TREKWM: LONG-HORIZON PLANNING VIA SHAPED LATENT GEOMETRY AND ADAPTIVE SUBGOALS

Anonymous authors
Paper under double-blind review

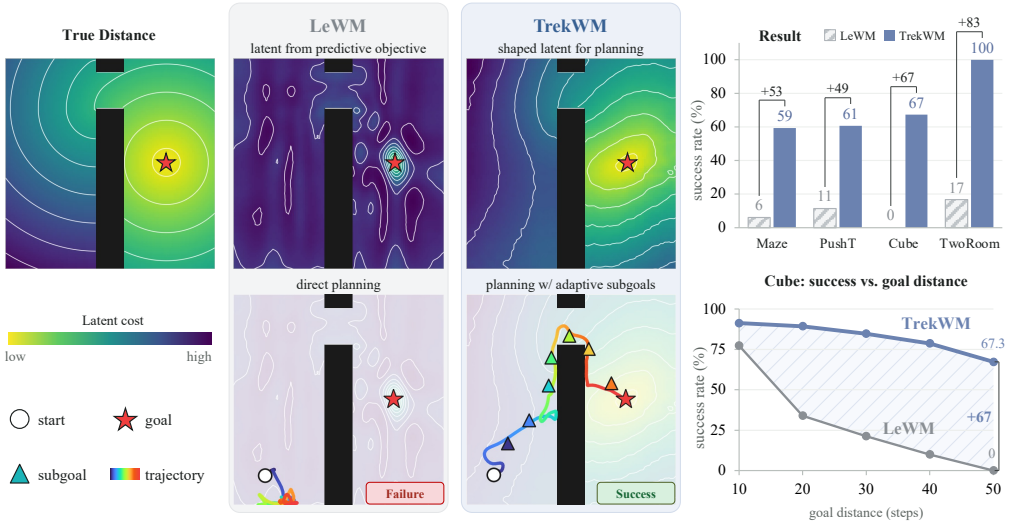

Figure 1: **Shaped latent geometry and adaptive subgoals make distant goals reachable.** *Left:* In TwoRoom, LeWM latent cost is informative only near the goal ($\star$), so direct planning gets stuck. The cost of TREKWM (ours) follows the true distance, and adaptive subgoals ($\blacktriangle$) lead the planner through the doorway to reach the goal. *Right:* TREKWM improves success over LeWM by 49–83 points on long-horizon settings in all four environments (top). On Cube, the gap widens as the goal moves farther away, reaching 67.3% vs. 0.0% at 50 steps (bottom).

ABSTRACT

Latent world models (LWMs) offer a promising framework for zero-shot visual control by planning in a latent representation space rather than raw pixels. However, latents trained exclusively for prediction do not naturally induce a metric suitable for evaluating planned trajectories: Euclidean distance between latents fails to reliably reflect true environment steps, leading to severe degradation over long task horizons. To address this, we propose TREKWM to improve long-horizon planning in LWMs via two complementary mechanisms. First, a ranking loss shapes the latent geometry so that latent Euclidean distance preserves temporal ordering when scoring candidate futures during planning. Second, we shorten the effective planning horizon using an adaptive subgoal proposer trained with a flexible temporal schedule, enabling the discovery of meaningful task landmarks rather than relying on rigid, fixed intervals. Together, these complementary improvements ensure both reliable local distance estimation and scalable global planning. Across eleven settings in five environments, TREKWM achieves the top performance in ten, outperforming competing methods by a substantial margin. The advantage is most pronounced on the longest-horizon tasks: 67.3% vs. 0.7% on Cube, 60.7% vs. 23.3% on PushT, and 100.0% vs. 54.7% on TwoRoom, in each case compared against the strongest baseline for that setting.

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.

1 INTRODUCTION

A long-standing goal of learning-based control is to solve new tasks at test time without task-specific rewards, demonstrations, or further training. Planning with a learned world model offers a compelling solution: dynamics are learned offline from unrewarded interaction, enabling zero-shot planning toward a goal specified at inference. For visual control, a popular approach uses Joint-Embedding Predictive Architectures (JEPAs) (Assran et al., 2023; Bardes et al., 2024), which predict future states and evaluate trajectories entirely in latent space (Zhou et al., 2024; Sobal et al., 2025; Maes et al., 2026). Yet, because these representations are trained solely for prediction, a critical question arises: does higher predictive accuracy alone translate to superior planning performance?

Empirically, we observe that optimizing for prediction alone fails to produce a latent space suited for planning. This is because at inference time the latent is used not only to predict the future, but also to score candidate futures via latent distance. While anti-collapse mechanisms such as SIGReg (Balestriero & LeCun, 2025) preserve variance, they do not necessarily shape the latent space for planning (Enigma Team, 2026). Effective planning instead requires the planner to discern which imagined state is genuinely closer to the goal, a property that prediction objectives do not explicitly optimize. As a result, planning performance degrades sharply as the goal moves farther away: on Cube, LeWM (Maes et al., 2026) succeeds on over 75% of tasks when the goal is 10 steps away, but almost never when it is 50 steps away (Fig. 1, bottom right). A two-room navigation task illustrates this failure (Fig. 1, left): the LeWM latent distance is meaningful only near the goal, providing no useful guidance from distant starting states.

To address the long-horizon planning challenge, we improve both the distance used by the planner and the range over which it must remain reliable. Specifically, we first shape the latent geometry of the world model. We use temporal progress in offline trajectories as supervision: given any future frame as a target, states that are fewer steps away from it should lie closer to it in latent space. We impose this ordering with a RankNet-style Burges et al. (2005) ranking objective directly on the world model's own latent space. Crucially, this requires no additional value function or planning head; at test time, the planner continues to compute the same latent Euclidean distance, but in a space explicitly trained to order states by progress toward the goal.

While shaping extends the range over which latent distances remain informative, it cannot do so indefinitely: for distant goals, the cost signal becomes uninformative and prone to local minima. We therefore replace direct long-horizon search with a sequence of learned latent subgoals. Rather than placing subgoals at rigid temporal intervals (Park et al., 2023), we train our proposer with a flexible schedule. Because trajectories reach the same functional stage of a task at varying times, regressing subgoals to fixed time steps blends distinct task phases and can produce out-of-distribution states. Inspired by Explorative Modeling (Gladstone et al., 2026), we instead match each subgoal to the nearest real state within a coarse temporal window. This enables the proposer to discover meaningful task landmarks shared across trajectories, such as traversing a doorway or grasping an object.

We name the resulting method TREKWM (**T**emporal **R**anking with **E**xplorative **K**eystates). Its two core components act synergistically within the shared latent space: geometric shaping extends the reliable range of latent distances, while the subgoal proposer ensures that every planning horizon remains within this reliable regime. Furthermore, the learned subgoals discover intrinsic task structure without human annotations: in the Cube environment, they naturally factor trajectories into distinct approach, grasp, transport, and placement phases rather than blending them (Fig. 4).

In summary, our main contributions are as follows:

- **Latent metric shaping:** We shape the world model's latent space using a temporal ranking objective, aligning Euclidean latent distance with actual goal progress.
- **Adaptive subgoals:** We decompose long-horizon tasks into latent subgoals trained with a flexible schedule, so that subgoals can settle on landmark states shared by trajectories rather than on fixed time steps.
- **Complementarity and long-horizon gains:** We show that geometry shaping and subgoal proposal are complementary. Across eleven settings, TREKWM improves the average success rate by 31.4 percentage points over the strongest baseline, and reaches 67.3% on Cube at task horizon 50, where every baseline is at or below 0.7%.

2 Related Work

Latent world models. World models have long been used for planning or control, from recurrent latent models and the Dreamer family (Ha & Schmidhuber, 2018; Hafner et al., 2018; 2020; 2023) to reward-predictive models for model-predictive control such as TD-MPC2 (Hansen et al., 2023). Some line of work scales visual fidelity (Micheli et al., 2022; Alonso et al., 2024; Bruce et al., 2024; Valevski et al., 2024; Bar et al., 2024; NVIDIA et al., 2025). Our setting follows a reconstruction-free line: Joint-Embedding Predictive Architectures (JEPAs) predict in representation space rather than raw pixels (Assran et al., 2023; Bardes et al., 2024), making the latent representation itself the interface to planning. DINO-WM plans with feature-space distance over frozen DINOv2 features (Zhou et al., 2024; Oquab et al., 2023); PLDM jointly learns the representation and latent dynamics from pixels (Sobal et al., 2025); and LeWM stabilizes end-to-end JEPA world model learning with SIGReg (Maes et al., 2026; Balestriero & LeCun, 2025). V-JEPA 2 extends JEPA-style world modeling to action-conditioned robot control (Assran et al., 2025), and concurrent work shows that centering the SIGReg target in time improves LeWM representations for policy learning (Liu et al., 2026a). Other work improves long-horizon prediction (Du et al., 2026) or amortizes planning in LWM latent spaces (Nguyen et al., 2026; Huang et al., 2026a). We build on LeWM, focusing on how to improve long-horizon goal reaching through a better latent space and hierarchical planning with generated subgoals.

Shaping latent geometry for planning. Many methods learn representations whose distances reflect temporal or goal-directed structure, through time-contrastive learning (Sermanet et al., 2018; Nair et al., 2022), distance or value estimation (Venkattaramanujam et al., 2019; Yang et al., 2020; Ghosh et al., 2018; Ma et al., 2022; Park et al., 2024b), or quasimetric and contrastive objectives (Liu et al., 2022; Wang et al., 2023; Eysenbach et al., 2022; Myers et al., 2024; 2025; Bae et al., 2024), but mostly for goal-conditioned policies or value functions rather than for the latent space of a world model. Closer to our setting, recent works target the planning cost of LWMs: TRM learns a separate reachability metric (Li et al., 2026a), Traj-LeWM adds a trajectory-level cost (Huang et al., 2026b), RC-aux and DA-LeWM add planning-aligned auxiliary supervision to LeWM (Li et al., 2026b; Wang et al., 2026), Value-guided JEPA shapes the embedding itself so that latent distance approximates a negative goal-conditioned value (Destrade et al., 2025), and concurrently, ProWorld orders states by goal progress in a hyperbolic latent space (Liu et al., 2026b) and Temporal-Distance JEPA regresses a directed temporal distance through a separate head and plans with either this head or latent Euclidean distance (Bai & Xiong, 2026). In contrast, our objective supervises only the ordering of states, directly on the Euclidean distance of the world model's own embedding.

