WHEN DOES TEMPORAL DISTANCE IMPROVE WORLD-MODEL PLANNING?

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models plan by keeping candidate action sequences whose predicted final state lies closest to the goal in latent space. This latent distance, however, is not trained to reflect reachability: states that are close in latent space may still require many steps to connect. Learned temporal distances can address this mismatch, but it remains unclear when replacing the latent distance will improve planning, or whether the benefit can be predicted before planning. We propose a simple offline diagnostic using trajectories already collected to train the world model. A useful planning distance should rank states requiring fewer steps to reach the goal as closer; we therefore measure how much a learned temporal distance improves this ordering over the original latent distance. With the world model and search fixed and the execution schedule matched, planning improves substantially when the temporal distance greatly improves temporal ordering, while gains are smaller when the original latent distance already captures that ordering. We further apply the diagnostic before planning in held-out settings, where its predictions match the observed distinction between large and smaller planning gains. Controlled experiments show that the large gains come from learning temporal structure rather than simply changing the representation, search budget, or execution schedule. The results suggest that temporal ordering in offline trajectories provides a practical way to identify when latent distance is a bottleneck in world-model planning.

1 INTRODUCTION

Latent world-model planners predict the outcomes of candidate action sequences and select actions according to how close the predicted states are to the goal in a learned representation (Hafner et al., 2019; Ding et al., 2025). Hierarchical world models additionally introduce coarser representations that propose intermediate targets, while a lower level plans the actions needed to reach them (Hafner et al., 2022; Zhang et al., 2026; Caselli et al., 2026; Cheng et al., 2026). In both flat and hierarchical planners, the distance in the learned representation can determine which predicted state appears to make the most progress toward the goal.

The distance used by the planner, however, is not necessarily trained to reflect *reachability*: how many steps are required to move from one state to another. Consider two-room navigation, where a wall with a single door divides the arena (Figure 1). A state just across the wall from the goal can appear close in latent space even though reaching the goal requires a long detour through the door. The latent distance can also assign nearly the same distance to states that require very different numbers of steps to reach the goal (Figure 3). A *temporal distance*, obtained by learning a representation through future prediction and measuring distance in that representation, can better reflect how many steps separate two states and therefore provide a better measure of reachability (Bae et al., 2024; Li et al., 2026; Singh, 2026; Sommer & Schilling, 2026). Whether replacing the original latent distance with a temporal distance improves planning more generally is less clear. On Push-T, a learned temporal distance improved the ranking of candidate plans more than planning success (Li et al., 2026). In separate work, retraining the world model around a temporal cost substantially improved state ranking but worsened planning (Bai & Xiong, 2026). Better temporal ordering is therefore not sufficient by itself, and whether a temporal distance will improve planning is typically known only after running the planner.

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.

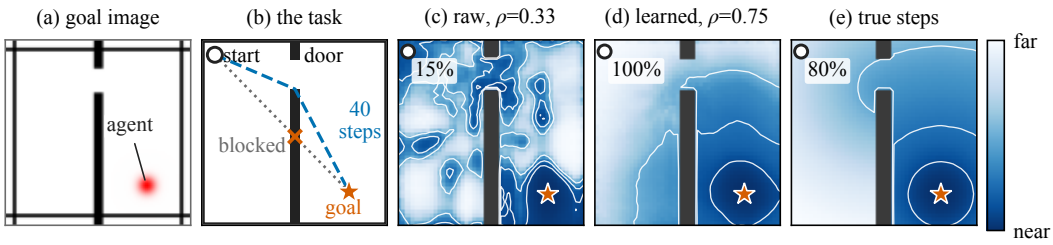


Figure 1: **Latent distance can misrepresent reachability.** (a) The goal image in two-room navigation. (b) The wall separates the example start (circle) from the goal (star), so reaching the goal requires a long detour through the doorway despite the small geometric separation. (c–e) Each position is coloured by the rank of its distance to the goal, from closest (dark) to farthest (light), under the original latent distance, the learned temporal distance, and the true number of steps to the goal. White curves show equal-rank contours, and ρ is the rank correlation with the true number of steps. The original latent distance ranks the example start among the closest 15% of positions, whereas the temporal distance assigns the start a large distance, similar to the true steps-to-go.

We ask whether the benefit of a temporal distance can be predicted before planning: *can trajectories collected to train a world model reveal whether the planner's current distance poorly represents reachability?* We use a simple offline diagnostic. A useful planning distance should assign smaller distances to states that require fewer steps to reach the goal. We therefore compare how well the original latent distance and a learned temporal distance preserve the ordering of states by temporal separation. When the temporal distance substantially improves this ordering, planning also improves substantially; when the original latent distance already captures the ordering well, the benefit is smaller. Diagnostics based instead on geometric state distance can give a different conclusion because geometric distance does not account for obstacles or other constraints that determine reachability. Section 3 develops this diagnostic and examines why temporal supervision produces the observed gains.

This question emerged from an unexpected result in hierarchical planning. We built an upper-level representation that maps the world-model latent into a compact abstract state trained to predict twenty environment steps ahead. Planning through this representation by selecting abstract subgoals helped little. In development runs, however, simply scoring the existing candidate plans by distance in the upper-level representation raised two-room success at offset 60 from below 50% to nearly 100%. The upper-level representation was therefore more useful as a measure of progress toward the goal than as a mechanism for planning in coarse steps. We return to this implication for hierarchical world models in the Discussion.

We study this question using LeWorldModel (LeWM) (Maes et al., 2026), which provides a common backbone for flat and hierarchical planners and allows the planning distance to be changed while keeping the world model and search procedure fixed. We first diagnose when the original latent distance poorly reflects temporal ordering and isolate the role of temporal supervision (Section 3). We then evaluate the diagnostic prospectively on fresh tasks and in held-out settings while keeping the world model and search fixed and matching the execution schedule (Section 4). This work focuses on diagnosing when the planning cost of a frozen world model is a bottleneck rather than proposing a new temporal-distance objective; temporal distances have a long history in goal-conditioned control (Hartikainen et al., 2020; Eysenbach et al., 2022; Qian et al., 2023; Wang et al., 2023; Park et al., 2024).

