12.7
Shuffled	0.50	16.7	215.9	6.7
Shuffled	1.00	17.3	215.8	6.7

Table 34: PUSHT offset 50 success across hybrid weights and pair head initializations, averaged over evaluation seeds 3072, 3073, and 3074. Raw latent MSE uses no pair head.

Head	Cost	λ	Succ.
None	Raw	–	40.0
Task state	Task hybrid	0.25	44.7
Task state	Task hybrid	0.50	46.7
Task state	Task hybrid	0.75	48.0
Task state	Task hybrid	1.00	52.7
Shuffled	Shuffled hybrid	0.25	42.7
Shuffled	Shuffled hybrid	0.50	38.7
Shuffled	Shuffled hybrid	0.75	42.7
Shuffled	Shuffled hybrid	1.00	42.7
Task state, 4072	Task hybrid	0.50	48.0
Task state, 4072	Task hybrid	1.00	50.0
Shuffled, 4072	Shuffled hybrid	0.50	42.7
Shuffled, 4072	Shuffled hybrid	1.00	43.3

Table 35: PUSHT offset 50 SASC results, averaged over 50 episodes for each of evaluation seeds 3072, 3073, and 3074 with 96 candidates per episode. Spearman compares terminal cost with realized final distance; lower oracle rank and selected distance are better.

Cost	Spearman	Oracle rank	Sel. dist.
Raw	0.490	20.41	156.6
Task pair	0.957	2.08	127.6
Task hybrid, $\lambda = .5$	0.703	12.96	144.8
Task hybrid, $\lambda = 1$	0.824	8.37	139.2
Shuffled pair	0.071	44.86	175.8
Shuffled hybrid, $\lambda = .5$	0.438	22.67	156.7
Shuffled hybrid, $\lambda = 1$	0.343	27.88	159.3

Table 36: PUSHT offset 75 SASC results, averaged over 50 episodes for each of evaluation seeds 3072, 3073, and 3074 with 96 candidates per episode. Spearman compares terminal cost with realized final distance; lower oracle rank and selected distance are better. Few candidates in these pools meet the success threshold.

Cost	Spearman	Oracle rank	Sel. dist.
Raw	0.490	25.73	195.1
Task pair	0.941	3.29	165.7
Task hybrid, $\lambda = .5$	0.696	16.64	184.0
Task hybrid, $\lambda = 1$	0.811	11.07	179.0
Shuffled pair	0.118	46.58	216.7
Shuffled hybrid, $\lambda = .5$	0.444	26.50	198.3
Shuffled hybrid, $\lambda = 1$	0.350	31.25	198.8

Table 37: PUSHT offset 50 CEM and SASC outcomes with 64 candidates, 10 iterations, and common random numbers, averaged over evaluation seeds 3072, 3073, and 3074. Refined is the executed final CEM mean, Pool best is an oracle diagnostic, and the selected columns report the candidate with minimum terminal cost.

Cost	Refined	Pool best	Sel. dist.	Sel. succ.
Raw	127.4	103.7	127.3	16.7
Task hybrid, $\lambda = 1$	75.3	59.1	75.3	16.0
Shuffled hybrid, $\lambda = 1$	137.8	113.2	137.8	14.0

Table 38: PUSHT offset 75 SASC outcomes with 64 candidates, 10 iterations, and common random numbers, averaged over evaluation seeds 3072, 3073, and 3074. Pool best is an oracle diagnostic; threshold and final state success follow the definitions above.

Cost	Sel. dist.	Final	Thresh.	Rank	Pool best
Raw	191.6	4.0	5.3	45.24	163.0
Task hybrid, $\lambda = 1$	114.5	4.0	4.0	34.63	96.7
Shuffled hybrid, $\lambda = 1$	183.4	4.0	4.0	43.04	158.3

Table 39: PUSHT offset 50 final distance diagnostic for evaluation seed 3072. Threshold success records a threshold crossing at any time. Mean and median final distance are computed from the state reached after the last executed action, and final state success tests only that state.