Subgoals and hierarchical planning. A complementary strategy is to shorten each planning problem, for example by planning over landmark states (Savinov et al., 2018; Eysenbach et al., 2019; Zhang et al., 2021), predicting subgoals with a high-level policy (Hafner et al., 2022; Park et al., 2023), deriving subgoals from the geometry of a Laplacian representation (Shehmar et al., 2026), or generating intermediate states (Pertsch et al., 2020; Jurgenson et al., 2019; Nasiriany et al., 2019; Chane-Sane et al., 2021; Janner et al., 2022; Chen et al., 2024); image-editing diffusion models have also been used to generate subgoal images for manipulation (Black et al., 2024). Recent work brings this idea to latent world models: HWM and FF-JEPA pass latent subgoals from a high-level model to a low-level planner (Zhang et al., 2026; Masip et al., 2026), and concurrently, Mind the Gap finds that subgoal quality is the main bottleneck in such hierarchies (Caselli et al., 2026). SAGE, also concurrent, generates latent subgoals but uses them to condition action generation rather than as targets for the planner (Cheng et al., 2026). Our method differs in two ways. Unlike HWM and Mind the Gap, which predict subgoals with a high-level world model and select them by searching under the latent distance, it generates the full chain directly from the start and goal. And unlike methods that regress to a fixed temporal target (Park et al., 2023), which imposes a single schedule on trajectories that progress at different paces, it is trained with a winner-take-all objective (Guzman-Rivera et al., 2012; Li & Malik, 2018; Gladstone et al., 2026) that matches each subgoal to its closest real state within a coarse stage, which favors states that trajectories commonly pass through at that stage, such as a doorway.

These two lines of work have largely been developed separately for LWMs: methods that shape the planning cost still plan toward a single distant goal, while hierarchical methods plan in an unshaped latent space. We combine both in a single framework and show they are complementary (Sec. 5.1).

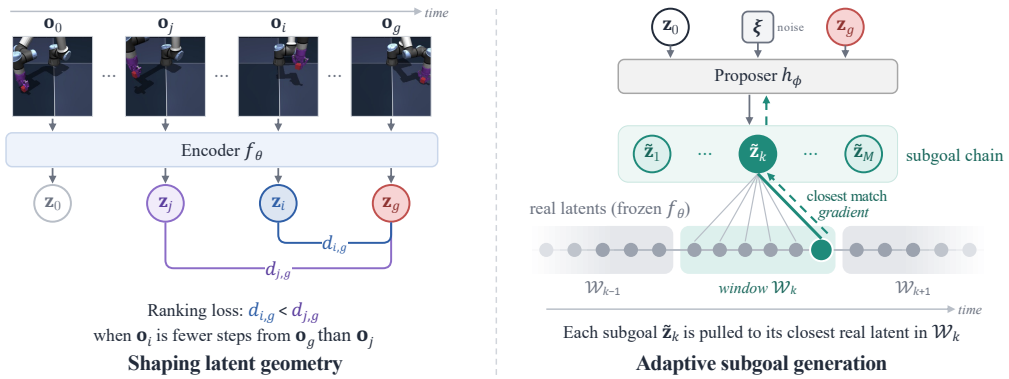

Figure 2: **TREKWM Overview.** *Left:* Besides the prediction objective, the world model is also trained a ranking objective of Sec. 3.2: for a goal latent $\mathbf{z}_g$, temporally nearer latents $\mathbf{z}_i$ are encouraged to have smaller distances $d_{i,g}$ than farther states $\mathbf{z}_j$ or cross-trajectory negatives. *Right:* After training the encoder f_θ, the proposer $h_\phi(\mathbf{z}_0, \mathbf{z}_g, \boldsymbol{\xi})$ of Sec. 3.3 maps the start and goal latents to subgoals $\tilde{\mathbf{z}}_{1:M}$. Each subgoal $\tilde{\mathbf{z}}_k$ is assigned a temporal window $\mathcal{W}_k$ and matched to the nearest real latent in that window.

3 METHOD

Figure 2 gives an overview of our approach to long-horizon planning with latent world models. We shape the latent geometry so that Euclidean distance reflects progress toward the goal (Sec. 3.2), and generate subgoals that break a distant goal into shorter planning steps (Sec. 3.3). Finally, we describe how the planner uses these subgoals (Sec. 3.4).

3.1 PRELIMINARIES

We consider goal-conditioned visual planning from an offline dataset of trajectories $(\mathbf{o}_{0:T}, \mathbf{a}_{0:T-1})$. Given an initial observation and a goal image, the agent uses a learned world model to plan and execute actions toward the goal. Rather than planning in pixel space, latent world models encode visual inputs into a low-dimensional representation space and learn forward dynamics:

$$\text{Encoder: } \mathbf{z}_t = f_\theta(\mathbf{o}_t), \quad \text{Predictor: } \hat{\mathbf{z}}_{t+1} = g_\psi(\mathbf{z}_t, \mathbf{a}_t). \tag{1}$$

The world model is trained using a standard latent prediction objective:

$$\mathcal{L}_{\text{pred}} \triangleq \|\hat{\mathbf{z}}_{t+1} - \mathbf{z}_{t+1}\|_2^2, \quad \hat{\mathbf{z}}_{t+1} = g_\psi(\mathbf{z}_t, \mathbf{a}_t). \tag{2}$$

We instantiate this setting with LeWM (Maes et al., 2026) (details in Appendix B). At planning time, however, latent Euclidean distance from a conventional LWM is informative only near the goal, which limits planning over long horizons.

3.2 LATENT GEOMETRY SHAPING VIA TEMPORAL RANKING

To make latent Euclidean distance reflect progress toward the goal during planning, we supervise the representation with a pairwise ranking objective (Fig. 2, left). The objective is applied directly to the world model's embedding $\mathbf{z}_t = f_\theta(\mathbf{o}_t)$, without auxiliary metric heads or test-time overhead.

Pairwise temporal ranking loss. Rather than regressing temporal distances, we enforce an ordinal structure over the latent space. Given a sampled goal frame g and two candidate frames i and j, where i is temporally closer to the goal than j, we define their respective latent distances to the goal as $d_{i,g} = \|\mathbf{z}_i - \mathbf{z}_g\|_2$ and $d_{j,g} = \|\mathbf{z}_j - \mathbf{z}_g\|_2$. We encourage $d_{i,g} < d_{j,g}$ by optimizing a RankNet-style logistic loss (Burges et al., 2005):

$$\mathcal{L}_{\text{rank}} = \sum_{S \in \{\text{intra, inter}\}} \frac{w_S}{|\mathcal{V}_S|} \sum_{(i,j) \in \mathcal{V}_S} \text{softplus}\left(\frac{d_{i,g} - d_{j,g}}{\tau}\right), \tag{3}$$

where $\mathcal{V}_{\text{intra}}$ and $\mathcal{V}_{\text{inter}}$ are the sets of pairs drawn from the same trajectory and from different trajectories, τ is a temperature, and w_S weights the two sets.

For each sampled goal frame, we sample the pair i and j in the following two ways. 1) *Intra-trajectory pairs* use only frames before the goal ($t_i, t_j \leq t_g$), since goal-reaching distances can be asymmetric (quasimetric) (Wang et al., 2023) and a trajectory only measures the forward direction. A pair is kept only if i is closer to the goal than j by a minimum relative and absolute margin. 2) *Inter-trajectory pairs* compare a frame a few steps before the goal with a frame from another episode in the batch. These pairs greatly increase the number of training pairs, since any frame from another episode in the batch can serve as a negative; restricting i to frames very close to the goal makes false negatives rare. Thresholds and sampling details are given in Appendix B.

Spectral regularization. Isotropic regularizers such as SIGReg used in LeWM (Balestriero & LeCun, 2025) pull the embeddings toward an isotropic Gaussian, which we found overly restrictive: the ranking loss $\mathcal{L}_{\text{rank}}$ needs some directions to be more prominent than others. We therefore replace SIGReg with a two-sided spectral hinge that guards against dimensional collapse and unbounded growth without imposing isotropy:

$$\mathcal{L}_{\text{spec}} = \frac{1}{K}\sum_{k=1}^{K} \frac{[\epsilon - \lambda_k]_+}{\epsilon} + \frac{[\overline{z^2} - c]_+}{c}, \tag{4}$$

where λ_k is the k-th eigenvalue of the batch covariance, $\overline{z^2}$ is the mean squared coordinate of the uncentered batch, ϵ sets a floor on the eigenvalues, and c caps the overall scale. The hinge is inactive in normal training and only acts when the embedding approaches collapse or grows too large, leaving the ranking loss free to shape the geometry.

Total objective. The encoder f_θ and predictor g_ψ are jointly optimized via the total loss $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pred}} + \alpha\,\mathcal{L}_{\text{rank}} + \beta\,\mathcal{L}_{\text{spec}}$, where all objectives are computed directly over the shared latent space $\mathbf{z}_t$.

3.3 Adaptive subgoals

Shaping extends the range over which latent distance is informative, but a distant goal may still lie outside it. We therefore replace a single distant goal with a chain of nearby subgoals. We train a subgoal proposer h_ϕ after the world model, with the encoder f_θ frozen, so that subgoals live in the same latent space the planner uses. Given the start and goal embeddings and a Gaussian noise input $\boldsymbol{\xi}$, the proposer generates M ordered subgoals in a single forward pass:

$$\tilde{\mathbf{z}}_{1:M} = h_\phi(\mathbf{z}_0, \mathbf{z}_g, \boldsymbol{\xi}), \qquad \boldsymbol{\xi} \sim \mathcal{N}(0, I). \tag{5}$$

Training objective. We adapt the reverse objective of Explorative Modeling (Gladstone et al., 2026), which matches each generated sample to the closest of several real targets and applies the loss only to that pair. Consider a training segment with latents $\mathbf{z}_0, \ldots, \mathbf{z}_T$, where $\mathbf{z}_0$ is the start and $\mathbf{z}_T$ is the goal. Instead of regressing subgoal k to the latent at the fixed time $kT/(M+1)$, we take a window $\mathcal{W}_k$ of time steps around it and train $\tilde{\mathbf{z}}_k$ toward the closest latent in that window:

$$\mathcal{L}_{\text{rev}} = \frac{1}{M}\sum_{k=1}^{M} \min_{t \in \mathcal{W}_k} \|\tilde{\mathbf{z}}_k - \mathbf{z}_t\|_2^2 . \tag{6}$$

We refer to Eq. 6 as the flexible-schedule objective. It lets each subgoal adapt to the pace of the trajectory rather than prescribing a single temporal schedule, and encourages it to stay close to a real state, which makes it a reliable target for the planner as shown in Tab. 3 and Tab. 4. The windows also prevent all subgoals from collapsing to the same state, retaining a coarse progression from start to goal (Fig. 4). Architecture, window construction and sampling details are given in Appendix C.