Our main contributions are:

- We introduce a simple offline diagnostic for when replacing a world model's latent distance with a learned temporal distance is likely to improve planning: measure how much the temporal distance improves the ordering of states by temporal separation.
- We evaluate the diagnostic prospectively while fixing the world model and search and matching the execution schedule. The diagnostic distinguishes settings with large planning

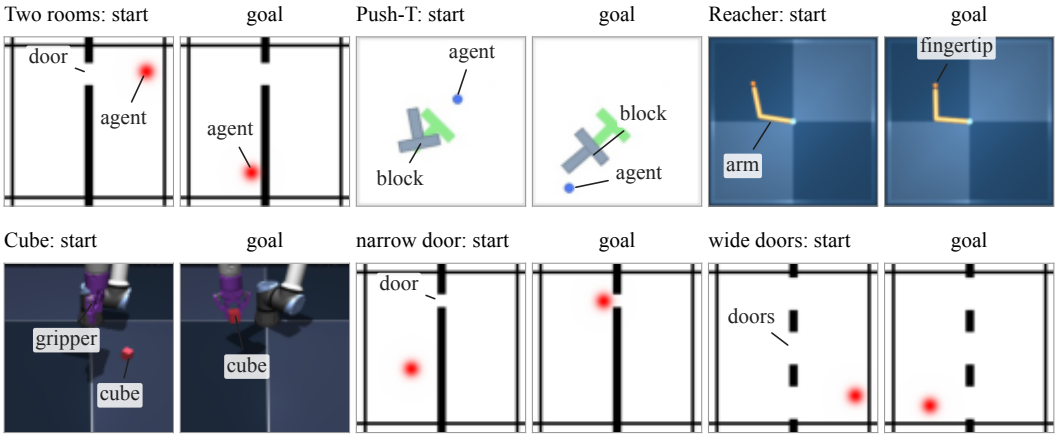


Figure 2: **Evaluation environments.** Each example shows the start and goal image of one task. In two-room navigation, the agent (red dot) must reach the position shown in the goal image. In Push-T, the agent (blue dot) must push the grey block to the target pose; the green marker is part of the environment visualization. In Reacher, the arm must match the configuration shown in the goal image. The planner observes only images.

gains from those with smaller gains before running the temporal-cost planner, including in held-out settings.

- We trace the large gains to temporal structure through controls on the training target, representation rescaling, search budget, and execution schedule. The results further show that an upper-level representation can help through the planning distance induced by the representation, even when explicitly planning through the upper level helps little.

2 Problem Setup

Environments and tasks. We study two-room navigation, Push-T, and Reacher. In *two-room navigation* (Sobal et al., 2025), a point agent moves in a square arena divided by a wall with a single door. In *Push-T*, a circular agent pushes a T-shaped block to a target pose. In *Reacher*, a two-link arm moves its fingertip to a target. A *task* consists of a start state and a goal state from the same recorded episode, separated by a fixed number of environment steps, which we call the *goal offset*. The planner receives the goal as an image and has twice the goal offset in steps. Success is determined by the environment's goal test.

We use offsets 60 and 95 for two-room navigation, 25, 50, and 75 for Push-T following Caselli et al. (2026), and 25 and 50 for Reacher (Figure 2). The two-room offsets are several times the planner's 25-step horizon; at offset 25, latent distance already solves most tasks (Singh, 2026).

World model and planner. We use the public LeWorldModel (LeWM) checkpoint for each environment (Maes et al., 2026). An image is encoded into a 192-dimensional latent state z, and the world model predicts future latent states under blocks of five environment actions. Given the current image and a goal image with latent state z_g, the planner searches over H action blocks $u = (u_1, \ldots, u_H)$ using the cross-entropy method (Rubinstein & Kroese, 2004). LeWM scores each candidate by the *original latent distance*

$$C_{\text{latent}}(u) = \|\hat{z}_H(u) - z_g\|_2^2, \tag{1}$$

where $\hat{z}_H(u)$ is the predicted final latent state. The planner runs 30 iterations of 300 candidates, keeps the best 30, executes the first R blocks and replans; R is the *execution schedule*.

Temporal distance. We train an encoder $\phi : \mathbb{R}^{192} \to \mathbb{R}^{64}$ using temporal prediction, inducing a distance that better reflects temporal separation. Training uses latent states cached from 1,500–2,000

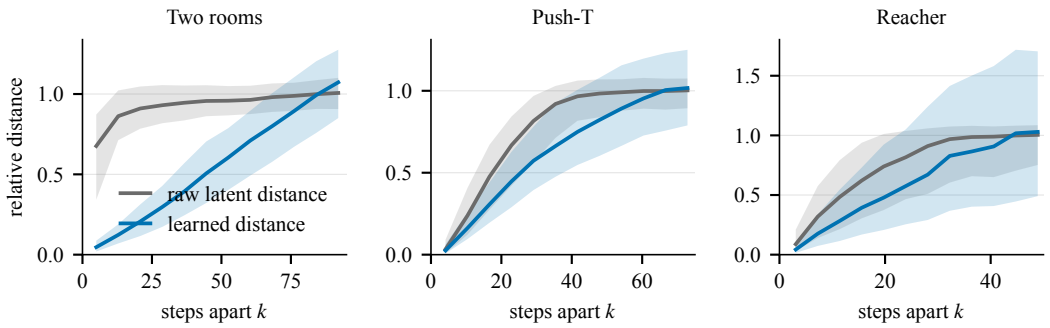


Figure 3: **Distance as a function of temporal separation.** Each curve shows distance against the number of steps separating two states from the same trajectory (median and interquartile range; each distance is normalized by its median at the largest temporal gaps). In two-room navigation, the original latent distance saturates quickly, whereas the temporal distance continues to increase. In Push-T, the difference is smaller but becomes more visible at longer temporal separations.

recorded episodes. A predictor f predicts the encoded latent twenty environment steps ahead, while a decoder g reconstructs the original latent to prevent representation collapse:

$$\mathcal{L} = \tfrac{1}{64}\left\| f(\phi(z_t), \bar{a}_{t:t+20}) - \bar{\phi}(z_{t+20}) \right\|_2^2 + \tfrac{1}{192}\left\| g(\phi(z_t)) - z_t \right\|_2^2, \tag{2}$$

where $\bar{a}_{t:t+20}$ summarizes the next four action blocks and $\bar{\phi}$ is an exponential-moving-average copy of ϕ. We train for 20,000 updates.

At planning time, only ϕ is used. A candidate is scored by

$$C_\lambda(u) = \|\phi(\hat{z}_H(u)) - \phi(z_g)\|_2^2 + \lambda \|\hat{z}_H(u) - z_g\|_2^2. \tag{3}$$

With $\lambda = 0$, the planner uses only the *temporal distance*; with $\lambda = 1$, the temporal and original latent distances are combined.

Planner variants. We compare four planners that share the same world model and search procedure. The published *LeWM planner* uses the original latent distance with $H = R = 5$. The *original latent cost, matched execution* planner uses the same distance with $H = R = 4$, matching the temporal-cost planners and separating the effect of the planning cost from the execution schedule. The *temporal cost* planner uses C_0, and the *temporal + latent cost* planner uses C_1, both with $H = R = 4$. Appendix G gives reproduction checks across software versions and task draws.

3 Diagnosing When Latent Distance Fails

3.1 Offline temporal-ordering diagnostic

A planning cost should order candidate endpoints correctly: a state requiring fewer steps to reach the goal should receive a smaller cost. We approximate this property offline using trajectories already collected to train the world model: state pairs separated by fewer trajectory steps should receive smaller distances.