Cost	Thresh.	Mean	Median	Final
Raw	42.0	131.5	40.8	28.0
Task hybrid, $\lambda = 1$	60.0	48.0	26.2	42.0
Shuffled hybrid, $\lambda = 1$	38.0	154.9	50.9	30.0

D NEGATIVE RESULTS AND FAILURE CASES

The supplementary results include settings in which the learned terminal cost does not repair control. These cases define the scope of the evidence. They also show that a terminal cost can rank candidate action sequences well even when the selected candidate action sequence does not succeed during execution. A compact boundary result map with the exact comparisons is included in the code and data supplement.

In the representative trace from the high distance TWOROOM evaluation, raw latent MSE favors action sequences whose terminal predictions remain near the wall instead of routing through the doorway. The resulting CEM plan is executed for the configured 25-action receding-horizon prefix before replanning. Across the 100 episodes in the high distance set, the controller using raw latent MSE ends in the wrong room in 45.3% of episodes and at the wall in 10.3%. In the layout transfer test, moving the doorway gives partial transfer, while rotating the wall causes every evaluated terminal cost to fail. In PUSHT, using the task state pair head as the terminal cost improves the ranking of candidate action sequences and reduces the final distance produced by the selected candidate action sequences. The environment still has to execute the selected action prefix, and success depends on contact, rollout accuracy, and recovery.

E FALSIFICATION CRITERIA

Several observations would weaken the terminal cost interpretation. If raw latent MSE ranked candidate action sequences with low oracle costs near the top of the same candidate pool, the ranking diagnosis would fail. The diagnosis would also fail if decoded XY or rowspace costs did not improve control despite accurate position probes, if shuffled heads matched temporal TRM, or if a larger raw CEM budget closed the success gap. The coverage explanation would weaken if balanced doorway coverage reduced validation error without improving either oracle best rank or closed loop success. The current TWOROOM results show the opposite pattern. Raw latent MSE assigns poor ranks, rowspace and decoded costs improve control, shuffled labels fail, a larger raw CEM budget does not improve success, and doorway coverage repairs the failure caused by rare doorway observations. The horizontal wall result limits the claim to related layouts, and PUSHT shows that better candidate ranking can coexist with limited contact execution and recovery.

F ADDITIONAL CHECKPOINT COMPARISON

We compare seven archived LEWM checkpoints on a separate public TwoRooms dataset with 9,000 training and 1,000 evaluation episodes. Each frozen checkpoint has its own freshly trained temporal

head. The comparison uses a common 50-episode screening set from the evaluation split, containing 34 same-room and 16 cross-wall cases, with goal offset 25 and an action budget of 50. The checkpoint identifiers in Table 40 retain their recorded run and epoch indices.

CEM uses 300 candidates, 30 optimization iterations, top-k 30, and solver seed 42. Planning horizon, receding horizon, and action block are each five. Each head is trained for 30 epochs with hidden dimension 256, 100,000 training pairs, and 10,000 validation pairs from the training split. Action and proprioceptive normalization are fitted on the training split.

Table 40: Seven-checkpoint screening comparison. Success is counted over the same 50 episodes for each checkpoint and selector. The gain is TRM minus raw success in percentage points, with the recorded 95% paired episode-bootstrap interval. Distance is the mean final Euclidean distance in pixels.

Run	Epoch	Raw success	TRM success	Gain [95% interval]	Raw distance	TRM distance
0	99	43/50	48/50	$+10$ $[-2, 22]$	16.68	7.91
1	55	48/50	49/50	$+2$ $[-4, 8]$	7.09	7.81
2	3	45/50	30/50	-30 $[-46, -14]$	10.50	29.13
3	19	41/50	45/50	$+8$ $[-6, 22]$	17.40	14.21
4	67	46/50	46/50	0 $[-10, 10]$	11.26	11.04
4	99	44/50	47/50	$+6$ $[-6, 18]$	11.95	8.80
5	99	46/50	46/50	0 $[-10, 10]$	9.06	9.60

Success increases for four checkpoints, is unchanged for two, and decreases for the epoch-3 checkpoint. The intervals for the four increases and two ties span zero. The epoch-3 change is -30 percentage points, with interval $[-46, -14]$. The result shows that the effect of terminal-cost replacement varies across checkpoints and evaluation settings.