3.4 Planning with adaptive subgoals

We use the same model-predictive control planner as LeWM, based on the cross-entropy method (CEM) (Rubinstein, 1999). At each replanning step, CEM samples action-block sequences, rolls them out through the frozen latent dynamics, and searches for

$$\mathbf{a}^*_{1:h} = \arg\min_{\mathbf{a}_{1:h}} J(\mathbf{a}_{1:h}), \qquad J(\mathbf{a}_{1:h}) = \|\hat{\mathbf{z}}_h - \mathbf{z}^\star\|_2^2 , \tag{7}$$

where $\hat{\mathbf{z}}_h$ is the latent predicted after executing $\mathbf{a}_{1:h}$ and $\mathbf{z}^\star$ is the current target. The first blocks of $\mathbf{a}^*_{1:h}$ are executed before replanning. Without subgoals, the target is always the goal embedding $\mathbf{z}^\star = \mathbf{z}_g$. With adaptive subgoals, the proposer generates a sequence of subgoals $\tilde{\mathbf{z}}_{1:M}$ at the start of the episode, and a pointer $k \in \{0, \dots, M\}$ tracks how many subgoals have been passed. The target is the next subgoal in the chain, $\mathbf{z}^\star = \tilde{\mathbf{z}}_{k+1}$; once $k = M$, the target is the final goal, $\mathbf{z}^\star = \mathbf{z}_g$. The agent never observes intermediate goal images; the subgoals exist only as latent targets.

The pointer k is advanced without a latent-distance threshold, which would require environment-specific calibration. Instead, we treat the subgoal nearest to the current latent as the one most recently passed, and set the target to the subgoal after it. This rule relies on latent distances being meaningful, which our shaped latent encourages. A time-based fallback additionally advances the pointer so that the planner cannot get stuck on a single subgoal. Full planner hyperparameters and the exact advancement rule are given in Appendix B.3.

4 Experiments

4.1 Experimental setup

Environments. We evaluate on five environments: Diverse Maze (Zhang et al., 2026) and TwoRoom (Sobal et al., 2025) for navigation, PushT (Chi et al., 2023) and Cube from OGBench (Park et al., 2024a) for contact-rich manipulation, and Reacher (Tassa et al., 2018) as a local precision-control reference. TwoRoom uses cross-room tasks that require passing through the doorway, and Cube uses only tasks in which the cube must be moved (Appendix A.1).

Protocol. Each task resets the agent to a state of a held-out trajectory and gives an image of a later state as the goal. Importantly, we focus on testing long-horizon planning, and use task horizons (i.e., start-goal separations) of up to 100 steps, up to four times the 25-step horizon used to evaluate LeWM (Maes et al., 2026). Diverse Maze instead follows HWM (Zhang et al., 2026): tasks are grouped by start-goal distance on held-out layouts, with a fixed 500-step budget (Appendix A.2).

Baselines. We compare with latent world models that plan directly toward the goal: DINO-WM (Zhou et al., 2024), PLDM (Sobal et al., 2025), LeWM (Maes et al., 2026) (our base model), and the concurrent Temporal-JEPA (Bai & Xiong, 2026), which adds a learned temporal-distance cost. We further compare with HWM (Zhang et al., 2026), a hierarchical latent world model that plans subgoals with a high-level model. All methods are trained on the same data and evaluated on the same tasks. Details are given in Appendix A.3.

4.2 Long-Horizon planning evaluation

Table 1 reports success rates across all environments, and TrekWM performs best in ten of the eleven settings often by a wide margin. The clearest separation occurs on long-horizon manipulation: on Cube at task horizon 50, every baseline is at 0.7% or below, while TrekWM reaches 67.3%. The same pattern appears at the longer PushT and TwoRoom horizons, where TrekWM exceeds the strongest baseline by 37.4 and 45.3 points, respectively. On Diverse Maze, it leads across all three distance bands, including the far band on held-out layouts.

Fig. 3 shows one successful rollout of TrekWM in each environment, with subgoals decoded to images for visualization only. The subgoals mark meaningful intermediate stages of each task, such as grasping the cube before transporting it or reaching the doorway in TwoRoom, and the executed trajectory passes through them on the way to the goal.

5 Ablations and analysis

5.1 Ablations

Table 2 varies two choices independently: the latent space used for planning (unshaped or shaped) and the target given to the planner (the final goal or adaptive subgoals). The first row is the LeWM baseline, and the last row is the full TrekWM. Full per-setting results are in Appendix Table 10.

Table 1: **Main results.** Success rate (%) over 150 tasks. Task horizons are start–goal separations in environment steps, or start–goal distance bands for Maze. **Bold**: best; underline: second best.

Environment	Diverse Maze			PushT		Reacher		TwoRoom		Cube		Avg.
Task horizon	near	mid	far	50	100	50	100	50	100	25	50	all
DINO-WM	–	–	–	38.7	13.3	77.3	74.0	86.7	43.3	62.7	0.7	–
PLDM	30.0	17.3	6.7	15.3	4.7	55.3	63.3	70.0	53.3	19.3	0.0	30.5
LeWM	9.3	6.0	3.3	36.0	11.3	50.7	53.3	24.0	16.7	26.7	0.0	21.6
Temporal-JEPA	52.7	28.7	14.7	32.7	12.7	**90.7**	93.3	85.3	54.7	40.7	0.0	46.0
HWM	20.0	4.7	6.7	41.3	23.3	44.0	52.0	38.7	23.3	20.0	0.0	24.9
TREKWM (Ours)	**80.7**	**59.3**	**34.7**	**79.3**	**60.7**	86.0	**96.0**	**100.0**	**100.0**	**87.3**	**67.3**	**77.4**

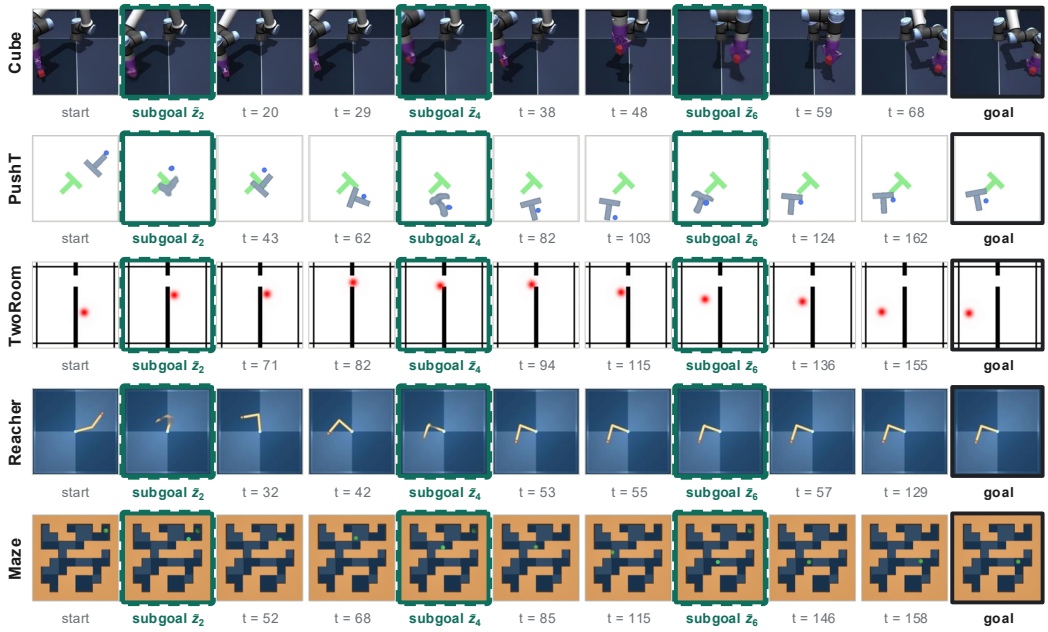

Figure 3: **Executions of TREKWM with adaptive subgoals.** Each row shows a successful task with selected observations, the goal (black frame), and three of seven latent subgoals (dashed green frames). Subgoals are decoded to pixels for visualization only.

The two components help in different environments. In the LeWM latent, adding subgoals raises the average success rate from 21.6% to 46.7%, with large gains on PushT, Cube, TwoRoom, and Reacher. Latent shaping alone raises the average to 57.8%, with large gains on TwoRoom, Reacher, and Maze.

The full method has the best average (77.4%) and is best or comparable in every environment. The interaction is clearest on Cube at task horizon 50 (Appendix Table 10): subgoals alone reach 32.0%, latent shaping alone reaches 0.7%, and their combination reaches 67.3%. A similar effect appears on the harder maze bands and TwoRoom at task horizon 100. In contrast, most of the PushT gain already comes from subgoals, while on Reacher and TwoRoom at task horizon 50, the shaped latent without subgoals is comparable to the full method.

Thus adaptive subgoals reduce the distance considered at each planning step, while latent shaping increases the range over which distance remains useful. Either component can be sufficient when it makes the task local enough. Their combination matters when the original goal is too far for direct planning and the unshaped latent is still inadequate for the shorter subgoal transitions.

5.2 ADAPTIVE VS. FIXED-SCHEDULE SUBGOALS

We now ablate the training objective of the adaptive subgoal proposer and compare it against a fixed-schedule regression baseline, where each subgoal targets the state at a fixed time. Table 3

Table 2: **Ablations on Latent shaping and adaptive subgoals.** Success rate (%) averaged over the task horizons of each environment, and over all eleven settings (Avg.). The full per-setting results are in Appendix Table 10.