For each environment, we sample 20,000 state pairs (z_t, z_{t+k}) from the same recorded episode, with temporal gap k sampled uniformly between one step and the largest goal offset. Along a recorded trajectory, k approximates the number of steps separating the two states; this approximation is particularly direct in navigation and less exact for demonstration trajectories such as Push-T. For each distance, we compute the rank correlation between k and squared distance and measure how often a temporally distant pair appears closer than a nearby pair.

The contrast is largest in two-room navigation. The rank correlation increases from 0.31 with the original latent distance to 0.81 with the temporal distance, while the fraction of temporally distant pairs that appear closer than nearby pairs falls from 21% to 1%. In Push-T and Reacher, the temporal

Table 1: **Offline temporal-ordering diagnostic.** Rank correlation between temporal gap k and squared distance, measured on recorded trajectories before planning. "Far pair looks closer" is the fraction of pairs with $k \geq K/2$ whose distance is smaller than that of pairs with $k \leq K/8$, where K is the largest evaluated goal offset.

		rank correlation		far pair looks closer	
environment	K	original	temporal	original	temporal
Two-room navigation	95	0.308	0.809	21%	1%
Push-T	75	0.676	0.734	1.5%	1.0%
Reacher	50	0.511	0.542	7.3%	6.5%

distance improves the rank correlation by only 0.06 and 0.03, respectively. Thus, the temporal distance substantially changes the ordering of states in two-room navigation but only modestly changes the ordering in the other two environments.

Importantly, the quality of the original latent distance alone does not predict the planning outcome. The original distance has lower rank correlation in Reacher than in Push-T, yet the temporal distance provides a larger planning benefit in Push-T. The relevant quantity is instead the *improvement in temporal ordering* obtained by replacing the original latent distance.

We therefore use the improvement in temporal-ordering quality as an offline diagnostic. A large improvement indicates that the original latent distance is likely to be a bottleneck for planning, while a small improvement suggests a smaller benefit or none. The diagnostic is intended as a screening tool rather than as a quantitative predictor of the exact change in planning success. In particular, better temporal ordering does not guarantee better planning if other components of the world model or planner remain limiting.

3.2 Why temporal separation rather than geometric distance?

Diagnostics based on geometric state distance can lead to a different conclusion because geometric proximity does not necessarily reflect reachability (Savinov et al., 2018; Eysenbach et al., 2019; Tian et al., 2020). In two-room navigation, positions on opposite sides of the wall can be geometrically close while requiring a long path through the doorway. Existing measurements based on state-space geometry rate LeWM's latent representation as relatively well aligned in two-room navigation compared with Push-T and Reacher (Wang et al., 2026; Hu et al., 2026), even though two-room navigation is the environment in which replacing the planning distance provides the largest benefit.

The difference is what the two diagnostics measure. Geometric alignment asks whether latent distance resembles a distance defined directly over states. Temporal ordering instead asks whether states requiring fewer steps to reach the goal are ranked as closer, which is the property used when candidate plans are compared. The wall in two-room navigation makes this distinction explicit: geometric distance ignores the detour through the doorway, whereas temporal separation reflects it.

A large improvement in temporal ordering is not sufficient to guarantee better planning—other failure modes can remain (Bai & Xiong, 2026)—but the diagnostic isolates whether the planning distance itself is a plausible bottleneck while keeping the underlying world model fixed.

3.3 Effect of temporal supervision

We next examine why the temporal distance improves planning when the effect is large. These controls use development tasks in two-room navigation, where the difference between the original and temporal distances is sufficiently large to distinguish alternative explanations. The studies guided the confirmatory evaluation in Section 4; full task counts, seeds, and results are reported in Appendix D.

Training target. We trained encoders with the same architecture and optimization schedule but changed the learning target. An encoder trained only to reconstruct the current latent state improves success by only 6–8 points at offset 60 and performs slightly worse than the original latent distance at offset 95. Predicting randomly permuted future states behaves similarly. In contrast, predicting

the actual future latent raises success by about 50 points at both offsets, consistently across three training seeds. The gain therefore comes from temporal supervision rather than simply adding another learned representation.

Representation rescaling. PCA and whitening change the geometry of the original latent representation without introducing temporal information. Both recover part of the planning gap, particularly at offset 60, but remain substantially below the temporally trained representation. Poor scaling therefore contributes to the failure of the original latent distance but does not explain the full effect.

Alternative temporal objectives. The specific temporal-learning objective is not essential. Temporal-difference heads and frame-separation heads (Singh, 2026) reach similar success when trained with comparable data and optimization budgets. Their differences are small relative to the shared improvement over the original latent distance, suggesting that temporal supervision is the important common factor.

3.4 Search budget and execution schedule

Increasing the number of candidates per cross-entropy iteration from 75 to 600 does not repair the original latent-distance planner in two-room navigation, whereas a temporal-cost planner remains strong even at the smallest budget tested (Appendix D). The failure therefore does not arise simply from insufficient search.

The execution schedule is an important confound. In early Push-T and Reacher experiments, replanning after every block made the temporal-cost planner appear substantially worse, but the same schedule also degraded the original latent-distance planner. Replanning after every block can reduce success by 21–56 points, whereas executing four blocks before replanning changes the original latent-distance planner by at most a few points. We therefore match the execution schedule in all comparisons that isolate the effect of the planning cost.

4 Prospective Evaluation of the Diagnostic

4.1 Pre-registered evaluation on fresh tasks

Because development experiments influenced the planner configuration, we evaluated one configuration once on fresh tasks with the hypotheses and analysis fixed in advance. The tested configuration was the temporal + latent cost ($\lambda = 1$). The temporal cost alone ($\lambda = 0$) was run on the same tasks and is reported alongside, but was not part of the pre-registered test.

Design. All four planners ran on every task. Evaluation seeds not used during development produced 400 tasks in two-room navigation, 900 in Push-T, and 1,280 in Reacher. Three tasks that duplicated development tasks were removed before any outcome was inspected, leaving 2,577 tasks.

We tested paired differences in success between the temporal + latent planner and the published LeWM planner: superiority in two-room navigation and non-inferiority with a five-point margin in Push-T and Reacher. Standard errors were clustered by dataset episode, and the tests were corrected with Holm's procedure (Holm, 1979) at a family-wise level of 0.025. The reported lower bounds are simultaneous confidence bounds at family-wise level 0.025; hypothesis decisions use Holm's procedure as described above. Sample sizes were determined from development variance, and the protocol, task lists, analysis code, checkpoints, datasets, and software inputs were fixed by hash before evaluation (Appendix B).