G Candidate Availability and Ranking

Protocol. The replay study uses the three main LeWM policy seeds, 3072, 3073, and 3074, with the matched high distance 100-episode evaluation set and evaluation/solver seed 44. For each policy seed, we record both the raw-cost and TRM controllers. CEM uses 300 candidates, 10 optimization iterations, top-k 30, planning and receding horizons of five, and action block five. Each episode has two recorded replanning queries. Every final-iteration candidate is executed for its complete 25-frame sequence in the TwoRoom simulator, from the corresponding query state. In total, the comparison contains 1,200 query states and 360,000 candidate outcomes.

Both scores are evaluated on the same candidates and simulator outcomes within each pool. Let d_0 be the diagnostic geodesic distance at the query state and d_i the distance after executing candidate i. For a selected candidate i_s, available progress and ranking regret are

$$P = d_0 - \min_i d_i, \qquad R_s = d_{i_s} - \min_i d_i. \tag{19}$$

The diagnostic categories use $\delta = 16$ pixels: generation limited means $P \leq \delta$; ranking limited means $P > \delta$ and $R_s > \delta$; near-best selection means $P > \delta$ and $R_s \leq \delta$. These are query-level progress and regret categories. Closed loop success uses the simulator's goal-hit criterion. The ranking comparison uses top-one selectors; the native controller executes its final CEM mean.

On the raw-generated pools, TRM reduces mean finite regret by 9.65 pixels. On the TRM-generated pools, raw scoring has 1.37 pixels lower regret. The two controllers also visit different query states.

Remaining native-controller failures. The recorded native controllers succeed on 21/300 episodes with raw cost and 291/300 with TRM. The nine failed TRM episodes contain 18 replanning states: zero are generation limited, five are ranking limited, and thirteen have near-best top-one selection.

Oracle selection within the pool. A separate closed loop run keeps each native candidate generator and selects the candidate with the smallest simulator-evaluated terminal geodesic distance. It

Table 41: Candidate availability and ranking under shared simulator outcomes. Each pool source contributes 600 replanning states. The three category columns are percentages of all states for that source. Mean regret is in pixels and uses 599 finite outcomes for raw scoring on raw pools and 600 for each other row.

Pool source	Selector	Generation limited	Ranking limited	Near-best selection	Mean regret
Raw	Raw	26.67	61.33	12.00	22.67
Raw	TRM	26.67	28.00	45.33	13.02
TRM	Raw	0.33	26.83	72.83	13.32
TRM	TRM	0.33	31.83	67.83	14.69

executes that candidate's complete 25-frame sequence and replans. Table 42 reports the results. Native success records reaching the goal at any time during execution; the oracle's selection objective uses the endpoint after the full sequence.

Table 42: Closed loop outcomes with simulator oracle selection within each generated pool. Each entry uses 100 episodes per policy seed. The oracle selects one complete candidate sequence.

Pool generator	Seed 3072	Seed 3073	Seed 3074	Total	Success (%)
Raw	28	24	10	62/300	20.7
TRM	99	100	99	298/300	99.3

Candidate-budget comparison. We also compare 50, 100, 300, and 1,000 candidates at the same 100 initial states for each policy seed and generator. This gives 2,400 state–arm observations and 870,000 candidate outcomes. The 300-candidate outcomes reuse the first-query replay above. The table reports available progress and finite-conditioned mean regret; the outcomes retain unreachable terminal distances. Increasing the pool budget raises available progress for both generators. These measurements concern one replanning query at each initial state.

Table 43: Initial-state candidate-budget comparison, pooled over three policy seeds. Progress and regret are in pixels. The last column gives the percentage of candidates with unreachable terminal geodesic distance.

Candidates	Generator	Available progress	Raw regret	TRM regret	Unreachable (%)
50	Raw	16.80	7.88	5.83	0.240
50	TRM	92.09	10.08	9.78	0.640
100	Raw	26.82	15.36	10.23	0.137
100	TRM	111.60	15.50	16.00	0.987
300	Raw	38.17	24.01	14.25	0.221
300	TRM	127.83	20.30	21.88	0.400
1000	Raw	45.63	30.19	16.67	0.235
1000	TRM	132.25	23.00	23.82	0.289