	Shaped latent	Adaptive subgoals	PushT	Cube	TwoRoom	Reacher	Maze	Avg.
LeWM	✗	✗	23.7	13.4	20.4	52.0	6.2	21.6
	✗	✓	63.4	58.0	44.4	77.0	9.6	46.7
	✓	✗	29.4	29.7	93.4	**93.3**	48.2	57.8
TREKWM (Ours)	✓	✓	**70.0**	**77.3**	**100.0**	91.0	**58.2**	**77.4**

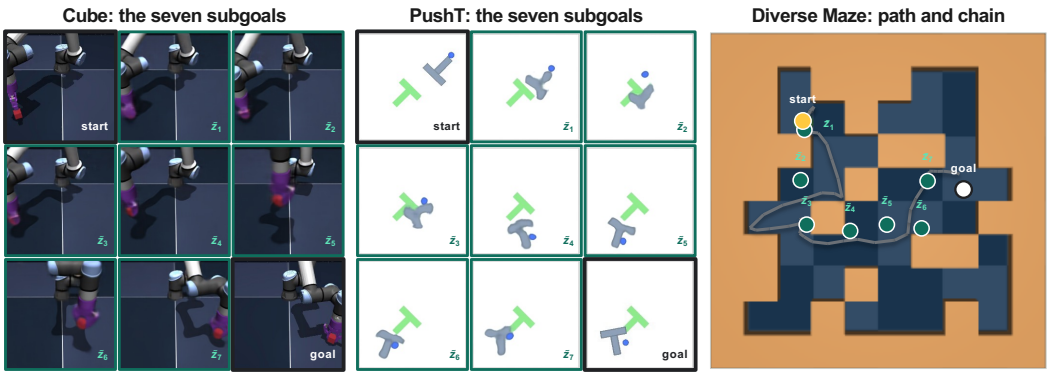

Figure 4: **Learned subgoal structure.** *Left:* decoded Cube subgoals follow approach, grasp, transport, and placement. *Middle:* PushT subgoals trace the contact-rich pushing sequence toward the target pose. *Right:* planning succeeds on a held-out maze. The gray line indicates the executed path.

shows that our flexible schedule improves over the fixed schedule in all three settings, with the largest gain on Cube. This is because, at the same fraction of time, different demonstrations may be in different phases of a task, such as approaching, grasping, or lifting the cube. Fixed-schedule regression averages over these phases and can produce a state that no demonstration passes through. Our flexible schedule instead matches each subgoal to the closest real latent in its temporal window. This pulls the subgoal toward a state that trajectories share at that stage, while the windows keep successive subgoals spread along the trajectory.

To further understand what the two objectives learn, we train proposers that emit a single subgoal on TwoRoom and read out where it lands on held-out cross-door tasks (Table 4). With the fixed schedule the groundtruth subgoal is the state precisely halfway through the demonstration. In this case, it lands in the doorway on 60% of the tasks, on an arc just below the door, not far above the demonstrations' own midpoint frames. With the flexible schedule the subgoal settles in the doorway on 99% of the tasks. The objective picks out the state that every crossing must pass through, rather than the state reached at a fixed time.

Figure 4 shows the resulting structure. Cube subgoals mark the approach, grasp, transport, and placement phases. PushT subgoals track the contact sequence for moving the object toward the target pose. On a held-out maze, the chain largely follows the corridors.

5.3 SENSITIVITY TO CHAIN LENGTH

Figure 5 varies the number of subgoals M, retraining the proposer for each value. With the shaped latent, performance is relatively insensitive once the chain has a few landmarks: Cube already reaches 48.7% at $M = 3$ and rises to 67.3% at $M = 7$, while TwoRoom reaches 100% for all $M \geq 3$. Without the shaped latent, the method depends much more strongly on the decomposition. Increasing M improves both, but even at $M = 7$ it remains far below the shaped latent: 32.0% versus 67.3% on Cube and 52.0% versus 100% on TwoRoom. Thus additional subgoals help by shortening each planning leg, but they do not by themselves fix a poorly ordered latent distance. With a shaped latent, a small number of landmarks is already enough; without a shaped latent, success depends more strongly on the particular decomposition.

Table 3: **Proposer supervision.** Success rate (%). The fixed schedule regresses each subgoal to a fixed time; the flexible schedule uses Eq. 6.

	Cube	Diverse Maze	
Schedule	horizon 50	mid	far
Fixed	59.3	52.7	33.3
Flexible (ours)	**67.3**	**59.3**	**34.7**

Table 4: **Single-subgoal localization.** Doorway hit rate of one subgoal on 188 held-out cross-room TwoRoom tasks. Learned rows average two seeds; the others are fixed references.

Single subgoal from	In doorway (%)
Straight-line midpoint	0.0
Demonstration midpoint frame	41.5
Fixed schedule	59.8
Flexible schedule (ours)	**99.2**

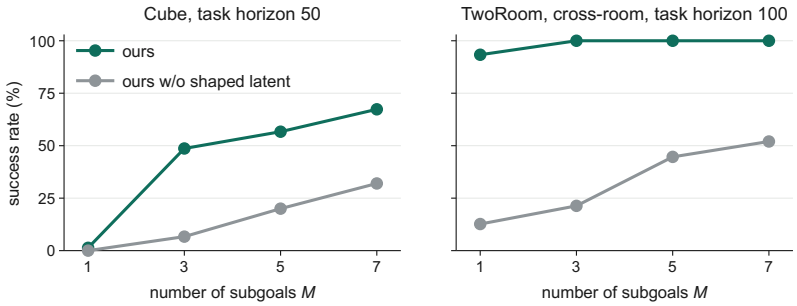

Figure 5: **Success rate against the number of subgoals** M on Cube at task horizon 50 and cross-room TwoRoom at task horizon 100. With the same proposer architecture trained on each latent, the shaped latent gives consistently higher success, showing that a better latent makes subgoals more effective. It is also less sensitive to the choice of M on TwoRoom.

5.4 Additional analyses

We provide additional analyses, including HWM on both latents (Sec. A.4), its Cube waypoint (Sec. A.5), differences from the published Maze protocol (Sec. A.6), and its search budget (Sec. D.5). We also study the ranking loss, regularizer, and metric head (Sec. D.2); the episode budget and clock (Sec. D.3); and the pointer-check interval (Sec. D.4). Further comparisons use the Temporal-JEPA latent (Sec. D.6) and an HIQL-style subgoal (Sec. D.7). We finally report prediction quality (Sec. D.8) and rank correlation with time-to-go (Sec. D.9).

6 Conclusion

Our current formulation leaves several directions for future work. First, both the dynamics and the latent metric are learned from offline data, so they are reliable only where the data provides good coverage. The ranking objective does not require expert demonstrations: on Diverse Maze, random-policy episodes still provide useful local ordering (Appendix B.2). However, with high-dimensional actions, random exploration may rarely reach task-relevant states, so the data must come from a policy that covers the relevant parts of the state space. Collecting such data, for example through exploration or online fine-tuning, would broaden the applicability of our approach. Second, the flexible schedule is flexible only within each window, and the number of subgoals M is fixed in advance. Tasks whose stages differ greatly in duration may need subgoals outside these windows, and the number of subgoals needed varies across environments (Fig. 5). Learning both the number and placement of subgoals with less temporal structure is an important next step.

In this work, we study long-horizon planning in latent world models. We show that improving this setting benefits from two complementary changes: shaping the latent geometry so that Euclidean distance better reflects progress toward the goal, and replacing a single distant goal with adaptive subgoals that provide intermediate latent targets. Across five environments, TrekWM achieves the strongest performance in ten of eleven evaluated settings. Our ablations show that the two components address distinct bottlenecks, suggesting that reliable long-horizon latent planning depends both on the geometry of the planning cost and on the distance over which each planning decision must use it.

AI USE STATEMENT

We used generative AI tools for the following tasks that require disclosure under the ICLR policy: assisting with translation; supporting qualitative analysis, such as organizing observations from experiment traces and ablation outcomes; cleaning or reformatting datasets and result files for analysis and reporting; and assisting with implementation of the proposed method and its experimental code. We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide ingredients for proofs, write proofs, propose or refine hypotheses, design the research methodology or experiments, or interpret the results.

We also used generative AI tools for tasks for which disclosure is recommended: suggesting experimental parameter settings to consider, writing or modifying software code for experiments and analysis, and editing the manuscript to improve readability. All AI-assisted code and data-processing changes were inspected by the authors and checked against the intended scripts, configuration files, logs, and aggregate tables. AI-assisted text was edited by the authors, and claims and comparisons were checked against the experiments and cited sources. The authors take responsibility for the final content of this work, including text, claims, code, data organization, and artifacts produced with the aid of generative AI.

REFERENCES

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for World Modeling: Visual Details Matter in Atari. *arXiv preprint arXiv:2405.12399*, 2024. URL https://arxiv.org/abs/2405.12399.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture. *arXiv preprint arXiv:2301.08243*, 2023. URL https://arxiv.org/abs/2301.08243.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning. *arXiv preprint arXiv:2506.09985*, 2025. URL https://arxiv.org/abs/2506.09985.

Junik Bae, Kwanyoung Park, and Youngwoon Lee. TLDR: Unsupervised Goal-Conditioned RL via Temporal Distance-Aware Representations. *arXiv preprint arXiv:2407.08464*, 2024. URL https://arxiv.org/abs/2407.08464.

Jiaxin Bai and Jiaxuan Xiong. Temporal-Distance JEPA: Plan-Aware Representation Learning for Latent World Model Predictive Control. *arXiv preprint arXiv:2607.25337*, 2026. URL https://arxiv.org/abs/2607.25337.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and Scalable Self-Supervised Learning Without the Heuristics. *arXiv preprint arXiv:2511.08544*, 2025. URL https://arxiv.org/abs/2511.08544.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation World Models. *arXiv preprint arXiv:2412.03572*, 2024. URL https://arxiv.org/abs/2412.03572.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting Feature Prediction for Learning Visual Representations from Video. *arXiv preprint arXiv:2404.08471*, 2024. URL https://arxiv.org/abs/2404.08471.