Results. All three pre-registered null hypotheses were rejected (Table 2). In two-room navigation, the temporal + latent cost raises success from 47% to 97% at offset 60 and from 17% to 80% at offset 95. In Push-T, performance is similar at offset 25 and improves by 12 and 15 points at offsets 50 and 75. In Reacher, the planners remain within about one point at both offsets.

These results follow the pattern suggested by the offline diagnostic. Two-room navigation has by far the largest improvement in temporal ordering and also the largest planning gain. Push-T shows a smaller improvement in temporal ordering and a more moderate gain, concentrated at longer goal

Table 2: **Pre-registered evaluation on fresh tasks.** $\hat{d}$ is the paired difference in success, in percentage points, between the temporal + latent planner and the LeWM planner, pooled over goal offsets. The final column reports the temporal cost on the same tasks; this configuration was not part of the pre-registered hypothesis test.

			temporal + latent (tested)			temporal only
environment	hypothesis	tasks	$\hat{d}$	lower bound	H_0	$\hat{d}$ [95% CI]
Two-room navigation	superiority	397	+56.4	+50.4	rejected	+60.5 [55.3, 65.6]
Push-T	non-inferiority (−5)	900	+9.4	+6.1	rejected	+9.3 [6.4, 12.3]
Reacher	non-inferiority (−5)	1,280	+1.2	−2.5	rejected	+0.3 [−2.8, 3.5]

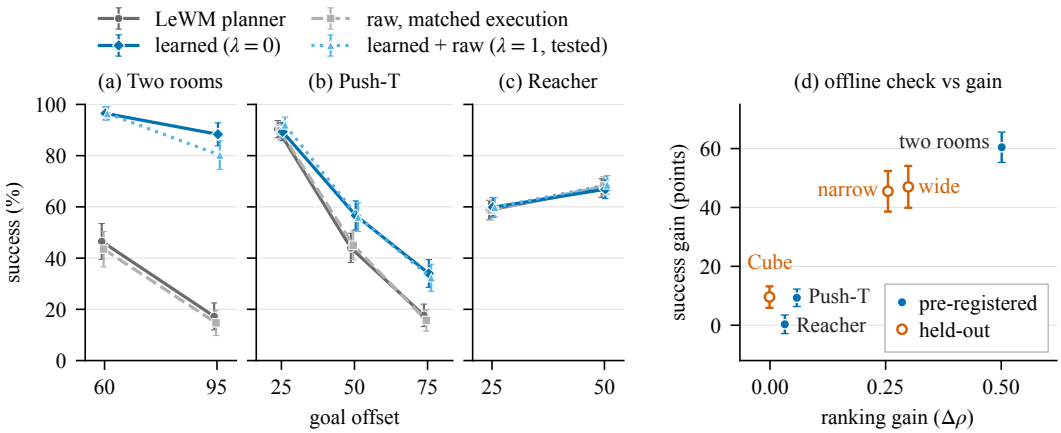


Figure 4: **Offline temporal ordering and planning improvement.** (a–c) Success as a function of goal offset for the published LeWM planner, the original latent cost with matched execution, the temporal cost, and the temporal + latent cost. (d) Improvement in temporal-ordering quality—the rank correlation of the temporal distance minus that of the original latent distance—against the corresponding gain in planning success. Filled markers denote the pre-registered evaluation; hollow markers denote settings in which the offline diagnostic was recorded before planning with the temporal cost. Error bars show 95% intervals, clustered by dataset episode where available.

offsets. Reacher shows little change in either temporal ordering or planning success. The diagnostic therefore separates the large gain from the smaller effects, although the limited difference between Push-T and Reacher does not support treating the diagnostic as a precise quantitative predictor.

The original latent cost with matched execution performs similarly to the published LeWM planner throughout the evaluation. The planning gains therefore arise from changing the planning distance rather than from changing the execution schedule.

Combining the temporal and original latent distances. The temporal cost performs as well as the temporal + latent cost in Push-T and Reacher and performs better at the farthest two-room offset (88% versus 80%). The original latent term, which we expected to improve precision near the goal, provides no benefit. We therefore use the temporal cost alone in the prospective checks below.

4.2 Evaluation in held-out settings

The pre-registered evaluation establishes a relationship between improvement in temporal ordering and improvement in planning, but the diagnostic would be useful if the relationship could be applied before planning in a new setting. We therefore considered settings that had not been used to establish the diagnostic and recorded the diagnostic outcome before running the temporal-cost planner.

We use OGBench Cube, the remaining LeWM manipulation environment available to us, and two new door layouts of two-room navigation. One navigation layout uses a narrower doorway, while the

Table 3: **Prospective application of the offline diagnostic.** The temporal-ordering measurements and predictions were recorded before planning with the temporal cost. Planning gain is the pooled difference in success between the temporal-cost planner and the LeWM planner.

setting	original ρ	temporal ρ	improvement	planning gain
Cube	0.405	0.404	-0.001	$+9.6$
Narrow-door navigation	0.201	0.456	$+0.255$	$+45.5$
Wide-door navigation	0.181	0.480	$+0.299$	$+47.0$

other uses multiple wide doorways. These changes alter the reachability structure without changing the basic navigation task.

Before running the temporal-cost planner, we fixed a coarse categorical rule based on the three preceding environments: an improvement in rank correlation of at least 0.25 predicts a planning gain of at least 15 percentage points, whereas an improvement of at most 0.15 predicts a gain below 10 points. Intermediate values make no prediction. The rule is deliberately categorical because the diagnostic is intended to screen for large failures rather than predict an exact success rate.

The observed gains match the pre-specified categories. Cube shows essentially no improvement in temporal ordering and a comparatively small planning gain. The narrow- and wide-door layouts show improvements of 0.26 and 0.30 in rank correlation and correspondingly large gains of 46 and 47 points. The Cube confidence interval extends slightly above the ten-point boundary, so the diagnostic should not be interpreted as a precise threshold or quantitative law.

The door-layout results also show that the original two-room observation is not tied to one particular doorway geometry. Even with substantially different connectivity, the offline diagnostic identifies when the original latent distance strongly misrepresents progress toward the goal. Together with Cube, these experiments support using the diagnostic as a practical screen before committing to planning with a learned temporal distance.

4.3 Planning overhead and qualitative analysis

Planning overhead. The temporal cost adds one small encoder evaluation per candidate and requires no additional world-model predictions. With $H = R = 4$, the temporal-cost planner makes the same number of world-model predictions per executed step as the published LeWM planner with $H = R = 5$. The temporal representation contains 0.39 million parameters at planning time and is trained from cached LeWM latents.

Qualitative analysis. Figure 5 illustrates the failure mode on fresh two-room tasks. Along the recorded paths, the original latent distance changes little until the agent approaches the goal, while the temporal distance decreases more consistently. Across the corresponding evaluation seed and offset, the median rank correlation with steps remaining is 0.43 for the original latent distance and 0.96 for the temporal distance, and planning success rises from 12% to 96%.