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-Shot Robotic Manipulation with Pretrained Image-Editing Diffusion Models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.10639.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, et al. Genie: Generative Interactive Environments. *arXiv preprint arXiv:2402.15391*, 2024. URL https://arxiv.org/abs/2402.15391.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning (ICML)*, pp. 89–96, 2005.

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

Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-Conditioned Reinforcement Learning with Imagined Subgoals. In *International Conference on Machine Learning (ICML)*, 2021. URL https://arxiv.org/abs/2107.00541.

Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple Hierarchical Planning with Diffusion. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2401.02644.

Letian Cheng, Qi Zhang, and Yisen Wang. SAGE: Subgoal-Conditioned Action Generation for Latent World Model Planning. *arXiv preprint arXiv:2607.17973*, 2026. URL https://arxiv.org/abs/2607.17973.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. *arXiv preprint arXiv:2303.04137*, 2023. URL https://arxiv.org/abs/2303.04137.

Matthieu Destrade, Oumayma Bounou, Quentin Le Lidec, Jean Ponce, and Yann LeCun. Value-guided action planning with JEPA world models. *arXiv preprint arXiv:2601.00844*, 2025. URL https://arxiv.org/abs/2601.00844.

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

Enigma Team. The obsessed encoder. https://www.enigma.inc/posts/obsessed-encoder, July 2026. Accessed 2026-09-19.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the Replay Buffer: Bridging Planning and Reinforcement Learning. *arXiv preprint arXiv:1906.05253*, 2019. URL https://arxiv.org/abs/1906.05253.

Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive Learning as Goal-Conditioned Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2206.07568.

Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning Actionable Representations with Goal-Conditioned Policies. *arXiv preprint arXiv:1811.07819*, 2018. URL https://arxiv.org/abs/1811.07819.

Alexi Gladstone, Heng Ji, and Yilun Du. Explorative Modeling: Unlocking a Third Pretraining Axis and End-to-End Generation. *arXiv preprint arXiv:2607.27372*, 2026. URL https://arxiv.org/abs/2607.27372.

Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple Choice Learning: Learning to Produce Multiple Structured Outputs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

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. *arXiv preprint arXiv:1811.04551*, 2018. URL https://arxiv.org/abs/1811.04551.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with Discrete World Models. *arXiv preprint arXiv:2010.02193*, 2020. URL https://arxiv.org/abs/2010.02193.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep Hierarchical Planning from Pixels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2206.04114.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Domains through World Models. *arXiv preprint arXiv:2301.04104*, 2023. URL https://arxiv.org/abs/2301.04104.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, Robust World Models for Continuous Control. *arXiv preprint arXiv:2310.16828*, 2023. URL https://arxiv.org/abs/2310.16828.

Hsiang-Wei Huang, Jianxu Shangguan, Junbin Lu, and Jenq-Neng Hwang. LeFlow: Generative latent flow planning for world models. *arXiv preprint arXiv:2608.24855*, 2026a.

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

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with Diffusion for Flexible Behavior Synthesis. *arXiv preprint arXiv:2205.09991*, 2022. URL https://arxiv.org/abs/2205.09991.

Tom Jurgenson, Edward Groshev, and Aviv Tamar. Sub-Goal Trees – a Framework for Goal-Directed Trajectory Prediction and Optimization. *arXiv preprint arXiv:1906.05329*, 2019. URL https://arxiv.org/abs/1906.05329.

Ke Li and Jitendra Malik. Implicit Maximum Likelihood Estimation. *arXiv preprint arXiv:1809.09087*, 2018. URL https://arxiv.org/abs/1809.09087.

Liangyu Li, Shengzhi Wang, and Qingwen Liu. Beyond euclidean proximity: Repairing latent world models with horizon-matched trajectory reachability metrics, 2026a.

Wenyuan Li, Guang Li, Keisuke Maeda, Takahiro Ogawa, and Miki Haseyama. Predictive but not plannable: Rc-aux for latent world models. *arXiv preprint arXiv:2605.07278*, 2026b.

Bo Liu, Yihao Feng, Qiang Liu, and Peter Stone. Metric Residual Networks for Sample Efficient Goal-Conditioned Reinforcement Learning. *arXiv preprint arXiv:2208.08133*, 2022. URL https://arxiv.org/abs/2208.08133.

Chang Liu, Fei Suo, Yanzhou Jin, Zeyu Ping, Yusuke Iwasawa, Yutaka Matsuo, and Yaonan Zhu. Temporally centered sigreg improves leworldmodel representations for robot policy learning. *arXiv preprint arXiv:2607.26924*, 2026a.

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

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. *arXiv preprint arXiv:2210.00030*, 2022. URL https://arxiv.org/abs/2210.00030.

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*, 2026. URL https://arxiv.org/abs/2603.19312.

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

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are Sample-Efficient World Models. *arXiv preprint arXiv:2209.00588*, 2022. URL https://arxiv.org/abs/2209.00588.

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. *arXiv preprint arXiv:2406.17098*, 2024. URL https://arxiv.org/abs/2406.17098.

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 (NeurIPS)*, 2025. URL https://arxiv.org/abs/2509.20478.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A Universal Visual Representation for Robot Manipulation. In *Conference on Robot Learning (CoRL)*, 2022. URL https://arxiv.org/abs/2203.12601.

Soroush Nasiriany, Vitchyr H. Pong, Steven Lin, and Sergey Levine. Planning with Goal-Conditioned Policies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://arxiv.org/abs/1911.08453.

Hoang Nguyen, Xiaohao Xu, and Xiaonan Huang. Latent geometry beyond search: Amortizing planning in world models. *arXiv preprint arXiv:2605.08732*, 2026.

NVIDIA, Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos World Foundation Model Platform for Physical AI. *arXiv preprint arXiv:2501.03575*, 2025. URL https://arxiv.org/abs/2501.03575.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, et al. DINOv2: Learning Robust Visual Features without Supervision. *arXiv preprint arXiv:2304.07193*, 2023. URL https://arxiv.org/abs/2304.07193.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline Goal-Conditioned RL with Latent States as Actions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2307.11949.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking Offline Goal-Conditioned RL. *arXiv preprint arXiv:2410.20092*, 2024a. URL https://arxiv.org/abs/2410.20092.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation Policies with Hilbert Representations. *arXiv preprint arXiv:2402.15567*, 2024b. URL https://arxiv.org/abs/2402.15567.

Karl Pertsch, Oleh Rybkin, Frederik Ebert, Chelsea Finn, Dinesh Jayaraman, and Sergey Levine. Long-Horizon Visual Planning with Goal-Conditioned Hierarchical Predictors. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://arxiv.org/abs/2006.13205.

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 (ICLR)*, 2018. URL https://arxiv.org/abs/1803.00653.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-Contrastive Networks: Self-Supervised Learning from Video. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018. URL https://arxiv.org/abs/1704.06888.

Dikshant Shehmar, Matthew Schlegel, Matthew E. Taylor, and Marlos C. Machado. Laplacian representations for decision-time planning. In *International Conference on Machine Learning (ICML)*, 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. *arXiv preprint arXiv:2502.14819*, 2025. URL https://arxiv.org/abs/2502.14819.

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

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion Models Are Real-Time Game Engines. *arXiv preprint arXiv:2408.14837*, 2024. URL https://arxiv.org/abs/2408.14837.

Srinivas Venkattaramanujam, Eric Crawford, Thang Doan, and Doina Precup. Self-supervised Learning of Distance Functions for Goal-Conditioned Reinforcement Learning. *arXiv preprint arXiv:1907.02998*, 2019. URL https://arxiv.org/abs/1907.02998.

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. *arXiv preprint arXiv:2304.01203*, 2023. URL https://arxiv.org/abs/2304.01203.

Ge Yang, Amy Zhang, Ari S. Morcos, Joelle Pineau, Pieter Abbeel, and Roberto Calandra. Plan2Vec: Unsupervised Representation Learning by Latent Plans. *arXiv preprint arXiv:2005.03648*, 2020. URL https://arxiv.org/abs/2005.03648.

Lunjun Zhang, Ge Yang, and Bradly C. Stadie. World model as a graph: Learning latent landmarks for planning. In *International Conference on Machine Learning (ICML)*, 2021.

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. URL https://arxiv.org/abs/2604.03208.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2411.04983.

Supplementary Material

A Environments and evaluation protocol

A.1 Environments

The benchmark suite covers navigation, contact-rich manipulation, and local precision control. Table 5 gives the data scale and evaluation settings. The frame counts are for the training data actually used: the 95% training split for PushT, Cube, TwoRoom, and Reacher, and the 25 training layouts for Diverse Maze. For all environments except Diverse Maze, a task is specified by a temporal offset Δ in demonstration steps, which we call the task horizon in the main text. Diverse Maze instead bins tasks by shortest-path distance in grid cells, following Zhang et al. (2026). We keep the renderer fixed within each environment so that evaluation observations match the rendering backend used for the demonstrations.

Table 5: Benchmark and dataset details used in our experiments.

Environment	Demonstrations	Frames	Offsets Δ	Success criterion	Renderer
Diverse Maze	expert + random	5.39M	BFS 5–8 / 9–12 / 13–16	≤ 0.5 cells from goal	EGL
TwoRoom	scripted expert	0.88M	50, 100	within 16 units	OSMesa
PushT	expert	2.22M	50, 100	< 20 units, $< 20^\circ$	OSMesa
Cube	expert pick-place	0.64M	25, 50	block within 4 cm	OSMesa
Reacher	multi-goal expert	1.90M	50, 100	joints within 0.05 rad	EGL

Diverse Maze. We use procedurally generated 10×10 point mazes from Zhang et al. (2026). Training uses 25 layouts and evaluation uses 20 held-out layouts, so the agent must generalize to unseen maps. Tasks are binned by the BFS distance between start and goal: 5–8, 9–12, or 13–16 cells. Each bin receives a budget of 500 environment steps, and success means ending within 0.5 cells of the goal.

TwoRoom. TwoRoom contains two rooms connected by one doorway. We evaluate on cross-room tasks, where the start and goal lie on opposite sides of the doorway and are at least 100 units apart. In these tasks, Euclidean proximity points through the wall for much of the episode, while the feasible path must pass through the doorway. Success is reaching within 16 units of the goal.

PushT. PushT (Chi et al., 2023) requires a point agent to push a T-shaped object to a target pose. A rollout succeeds when the joint agent–object position error is below 20 units and the object orientation error is below 20°. The T-shaped block has no rotational symmetry, so the orientation error is the smallest angular difference over the full circle; a quarter or half turn counts as a failure.

Cube. Cube is the single-object pick-and-place task from OGBench (Park et al., 2024a). Solving it requires reaching the cube, grasping it, transporting it, and releasing it near the target. Success is the cube ending within 4 cm of its target position.

Reacher. Reacher is a two-link arm from DeepMind Control. We train on multi-goal expert trajectories, with roughly five reaches per episode, and disable collisions. Success requires both joint angles to be within 0.05 rad of the target, with angular distance measured on the circle. Because each episode contains several separate reaches, larger temporal offsets define different task pools rather than a clean long-horizon difficulty axis.

A.2 Task construction and evaluation protocol

Tasks. For PushT, Cube, TwoRoom, and Reacher, each task is built from one held-out demonstration. We reset the simulator to a demonstration frame and use the frame Δ steps later in the same trajectory as the goal image. The default episode budget is $B = 2\Delta$ environment steps. Cube instead uses $B = 100$ at both offsets, and Diverse Maze uses $B = 500$ for every distance bin. The factor of two follows LeWM's released planning configuration, which pairs a 25-step offset with a

50-step budget in every environment. We keep that rule and only raise Δ. The agent receives only image observations and the goal image

Cube tasks. For Cube, we apply two additional filters. First, success is measured by the block position, but the block is stationary for long stretches before grasping and after release. Sampling start and goal frames by offset alone therefore produces many already-solved windows: in the offset-25 pool, 38% of candidate windows move the block by less than the 4 cm success radius, and a random policy succeeds on 43% of tasks. Second, we truncate each episode at the first successful placement (about 70 frames), since the recorded demonstrations continue into a second pick-and-place attempt. This prevents an evaluation offset from spanning two attempts.

Split. For PushT, Cube, TwoRoom, and Reacher, every twentieth demonstration episode is held out. All evaluation tasks come from this 5% held-out set, and all world models, baselines, and proposers are trained on the remaining 95%. The proposer is trained from the latent cache of the same training split, so it does not see evaluation episodes either. For Diverse Maze, the split is by layout: 25 maps are used for training and 20 disjoint maps are reserved for evaluation.

Reporting. Each reported cell is a success rate over 150 evaluation tasks, split across disjoint planner-seed task sets. Within a row, compared arms use the same task set, renderer, step budget, CEM budget, and low-level planner whenever applicable. Chained arms use the same proposer architecture and hyperparameters; when the latent changes, the proposer is retrained on that latent's own cache.

Cost of a run. Planning is the dominant evaluation cost. Each replanning step runs 30 CEM iterations with 300 candidate action sequences, and this solver budget is fixed across Table 1.

A.3 Baselines

Table 1 compares against five baselines. Each baseline is trained from scratch on the same training split as our method and evaluated on the same held-out tasks, renderer, step budgets, and CEM budget. The details below specify the representation and planning cost used by each method. HWM additionally has a high-level planner, described in the main text.

LeWM. LeWM (Maes et al., 2026) is the base world model used throughout the paper. It uses the same architecture, data, optimizer, and training schedule as our world model, but keeps the published LeWM objective and omits our geometry shaping terms. At test time, it plans directly to the final goal image using Euclidean distance in the learned latent space.

Temporal-JEPA. Temporal-JEPA (Bai & Xiong, 2026) keeps the LeWM-style backbone and adds a directed temporal-distance objective trained from same-trajectory ordering, cross-trajectory negatives, and rollout-consistency supervision. Public checkpoints are not available, so we train it with the authors' implementation and configuration on our training split and plan it with our CEM solver. The planning cost follows the configuration Temporal-JEPA specifies per environment: the learned temporal distance alone on TwoRoom, and on PushT and Cube that distance blended with latent Euclidean distance at the reported weight of 0.1. The Temporal-JEPA paper covers three of our five environments; on Diverse Maze and Reacher we train it the same way and plan with the temporal distance alone. The released configuration trains for 10 epochs; in our benchmark we use the fixed per-environment epoch budget in Table 9 and read the final, fully annealed checkpoint.

DINO-WM and PLDM. Both are trained from pixels and actions on the same training split and planned with the same CEM budget. DINO-WM (Zhou et al., 2024) freezes a DINOv2-small encoder (Oquab et al., 2023) and trains an action-conditioned predictor over patch tokens. Its cost is the distance between predicted and goal patch features, following the original method. On Diverse Maze, this patch-feature cost makes a single 500-step evaluation task take about 35 minutes to plan, so those cells are left blank in Table 1. PLDM (Sobal et al., 2025) uses the LeWM ViT encoder and predictor, but trains the representation and dynamics end to end with the VICReg-style objective and coefficients used in the LeWM reimplementation. It plans with Euclidean distance in its learned latent space.

A.4 HWM ON BOTH LATENTS

Table 6 trains HWM (Zhang et al., 2026) on both the LeWM latent and our shaped latent, pairing each high-level model with the matching low-level planner, with the same CEM budget and replanning period as Table 1. On the LeWM latent, HWM helps in some navigation settings but leaves the hardest long-horizon failures unchanged: Cube at offset 50 remains unsolved. On the shaped latent, HWM improves where direct planning is already close, such as TwoRoom at offset 100 and the far Maze band, but still falls well short of adaptive subgoals on the contact-rich tasks. A learned hierarchy thus inherits the quality of the latent it plans in, and how intermediate targets are produced still matters.

Table 6: **HWM depends on the latent.** Success rate (%) over 150 evaluation tasks. Direct plans to the final goal; +HWM uses the first waypoint from a high-level plan; TREKWM uses generated subgoals.

		LeWM latent		Shaped latent (ours)		
Environment	Setting	Direct	+ HWM	Direct	+ HWM	+ adaptive subgoals
Diverse Maze	near, 5–8 cells	9.3	20.0	78.7	70.0	**80.7**
	mid, 9–12 cells	6.0	4.7	46.0	47.3	**59.3**
	far, 13–16 cells	3.3	6.7	20.0	34.0	**34.7**
PushT	offset 50	36.0	41.3	50.0	50.7	**79.3**
	offset 100	11.3	23.3	8.7	17.3	**60.7**
Reacher	offset 50	50.7	44.0	**89.3**	86.7	86.0
	offset 100	53.3	52.0	**97.3**	90.0	96.0
TwoRoom	cross-room, off 50	24.0	38.7	**100.0**	**100.0**	**100.0**
	cross-room, off 100	16.7	23.3	86.7	**100.0**	**100.0**
Cube	offset 25	26.7	20.0	58.7	60.0	**87.3**
	offset 50	0.0	0.0	0.7	0.7	**67.3**

A.5 WHAT THE HWM WAYPOINT ASKS FOR ON CUBE

On Cube at offset 50, HWM fails even on the shaped latent (0.7%), while adaptive subgoals reaches 67.3% with the same low-level planner (Table 6). The difference is therefore not only whether an intermediate target is used, but what state that target asks the planner to reach. To inspect the targets, we fit a ridge probe from latent states to hand and block positions on the training latent cache. The probe has median errors of 1.8 cm for the hand and 2.0 cm for the block on held-out evaluation frames. We then replay the first planning step on all 150 offset-50 tasks and decode the first target passed to the low-level planner (Table 7).

The first feasible phase of a Cube trajectory is to reach the block. In the expert demonstrations, the hand starts 17.7 cm from the block; after ten steps, it has moved 13.2 cm and is within 3.3 cm of a block that has not yet moved. HWM's planned waypoint instead mixes this approach phase with the later transport phase: the decoded hand is still 6.3 cm from the block, but the block has already moved 7.8 cm toward the goal. This is a contact violation in 54% of tasks under the criterion in Table 7. The low-level planner can move the hand toward such a target, but it cannot move a block that the hand has not reached, so the rollout stalls before useful contact.

This effect is already present in the high-level predictor. When driven by the expert's own macro-actions, rather than by outer CEM, the predictor still violates the contact condition in 40% of tasks. Search makes the issue worse: the planned HWM waypoints are 2.52 latent units from the nearest real training frame, compared with 0.67 for real held-out frames. By contrast, the first adaptive subgoals target asks for the contact-making phase: the hand moves 15.1 cm, ends 2.6 cm from the block, and the block moves only 1.8 cm. No first handed-down slot violates contact, only 2% of all generated slots do, and the slots remain close to real frames (0.92). This is the behavior the flexible-schedule objective is designed to encourage: each slot is pulled toward a real state from the corresponding part of the demonstration, so the generated chain preserves the task phases instead of averaging approach and transport into a single unreachable target.

Table 7: **Decoded first targets on Cube, offset** 50**.** Medians over 150 evaluation tasks. A target violates contact when the block has moved by more than 3 cm while the hand is still more than 5 cm from it. The last column is the latent distance to the nearest real training frame; real held-out frames are at 0.67. HWM waypoints are either selected by the outer CEM planner or produced by the predictor under the expert's own macro-actions.

	First target (cm)			Contact violated (%)		Latent
Target	hand travel	to block	block shift	first	all	to real
Expert, 10 steps ahead	13.2	3.3	0.0	0	0	0.67
HWM, planned	8.9	6.3	7.8	54	15	2.52
HWM, expert macro-actions	11.6	4.2	3.3	40	8	1.42
adaptive subgoals, handed slot	15.1	2.6	1.8	0	2	0.92

A.6 HWM ON DIVERSE MAZE: OUR NUMBERS VERSUS THE PUBLISHED ONES

Diverse Maze is also one of HWM's benchmark domains, but the HWM numbers in Table 6 should not be read as a reproduction of the published Diverse Maze table. Zhang et al. (2026) report $100/84/44\%$ for their single-level PLDM planner and $100/95/83\%$ for the hierarchical planner on the near, mid and far bands. In Table 6, HWM is instead evaluated as a baseline inside our common benchmark: the high-level model is retrained on the latent cache of the low-level model it is paired with, and it uses the same tasks, image-only observations, low-level planner budget and evaluation protocol as the other rows.

The lower absolute numbers in our table mainly reflect protocol differences. As a sanity check, we ran the authors' released code and checkpoints without modification on their own evaluation (40 tasks per band, 3 seeds). We obtained $94/65/51\%$ for the single-level planner and $100/88/88\%$ for the hierarchical planner, close to the published $100/84/44\%$ and $100/95/83\%$.