4.4 Comparison with hierarchical planning

Hi-LeWM (Caselli et al., 2026) adds a learned upper level to the same frozen LeWM backbone and reports results at the same Push-T offsets. Its strongest reported variants improve over its flat planner by approximately 11 and 15 points at offsets 50 and 75. When we replace only the planning cost inside Hi-LeWM's evaluation code, the untuned temporal-cost planner improves over the corresponding flat planner by 9 and 10 points on the same tasks. Thus, changing the distance used by the flat planner recovers a planning improvement of similar magnitude without explicitly searching over upper-level subgoals.

The temporal-cost planner is also substantially cheaper. At these offsets, the lower level of Hi-LeWM uses approximately six to eight times as many world-model predictions per executed step, in addition to its upper-level search. Its second level adds 12.5 million parameters and requires 7.6 GPU-hours of training, whereas our temporal representation adds 0.39 million parameters at

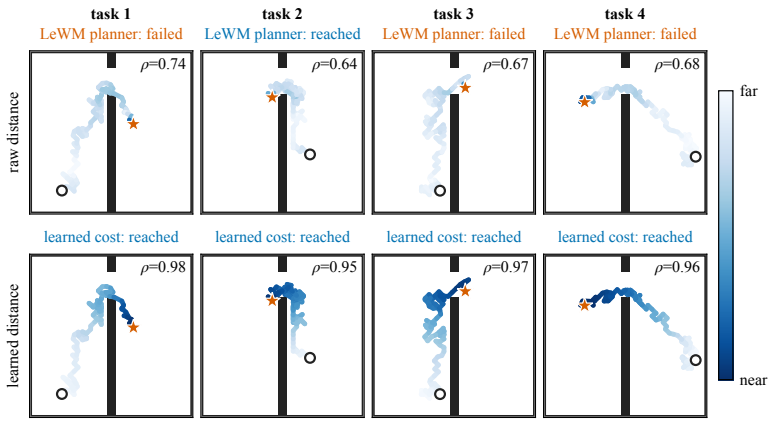


Figure 5: **Temporal distance along example trajectories.** Four fresh two-room tasks from the pre-registered evaluation are shown along their recorded start-to-goal trajectories. Each trajectory is coloured by distance to the goal under the original latent distance (top) and temporal distance (bottom), from closer (dark) to farther (light). The original latent distance changes little along much of the trajectory, whereas the temporal distance decreases more consistently toward the goal. Titles report whether the corresponding planner reached the goal.

planning time and trains in about two minutes from cached latent states. Appendix G gives the complete comparison.

These results do not imply that subgoal planning is unnecessary. Prior work shows that intermediate targets can provide additional gains (Caselli et al., 2026; Liu et al., 2026). Rather, substantial gains can arise from the induced planning distance even without explicit upper-level search.

5 Discussion

Implications for hierarchical world models. Our temporal representation originated as the upper level of a hierarchical world model. Explicitly planning through abstract subgoals helped little in two-room navigation, whereas using distance in the same representation to score lower-level candidate plans produced a large improvement. This suggests two complementary roles for an upper-level representation: proposing subgoals and providing a useful measure of progress toward the goal. Existing hierarchical planners primarily emphasize the former (Hafner et al., 2022; Zhang et al., 2026; Caselli et al., 2026; Cheng et al., 2026); our results show that the latter can also be important.

Limitations. The diagnostic is a screening tool rather than a quantitative predictor: a small ordering gain can still accompany a modest planning gain, as in Cube. Our temporal cost is symmetric, whereas reachability can be directional, and we tested one world model, LeWM, with one encoder per environment (Appendix A).

6 Conclusion

Latent world-model planners rely not only on accurate predictions but also on a distance that measures useful progress toward the goal. We showed that the improvement in temporal ordering provided by a learned temporal distance can serve as an offline diagnostic for when the original latent distance is a planning bottleneck. When temporal ordering improves substantially, replacing the planning cost can yield large gains; when the original latent distance already captures the ordering, the benefit is smaller. Controlled experiments trace the large gains to temporal supervision, and prospective checks in held-out settings support applying the diagnostic before planning. The results also suggest that hierarchical representations should be evaluated not only by the subgoals they produce, but also by the planning distances they induce.

REPRODUCIBILITY STATEMENT

The supplementary material contains the code of the learned distance, the planners and the evaluation; the pre-registered protocol, the task lists and the catalog of SHA-256 hashes of every source file, checkpoint and dataset (with identifying strings replaced by neutral tokens); the raw per-task outcomes of all 168 runs of the test and the frozen analysis that produced Table 2; the scripts and inputs that draw every figure; and, for the comparison with Hi-LeWM, the wrapper that runs our cost inside its evaluation code, the sweep configurations, the raw outputs, the parameter counts and training timing, and the script that writes Appendix G. The protocol is described in Appendix B and the development studies in Appendix D. Public LeWM checkpoints and datasets were used throughout.

AI USE STATEMENT

In this work, we used generative AI tools (large language model agents, including Claude and ChatGPT) extensively and under our direction for the following tasks with required disclosure: proposing and refining hypotheses, designing experiments and giving feedback on their design, implementing the methods, the planners and the analysis, and interpreting results. We did not use them to generate synthetic datasets, to clean or reformat datasets, for translation, or for qualitative data analysis, and the tasks concerning mathematical claims and proofs do not apply to this work. We also used them for tasks with recommended disclosure: suggesting experimental parameters, writing and editing code, running and monitoring experiments, creating the figures, searching and summarising the literature, formatting references, suggesting the structure and title, and drafting and editing the text. Two separate agent sessions reviewed each other's experimental designs, statistical analysis and code before the pre-registered test, and the test was run from inputs fixed by hash before any outcome was seen. The numbers in the paper were checked against the stored raw outcomes by an automated audit, whose discrepancies were corrected, and the references against bibliographic records. No language model is a component of the methods studied. We have reviewed all AI-assisted work, and we take responsibility for the final content of this paper, including text, claims and code produced with the aid of generative AI.

REFERENCES

Junik Bae, Kwanyoung Park, and Youngwoon Lee. TLDR: Unsupervised goal-conditioned RL via temporal distance-aware representations. *arXiv preprint arXiv:2407.08464*, 2024.

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

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

Letian Cheng, Qi Zhang, and Yisen Wang. SAGE: Subgoal-conditioned action generation for latent world model planning, 2026.

Jingtao Ding, Yunke Zhang, Yu Shang, Yuheng Zhang, Zefang Zong, Jie Feng, Yuan Yuan, Hongyuan Su, Nian Li, Nicholas Sukiennik, et al. Understanding world or predicting future? a comprehensive survey of world models. *ACM Computing Surveys*, 58(3):1–38, 2025.

Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. In *Advances in Neural Information Processing Systems*, 2022.

Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*, 2020.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pp. 65–70, 1979.

Jiaming Hu, Yan Zheng, Tian Wang, Florian Dubost, Alejandro Mottini, Junze Liu, Arvind Srinivasan, Kai Zhong, Kun Qian, Sharon Gao, and Qingjun Cui. SCALE: State-calibrated latent embeddings for JEPA planning in the right geometry, 2026.

Liangyu Li, Shengzhi Wang, Libin Qiu, Mingliang Xiong, and Qingwen Liu. World model control by trajectory reachability metrics, 2026. First version 21 May 2026.

Xvyuan Liu, Jianjie Fang, Chen Gao, and Yong Li. Aim short to reach far: Your frozen world model can plan better than you think, 2026.

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

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with Hilbert representations. In *International Conference on Machine Learning*, 2024.

Zhifeng Qian, Mingyu You, Hongjun Zhou, Xuanhui Xu, and Bin He. Goal-conditioned reinforcement learning with disentanglement-based reachability planning. *IEEE Robotics and Automation Letters*, 8(8):4721–4728, 2023.

Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer, 2004.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018.

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

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models, 2025.

Armin Sommer and Jannik Schilling. Reinforced planning with latent world models, 2026.

Stephen Tian, Suraj Nair, Frederik Ebert, Sudeep Dasari, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Model-based visual planning with self-supervised functional distances. *arXiv preprint arXiv:2012.15373*, 2020.

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

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning*, 2023.

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

A LIMITATIONS

The offline diagnostic is a screening tool rather than a quantitative predictor: a small improvement in temporal ordering can still accompany a modest planning gain, as in Cube. Temporal separation along recorded trajectories is also only an approximation to reachability, especially for demonstrations that need not follow efficient paths. Our temporal cost is also symmetric, whereas reachability can be directional in manipulation tasks; incorporating asymmetric reachability measures is an interesting direction for future work. We train one temporal encoder per environment and evaluate primarily with LeWM, so the diagnostic remains to be tested on world models with different latent geometries. Finally, our study focuses on changing the planning cost of a frozen world model rather than methods that retrain the world model itself.

B THE PRE-REGISTERED PROTOCOL

Before the run we fixed the tested configuration, the four planners, the 168 runs (seven environment–offset pairs; four evaluation seeds in two-room navigation, six in Push-T and eight in Reacher; four planners), the 2,580 drawn tasks and the three excluded ones, the three hypotheses, the analysis and two deadlines. A second, separate agent session (see the AI use statement) checked the analysis code on synthetic data before the run, and the protocol and its hash were recorded, dated, before any outcome existed; that record will be released after review. Rules fixed in advance included the following. The decision uses the larger of the complete-case and worst-case p-values, where the worst case counts a missing outcome of the tested planner as a failure and a missing outcome of the LeWM planner as a success. A test with zero variance or fewer than 30 clusters cannot reject. Per-offset results are reported next to pooled ones. The runner refused to start unless every source file, checkpoint and dataset matched a catalog of SHA-256 hashes, and it checked each task list against the registered list before running it. All 168 runs completed on the first attempt, so the complete-case and worst-case analyses coincide.

C ENCODER HORIZON AND A POSITION-BASED COST IN TWO-ROOM NAVIGATION

We varied the prediction horizon of the encoder: its rank correlation falls from 0.81 at twenty steps to 0.64 at eighty steps. Both the distortion of the latent space and the wall may matter. On the public two-room checkpoint, Singh (2026) found that a cost decoding the agent's position, which undoes the distortion but ignores the wall, raised success at a long offset from 14 to 70%, whereas our learned distance, which better reflects temporal separation, reaches 97 and 88% at offsets 60 and 95; the protocols differ, so this comparison is indicative only.

D DEVELOPMENT STUDIES

The controls in Table 4 used 20 tasks per goal offset and evaluation seed, with two evaluation seeds per study (40 tasks per offset); the seeds differ between studies. The training-target controls and the other learned costs average three independently trained encoders or heads. The search-budget curve (Figure 6a) uses evaluation seeds 410 to 413; the execution comparison (Figure 6b) uses seeds 416–417 and 420–421, with the raw-cost and LeWM planners always compared on the same tasks. The development phase ran eleven iterations; all, including failed ones, are in the released log. Two other choices were settled during development. Scoring every intermediate block instead of only the final one made no difference. Appending coarse steps of the learned predictor f after the fine blocks, as the original two-level model did, added up to 10 points (6 on average) at the farthest two-room offset and lowered success on Push-T, so we plan with ϕ alone.

E PER-OFFSET RESULTS OF THE PRE-REGISTERED TEST

Table 5 gives the success rates plotted in Figure 4a–c. A post-hoc analysis, not part of the protocol: some fresh tasks come from dataset episodes that were among those the learned distance was trained

Table 4: Two-room development controls (success %, goal offsets 60 / 95; 40 tasks per cell, so differences below about 20 points are within noise). Rows within a block share tasks and differ only in the cost. In the training-target block, and for the temporal-difference and frame-separation heads, each entry averages three independently trained models. "Released budget" is the training budget of the released implementation of Singh (2026) (60 episodes); "full data" uses the same cached episodes and number of updates as our encoder.

study	cost used by the planner	success
Training target	raw latent distance	42.5 / 32.5
	reconstruct only	50.8 / 29.2
	predict shuffled futures	48.3 / 22.5
	predict the latent 20 steps ahead (ours)	**93.3 / 82.5**
Rescaling raw space	raw latent distance	32.5 / 20.0
	PCA of the raw latents	52.5 / 45.0
	whitened raw latents	62.5 / 27.5
	learned distance	**90.0 / 85.0**
Other temporal costs	raw latent distance	50.0 / 17.5
	learned distance	92.5 / 82.5
	directed temporal-difference head	89.2 / 80.0
	symmetric temporal-difference head	90.0 / 83.3
	frame-separation head (Singh, 2026), released budget	73.3 / 52.5
	same, full data	92.5 / 87.5
	same, full data, real and imagined pairs	94.2 / 91.7