Table 8 summarizes the main differences. The maze generator, layout split, distance bands, 500-step budget, and 0.5-cell success threshold are shared. The observation stream, data distribution, model class and planner differ: HWM's published stack uses proprioceptive velocity, a small convolutional PLDM model, and much longer MPPI searches, while our benchmark uses the image-only LeWM-style stack and the common CEM budget from Tables 1 and 6. Thus Table 6 should be read as a controlled comparison within our stack, not as a reproduction of HWM's released system.

Table 8: Diverse Maze as evaluated by Zhang et al. (2026) and in this paper. Our column is shared by every method in Tables 1 and 6.

	Zhang et al. (2026)	This paper
Layouts	8×8 interior (10×10 with wall), 25 train / 20 test	same generator, same split
Bands, budget, success	5–8 / 9–12 / 13–16 cells, 500 steps, 0.5 cells	same
Observation	98×98 image + proprioceptive velocity	224×224 image only
Training data	random policy, 2,000 $\times$ 100 steps per map (5M)	expert + random, 5.4M frames
Low-level model	PLDM CNN (33k + 20k params), VICReg, 3 epochs	LeWM ViT-tiny, 5 epochs
High-level model	conv predictor, stride 10, 6 waypoints	latent predictor, stride 25
Planner	MPPI, replanned every 4 steps	CEM 300 $\times$ 30, replanned every 25 steps
Single-level horizon	150–250 steps	25 steps
Hierarchical search	2,000–4,000 samples, 25–47 macro-steps	300 samples, 10–19 macro-steps
Tasks per band	40	150 evaluation tasks

Within this common stack, HWM is most useful once the latent geometry is improved, which is the comparison used in the main text.

Table 9: Per-environment settings. E is the number of extra frames drawn per item for Eq. 3; $n_{\max}$ is the near-set radius for cross-trajectory pairs, i.e., only frames at most $n_{\max}$ steps before the goal are paired with frames from other episodes. h is the CEM horizon in action blocks and r is the number of executed blocks; each block contains 5 primitive actions, so CEM looks ahead $5h$ environment steps and replans every $P = 5r$ steps.

	World model				Planner	
Environment	Demonstrations	Epochs	E	$n_{\max}$	$h \times r$	P
Diverse Maze	expert + random, 25 maps	5	6	5	5×5	25
PushT	expert	30	6	5	5×5	25
Reacher	expert, multi-goal	20	6	5	5×5	25
TwoRoom	demonstrations	20	6	5	5×5	25
Cube	expert, pick-and-place	30	6	5	2×2	10

B Implementation details

B.1 Shared hyperparameters

Unless stated otherwise, world models share the same optimizer and schedule: AdamW with learning rate 5×10^{-5}, weight decay 10^{-3}, batch size 128, bf16 precision, gradient clipping at 1.0, linear warmup over the first 1% of steps, and cosine decay. Following HWM (Zhang et al., 2026), we set the number of training epochs before evaluation and read the final checkpoint from that budget. The fixed epoch budgets are listed in Table 9. The batch-coupled regularizer in Eq. 4 is computed on the full batch, so world models are trained on a single GPU. Proposers are trained for 10^5 steps with AdamW, learning rate 3×10^{-4}, weight decay 10^{-2}, batch size 256, and a cosine schedule. The metric-shaping constants are shared across environments: the gate uses $\rho = 2$ and $\delta_{\min} = 10$ environment steps, the temperature is $\tau = 1$, each item draws $K = 8$ inter-trajectory frames, and the pair weights are $w_{\text{intra}} = 0.1$ and $w_{\text{inter}} = 0.5$. The spectral hinge uses $\kappa = 10^{-2}$ and $c = 4$ with $\beta = 1$. The near-set radius $n_{\max}$ is 5.

The subgoal-chain settings are otherwise shared: $M = 7$ slots, one chain sampled per episode, pointer checks every $C = 10$ environment steps, and F final replanning cycles reserved for the true goal. We set F to two cycles or 20% of the episode's replanning cycles, whichever is larger (Sec. B.3). The remaining environment-specific quantities are listed in Table 9.

B.2 Per-environment settings and sampling

Table 9 lists the quantities that vary across environments. All settings use $h = r$, so each planned sequence is executed in full before the next CEM call; no action tail is carried over between replanning steps.

Span sampling for the proposer. For each proposer example we first choose an episode, then sample a start-goal span log-uniformly and clip it to the episode length. The start index t_0 is uniform over valid positions for that span, and t_1 is the resulting goal index, giving endpoints $\mathbf{z}_0 = \mathbf{Z}_{t_0}$ and $\mathbf{z}_g = \mathbf{Z}_{t_1}$. Log-uniform sampling exposes the proposer to shorter and longer segments.

Random-policy data in Diverse Maze. Diverse Maze is the only environment where the world model is trained on both expert and random-policy episodes. The random rollouts broaden local dynamics coverage: shortest-path experts move directly through the maze, while receding-horizon plans at test time often contain shorter corrective motions.

For the world-model objective, random episodes are sampled in the same way as expert episodes. The goal frame and the E extra frames in Eq. 3 are drawn from the episode that supplies the training window. Temporal order is a noisier progress signal on a random walk, but the gate removes close or ambiguous pairs before they enter the ranking loss. As a sanity check, among gate-valid random-walk pairs, temporal order agrees with straight-line progress toward the episode endpoint 84.5% of the time, versus 50% for an uninformative label. The proposer is trained only on expert episodes, since random rollouts do not define goal-directed intermediate states.

B.3 PLANNER AND SUBGOAL ADVANCEMENT

The planner is the same model-predictive controller used by the base world model. At each replanning step, CEM samples 300 action-block sequences of length h. Each block contains $\kappa = 5$ primitive actions. The frozen latent dynamics rolls out every candidate, the terminal state is scored by $\|\hat{\mathbf{z}}_h - \mathbf{z}^\star\|_2^2$, and CEM refits its sampling distribution to the best 30 candidates for 30 iterations. The first r blocks are executed before replanning, so the replanning period is $P = r\kappa$ environment steps. The values of h and r are listed in Table 9. Without adaptive subgoals, the target remains $\mathbf{z}^\star = \mathbf{z}_g$ for the whole episode.

With adaptive subgoals, we draw one noise vector at the start of the episode and generate a chain $\tilde{\mathbf{z}}_{1:M}$ with $M = 7$ from $(\mathbf{z}_0, \mathbf{z}_g)$. A monotone pointer $k \in \{0, \dots, M\}$ chooses the active target: $\mathbf{z}^\star = \tilde{\mathbf{z}}_{k+1}$ for $k < M$ and $\mathbf{z}^\star = \mathbf{z}_g$ for $k = M$. The pointer is reconsidered every $C = 10$ environment steps using two signals that do not require a calibrated distance threshold. The first signal is the subgoal nearest to the current latent,

$$j^\star = \arg\min_{1 \le j \le M} \|\mathbf{z}_t - \tilde{\mathbf{z}}_j\|_2 , \tag{8}$$

which marks the latest slot the trajectory appears to have passed. The second signal is a proportional clock. With episode budget B and F replanning cycles reserved for the final goal, let $t_{\text{eff}} = B - FP$ and

$$k_{\text{clock}}(t) = \begin{cases} \left\lfloor \dfrac{t(M+1)}{t_{\text{eff}}} \right\rfloor, & t < t_{\text{eff}}, \\ M, & t \ge t_{\text{eff}}. \end{cases} \tag{9}$$

The update rule is $k \leftarrow \max\big(k,\, j^\star,\, \min(k_{\text{clock}}(t), M)\big)$. This makes the pointer monotone and forces the final F replanning cycles to target $\mathbf{z}_g$. The number of reserved cycles is $F = \max(2,\, \lfloor 0.2\, B/P \rfloor)$: two cycles or 20% of the replanning cycles, whichever is larger. This gives two reserved cycles in every setting of Table 1 except Diverse Maze, where the 500-step budget contains 20 replanning cycles and reserves four.

C ADDITIONAL DETAILS FOR THE ADAPTIVE SUBGOALS PROPOSER

C.1 LATENT CACHE, ARCHITECTURE AND WINDOWS

Latent cache and training tuples. After training the world model, we freeze the encoder and encode each demonstration frame once to get $\mathbf{Z} \in \mathbb{R}^{N \times d}$. The proposer is trained from this cache only; its loss does not update the encoder. Each training example selects two latents from one episode,

$$(\mathbf{z}_0, \mathbf{z}_g) = (\mathbf{Z}_{t_0}, \mathbf{Z}_{t_1}), \tag{10}$$

where the span from t_0 to t_1 is sampled as described in Sec. B.2. The proposer receives only $\mathbf{z}_0$, $\mathbf{z}_g$, and noise. Intermediate frames from the segment are used to build the targets in Eq. 6, but are not inputs to the network.

Proposer architecture. For slot k, let $f_k = k/(M+1)$. The slot is anchored at the linear interpolation between the endpoints,

$$\mathbf{A}_k = (1 - f_k)\mathbf{z}_0 + f_k \mathbf{z}_g, \tag{11}$$

and the network predicts a residual from that anchor,

$$\tilde{\mathbf{z}}_k = \mathbf{A}_k + \mathbf{R}_k. \tag{12}$$

The input sequence has $M + 2$ tokens: M slot tokens and two conditioning tokens for $\mathbf{z}_0$ and $\mathbf{z}_g$. Slot k contains a Gaussian noise vector $\boldsymbol{\epsilon}_k \sim \mathcal{N}(0, I_d)$ and a learned slot embedding. A pre-LN transformer with four blocks, width 256, and 4 attention heads processes the tokens jointly. We apply the same transformer blocks for $R = 4$ recurrent passes with shared weights, adding the embedded input back before each pass. The final slot representations are projected to $\mathbf{R}_{1:M}$. At training time each tuple uses one noise sample; at evaluation time each task uses one generated chain.

Why the flexible-schedule objective is windowed. A single target frame for slot k would force every demonstration to follow the same temporal pacing. We instead associate slot k with a short

window of real latents around fraction f_k of the segment and match the generated waypoint to the closest latent in that window.