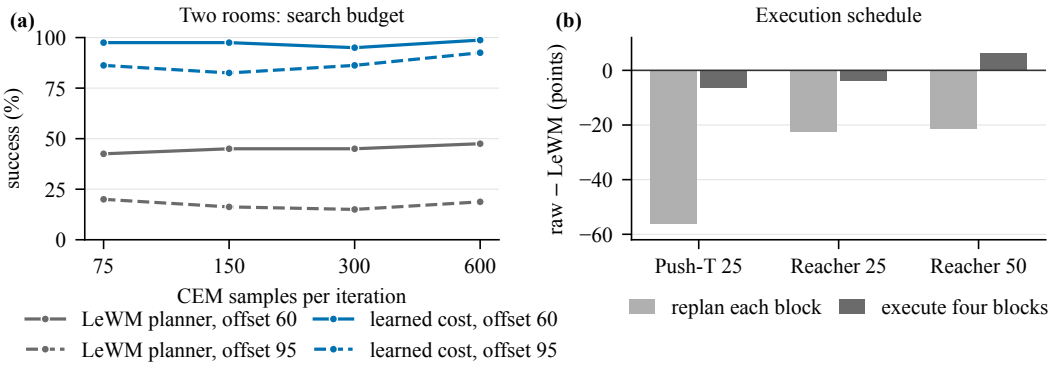


Figure 6: Development controls. (a) Two-room success against the number of samples per iteration; the learned-cost curve is a development variant whose plans append six coarse steps of the predictor f. (b) Raw-cost planner minus the LeWM planner on the same tasks, when replanning after every block (R=1) or executing four blocks (R=4).

on: 67 of 397 in two-room navigation, 108 of 900 in Push-T and 285 of 1,280 in Reacher. On the remaining tasks, the LeWM planner and the learned cost reach 32.4 and 93.0% in two-room navigation, 50.4 and 59.2% in Push-T and 63.5 and 62.1% in Reacher; on the overlapping tasks they reach 29.9 and 89.6%, 52.8 and 65.7%, and 61.4 and 67.7%. The two-room and Push-T gains do not depend on the overlap; in Reacher the small pooled gain comes from the overlapping tasks. Two-room navigation has 197 tasks at offset 95 because the three tasks that repeated development tasks were excluded before any outcome was seen, and the intervals in the figure are clustered by dataset episode, as in the test.

F THREE HELD-OUT SETTINGS

Settings. After the pre-registered test we took three settings we had not used and applied the check before any planning with the learned cost. *Cube* is OGBench's single-cube manipulation task, the fourth environment of LeWM. Its only LeWM checkpoint available to us is the one Hi-LeWM ships

Table 5: Success (%) on the fresh tasks by goal offset. All planners ran on the same tasks.

environment	offset	tasks	LeWM planner	raw, matched	learned	learned + raw
Two rooms	60	200	46.5	43.5	**96.5**	**96.5**
	95	197	17.3	14.7	**88.3**	80.2
Push-T	25	300	90.3	89.7	89.3	**92.0**
	50	300	44.0	45.0	**56.7**	56.0
	75	300	17.7	15.7	**34.0**	32.3
Reacher	25	640	58.8	58.8	**59.8**	**59.8**
	50	640	67.3	68.1	66.9	**68.6**

(for Push-T, Hi-LeWM's checkpoint is identical to the public one). We cached latents of 1,000 of its 10,000 training episodes with that checkpoint and evaluated in Hi-LeWM's code on its seeds 42 to 44 with 50 tasks each at offsets 25, 50 and 75 and a budget of twice the offset. The *door layouts* change only the wall of two-room navigation: one door 17 pixels wide instead of the default 29, or three doors 43 pixels wide, as measured in rendered frames. For each we collected 6,000 episodes with the expert policy and noise settings that match the released data's action statistics. We trained LeWM with its released training code and two-room configuration for 47 epochs and evaluated with the evaluator of our test on two fresh seeds with 50 tasks each at offsets 60 and 95. The learned distance was trained with the settings of Section 2 on latents of 1,500 episodes.

Prediction. Before any check was computed, we wrote down the rule of Section 4.2. A ranking gain of at least 0.25 predicts a pooled gain of the learned cost over LeWM's planner of at least 15 points, one of at most 0.15 a gain below 10, and anything between no prediction. Each check was then computed and its prediction recorded before any planning with the learned cost in that setting; the record is in the supplementary material. The checks (20,000 pairs, largest offset 75 for Cube and 95 for the door layouts) gave rank correlations of 0.405 and 0.404, 0.201 and 0.456, and 0.181 and 0.480 for the raw and the learned distance (Figure 7).

Evaluation of the door layouts. Stable-worldmodel's evaluation from a dataset resets the environments without the variation that sets the door, so we wrapped the reset to pass it. We also measured the door openings in the last rendered frame of every run, and all door-layout runs planned with the recorded openings. A first launch with an older copy of our evaluator, which replans after every block in the learned-cost modes, was stopped within two minutes and its outputs set aside unread.

Results. Table 6 gives the success rates. Pooled over offsets, the learned cost (λ=0) gained $+9.6$ ($+5.9$ to $+13.2$), $+45.5$ ($+38.6$ to $+52.4$) and $+47.0$ ($+39.9$ to $+54.1$) points over LeWM's planner on Cube, with the narrow door and with the wide doors. The intervals for the door layouts are clustered by dataset episode. Against the raw cost at matched execution, the learned cost gained $+8.9$, $+49.5$ and $+52.5$ points, and the raw cost at matched execution differed from LeWM's planner by $+0.7$, -4.0 and -5.5 points, so the gains come from the cost and not from the execution schedule. All three predictions held. On Cube, per offset, the learned cost gained $+6.0$, $+10.0$ and $+12.7$ points at offsets 25, 50 and 75 in Hi-LeWM's code and on its seeds. The best of Hi-LeWM's Cube variants reports $+5.4$, $+2.7$ and $+16.0$ over its flat planner (Caselli et al., 2026, Tables 3 and 4), so the comparison of Section 4 extends to a second environment.

G Comparison with Hi-LeWM on Push-T

Hi-LeWM (Caselli et al., 2026) trains a coarser second level on top of a frozen LeWM and uses it to plan subgoals for LeWM's planner. Its constrained variant samples the upper level's candidate plans around macro-actions encoded from training trajectories, and its staged variant plans the subgoals once per episode instead of replanning them. Of our three environments it reports only Push-T, at our goal offsets and step budgets. Each of its results except the staged ones is the best of a sweep over planning configurations, by mean success over three seeds of 50 tasks. Table 7 compares it with our learned cost in three ways.

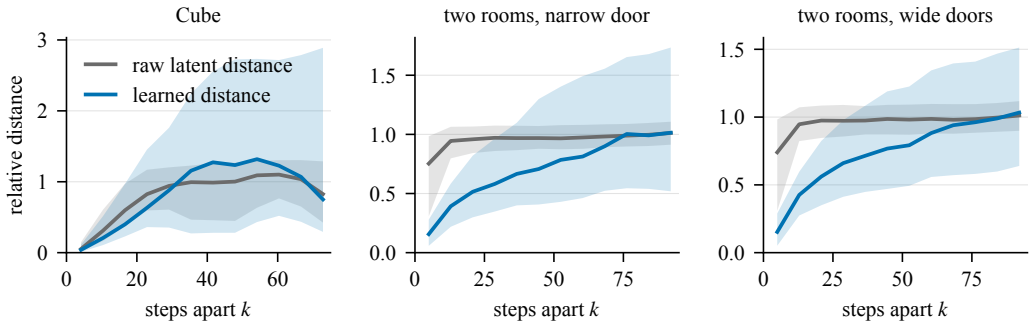


Figure 7: Distance against the number of steps between two states of the same recorded episode in the three held-out settings (start and goal images in Figure 2), computed as in Figure 3. With either door layout the raw distance saturates within about fifteen steps while the learned distance keeps increasing; on Cube the two distances order states alike, matching its ranking gain of about zero (Table 1).

Table 6: Success (%) in the three held-out settings, by goal offset. Cube: Hi-LeWM's evaluation code, seeds 42 to 44, 150 tasks per offset (the LeWM planner's seed 42 comes from our rerun of Hi-LeWM's sweep). Door layouts: the evaluator of our test with the door applied at reset, two fresh seeds, 100 tasks per offset.

		goal offset		
setting	planner	25	50	75
Cube	LeWM planner	74.7	53.3	64.0
	raw, matched execution	70.7	57.3	66.0
	learned (λ=0)	80.7	63.3	76.7
		60	95	
narrow door	LeWM planner	57.0	48.0	
	raw, matched execution	53.0	44.0	
	learned (λ=0)	98.0	98.0	
	learned + raw (λ=1)	86.0	93.0	
wide doors	LeWM planner	60.0	46.0	
	raw, matched execution	52.0	43.0	
	learned (λ=0)	100.0	100.0	
	learned + raw (λ=1)	87.0	95.0	

Code, reproduction and task sets. Hi-LeWM evaluates with an older release of the same library (stable-worldmodel 0.0.6, against 0.1.1 here) and a LeWM checkpoint whose weights are identical to the public one. Reading both releases along the path from the dataset to the success test, we found one behavioural difference: the older release places the goal one environment step earlier. Start states for a given seed, image processing, model rollouts, the cross-entropy method and the success test are the same. Running its released code over its sweeps of the flat planner and the unconstrained variant on seed 42, we obtained 96, 58 and 18% for the flat planner (published: 94.0, 52.7 and 18.0%) and 84, 44 and 22% for the best unconstrained configuration (published: 89.3, 38.7 and 15.3%), each within two standard errors of 50 tasks. We did not rerun the constrained variant. Success also varies between task sets drawn with different seeds: at offset 50 the flat planner reaches 36 to 56% across the six task sets of our test, and 40 to 58% across seeds 42 to 44 of Hi-LeWM's code. Absolute success rates from different task sets can therefore differ by as much as the gains of interest, and we compare gains over the flat planner on the same tasks.

Same code and tasks. We ran our learned cost inside Hi-LeWM's evaluation code, replacing only the planner's cost, untuned, in the configuration the learned cost had in our test (λ=0, H=R=4, 300 samples; the tested planner added the raw term), and ran its flat planner on the same tasks: seeds 42 to 44, starting from its default seed, with 50 tasks each. The learned cost reached 88.7, 56.0

and 27.3% against 90.7, 46.7 and 17.3% at offsets 25, 50 and 75. The paired differences are -2.0 (-7.7 to $+3.7$), $+9.3$ ($+1.4$ to $+17.3$) and $+10.0$ ($+2.0$ to $+18.0$) points, with 95% intervals over 150 tasks, against -1.0, $+12.7$ and $+16.3$ in our pre-registered test. The best of Hi-LeWM's four published variants differs from its flat planner, itself the best of a sweep, by -3.3, $+11.3$ and $+14.7$ points; in absolute terms it reaches 90.7, 64.0 and 32.7%, on task draws of its own.

Same tuning. We also gave our cost Hi-LeWM's tuning protocol, in our code on fresh seeds 440 to 442 with 50 tasks each. At each offset we took the configuration with the best mean over the three seeds among those varying the execution schedule (R=2, 3, 4 with H=4), the raw term (λ=0, 1) and, at offsets 50 and 75, the number of samples (900 and 1,200 as in Hi-LeWM's lower level, keeping the best tenth): six, eight and eight configurations at offsets 25, 50 and 75. The best reached 92.0, 56.0 and 41.3% against the flat planner's 82.0, 38.7 and 18.7% on the same tasks, differences of $+10.0$, $+17.3$ and $+22.6$ points; the selected configurations were λ=0, R=4, 300 samples; λ=0, R=4, 300 samples; λ=0, R=4, 1,200 samples. At offsets 25 and 50 the selection is the configuration the learned cost had in our test. As in Hi-LeWM's protocol, the flat planner keeps LeWM's configuration: Hi-LeWM's flat sweep fixed 300 samples and selected H=5, R=5 at every offset. Like Hi-LeWM's, these numbers are selected on the tasks they are reported on and are therefore optimistic.

Cost. Per executed block, our planner evaluates 300 candidates over 30 iterations, each predicted four blocks ahead, and plans again after four blocks: 9,000 world-model predictions, as many as LeWM's own planner. Hi-LeWM's best configurations at offsets 50 and 75 plan their lower level two blocks ahead with 900 and 1,200 samples over 30 iterations and plan again after every block (its Tables 4 and 5): 54,000 and 72,000 predictions per executed block, six and eight times ours, in addition to its upper level's search with 1,500 samples over 40 iterations and 1,200 over 60. Its second level adds 12.5 million parameters to LeWM's 18.0 million, and its Push-T model trained for 7.56 hours on one A100 GPU. Our encoder has 0.39 million parameters at planning time and 1.1 million together with the predictor and decoder used only in training; it trained in 137 seconds on one RTX 6000 Ada GPU, on latents cached in one pass of LeWM's encoder.

Table 7: Push-T success (%) by goal offset. Hi-LeWM paper: published means over three seeds of 50 tasks; every entry except the staged ones is the best configuration of a sweep. Hi-LeWM code: its evaluation code on seeds 42 to 44 (150 tasks per offset), with our cost untuned. Our code, best of sweep: fresh seeds 440 to 442 (150 tasks per offset), best configuration per offset. Our code, pre-registered: Table 5, 300 fresh tasks per offset.

		goal offset		
setting	planner	25	50	75
Hi-LeWM paper	LeWM planner	94.0	52.7	18.0
	Hi-LeWM, unconstrained	89.3	38.7	15.3
	Hi-LeWM, unconstrained, staged	90.7	42.0	15.3
	Hi-LeWM, constrained	88.7	48.7	32.7
	Hi-LeWM, constrained, staged	70.7	64.0	22.0
Hi-LeWM code	LeWM planner	90.7	46.7	17.3
	learned (λ=0), untuned	88.7	56.0	27.3
our code, best of sweep	LeWM planner	82.0	38.7	18.7
	learned, best configuration	92.0	56.0	41.3
our code, pre-registered	LeWM planner	90.3	44.0	17.7
	learned (λ=0)	89.3	56.7	34.0
	learned + raw (λ=1, tested)	92.0	56.0	32.3