The windows act as a coarse stage prior. Their centers move forward with k, so early slots see early parts of the segment and late slots see later parts. Within a window, however, the nearest matched frame is chosen independently. After rounding window boundaries to integer frames, neighboring windows may also share a frame on short segments. Thus the loss does not impose a strict temporal ordering on the matched frames. Ordering at deployment comes from the slot index and the monotone pointer in Sec. 3.4.

Window construction. Slot k uses a temporal window centered at $f_k = k/(M+1)$ of the training segment. We sample $T = 16$ candidate frames uniformly from that window. The half-width w is derived from the slot spacing rather than tuned: adjacent slot centers are $1/(M+1)$ apart, so neighboring windows are disjoint when $w \leq 1/(2(M+1))$. With $M = 7$, this bound is 0.0625; we use $w = 0.06$ for all environments. On short segments, integer rounding can still make adjacent windows share a frame.

D Additional analysis

In this section, we first ablate the method components, then vary planner-side choices such as budget, clock and pointer cadence, and finally report auxiliary baselines and diagnostics for the learned latent.

D.1 Full ablation of latent shaping and adaptive subgoals

Table 10 gives the per-setting results behind the averages in Table 2.

Table 10: **Latent shaping $\times$ adaptive subgoals, per setting.** Success rate (%) over 150 evaluation tasks. Direct plans to the final goal; the final column is the full TrekWM. Table 2 reports the per-environment averages.

		LeWM latent		Shaped latent	
Environment	Setting	Direct	adaptive subgoals	Direct	adaptive subgoals
PushT	offset 50	36.0	68.0	50.0	**79.3**
	offset 100	11.3	58.7	8.7	**60.7**
Cube	offset 25	26.7	84.0	58.7	**87.3**
	offset 50	0.0	32.0	0.7	**67.3**
TwoRoom	cross-room, off 50	24.0	36.7	**100.0**	**100.0**
	cross-room, off 100	16.7	52.0	86.7	**100.0**
Reacher	offset 50	50.7	74.0	**89.3**	86.0
	offset 100	53.3	80.0	**97.3**	96.0
Diverse Maze	near, 5–8 cells	9.3	18.0	78.7	**80.7**
	mid, 9–12 cells	6.0	6.0	46.0	**59.3**
	far, 13–16 cells	3.3	4.7	20.0	**34.7**

D.2 Regularizer and ranking loss

The ranking loss is the part of the shaped latent that adds planning structure: it directly trains Euclidean distance in the world-model embedding to preserve temporal order relative to a goal. The spectral hinge in Eq. 4 is a supporting regularizer. Its role is only to keep the embedding from collapsing or growing without bound while leaving the ranking loss free to make some directions more important than others. SIGReg is a stronger constraint for this purpose because it pulls batches toward an isotropic Gaussian, which can work against the anisotropy induced by ranking.

Table 11 separates these choices under direct planning. The rows choose the regularizer, SIGReg or spectral hinge, and whether the ranking loss is included; the data, planner, and evaluation tasks

otherwise follow Table 1. For comparability within this diagnostic, all rows are read at the same point in the training schedule: epoch 10 on Cube, PushT, TwoRoom and Reacher, and epoch 5 on Diverse Maze. This matched readout is different from the per-environment epoch budget used for the main table.

Table 11: Regularizer × ranking loss ablation with direct planning: success rate (%) at the matched read epoch (10 on Cube, PushT, TwoRoom and Reacher; 5 on Diverse Maze).

Environment	Diverse Maze			PushT		Reacher		TwoRoom		Cube	
Setting	near	mid	far	50	100	50	100	50	100	25	50
SIGReg, no ranking (LeWM)	9.3	6.0	3.3	40.7	12.0	56.7	52.0	30.7	16.0	18.0	**0.7**
Spectral hinge, no ranking	23.3	11.3	4.7	1.3	0.0	5.3	4.0	6.0	8.0	42.7	**0.7**
SIGReg + ranking	68.7	35.3	13.3	42.7	**14.7**	68.7	73.3	**95.3**	**89.3**	24.0	0.0
Spectral hinge + ranking (ours)	**78.7**	**46.0**	**20.0**	**46.7**	13.3	**90.7**	**96.0**	76.0	52.7	**65.3**	**0.7**

Two conclusions emerge from this comparison. First, the ranking loss provides the main improvement: adding it raises success in nearly every setting, with particularly large gains on Diverse Maze, Reacher, and TwoRoom. Second, the choice of regularizer is secondary and task dependent. With ranking, the spectral hinge performs best on all three maze ranges, both Reacher offsets, PushT at offset 50, and Cube at offset 25, whereas SIGReg is stronger on TwoRoom and PushT at offset 100. We therefore treat the spectral hinge as a lightweight default rather than an essential part of the method; its purpose is to stabilize the embedding while imposing fewer geometric constraints than SIGReg.

Embedding versus a metric head. The ranking loss can either shape the world-model embedding itself, as in TREKWM, or train a separate metric head ϕ on top of the embedding. The head variant is a useful diagnostic because it asks whether the temporal ordering only needs to be available to the planner's cost, or whether it should live in the same representation used by the predictor and the proposer. We train a two-layer MLP head with width 512, LayerNorm, and output dimension 128, using the same ranking pairs and matched read epochs as above. At evaluation time we test two costs: Euclidean distance in the underlying embedding, and Euclidean distance in the head space.

Table 12 compares direct embedding shaping with a metric head under the same spectral-hinge regularizer. TREKWM have the SOTA performance across all benchmarks. A head-space cost can recover part of the ranking benefit in Reacher and Diverse Maze, but it is weaker on the contact-rich tasks and fails on TwoRoom. Planning in the underlying embedding of the head-trained model is also worse, which means the useful ordering has not been placed where the latent dynamics and the subgoal proposer operate. We therefore apply the ranking loss directly to the world-model embedding. This keeps the prediction model, CEM cost, and generated subgoals in one shared geometry, without adding a test-time metric head.

Table 12: Ranking on the embedding versus on a separate metric head ϕ: success rate (%) of direct planning at the matched read epoch of Table 11. "Cost in embedding" plans with latent distance in the world-model embedding; "cost in ϕ" plans with distance in the head space.

Environment	Diverse Maze			PushT		Reacher		TwoRoom		Cube	
Setting	near	mid	far	50	100	50	100	50	100	25	50
No ranking (LeWM)	9.3	6.0	3.3	40.7	12.0	56.7	52.0	30.7	16.0	18.0	**0.7**
Embedding (ours)	**78.7**	**46.0**	**20.0**	**46.7**	**13.3**	**90.7**	**96.0**	**76.0**	**52.7**	**65.3**	**0.7**
Head, cost in embedding	68.7	30.0	14.0	30.7	6.7	40.0	44.0	21.3	20.7	28.7	**0.7**
Head, cost in ϕ	74.0	45.3	**20.0**	42.0	8.0	88.0	90.7	14.0	24.7	56.0	**0.7**

D.3 BUDGET AND ADVANCEMENT CLOCK

The main protocol follows the step budgets used by the closest prior benchmarks: temporal-offset tasks use LeWM's released rule $B = 2\Delta$, while Diverse Maze uses HWM's 500-step budget for each distance band. These budgets are shared by all compared methods.

Table 13: Success rate (%) under a Δ-independent budget and without the advancement clock. B is the episode budget in environment steps; the protocol of Table 1 uses $B = 100$ at offset 50 and $B = 200$ at offset 100. "No clock" removes the clock floor of Sec. B.3 under the standard budget. Each cell uses 150 evaluation tasks.

		No chain, offset 50		Chain, offset 50			Chain, offset 100	
Environment	Latent	$B = 100$	$B = 200$	$B = 100$	$B = 200$	no clock	$B = 200$	no clock
PushT	Ours	**50.0**	**50.0**	**79.3**	**86.0**	**64.7**	**60.7**	**56.7**
PushT	LeWM	36.0	40.7	68.0	73.3	44.0	58.7	54.0
TwoRoom	Ours	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**99.3**
TwoRoom	LeWM	24.0	29.3	36.7	66.7	46.0	52.0	49.3
Reacher	Ours	**89.3**	**90.7**	**86.0**	94.7	**82.0**	**96.0**	**92.7**
Reacher	LeWM	50.7	64.7	74.0	**96.0**	56.0	80.0	76.0

Table 13 gives two controls on PushT, TwoRoom and Reacher, where B normally changes with the offset. First, offset-50 tasks are also given $B = 200$, matching the offset-100 budget. Second, we remove the clock floor and advance the pointer only by the nearest-slot rule, while keeping the reserved final cycles for the true goal. With a fixed budget, every chained cell is unchanged or higher and the chain keeps its margin over direct planning. Removing the clock hurts some settings, especially PushT and Reacher at offset 50, but the effect is not enough to explain the gains: for example, the clock-free chain still reaches 56.7% on PushT offset 100, compared with 8.7% for direct planning, which indicates that the subgoals remain useful without the clock.

Table 14: Success rate (%) with the deployed pointer check interval $C = 10$ and with the alternative interval $C = P$. All other settings follow Table 1; each cell uses 150 evaluation tasks.

		LeWM latent		Shaped latent	
Environment	Setting	$C = 10$	$C = P$	$C = 10$	$C = P$
PushT	offset 50	68.0	64.0	**79.3**	73.3
PushT	offset 100	58.7	51.3	60.7	**62.0**
TwoRoom	cross-room, off 50	36.7	42.0	**100.0**	99.3
TwoRoom	cross-room, off 100	52.0	71.3	**100.0**	**100.0**
Reacher	offset 50	74.0	62.0	**86.0**	83.3
Reacher	offset 100	80.0	81.3	**96.0**	95.3
Diverse Maze	near, 5–8 cells	18.0	18.7	**80.7**	78.0
Diverse Maze	mid, 9–12 cells	6.0	9.3	**59.3**	58.0
Diverse Maze	far, 13–16 cells	4.7	4.7	**34.7**	**34.7**

D.4 Sensitivity to the pointer check interval

The pointer is reconsidered every C environment steps (Sec. B.3). Our main experiments use $C = 10$. To check that the comparison is not tied to this, Table 14 repeats the chained evaluations with the alternative interval $C = P$, where $P = r\kappa$ is the replanning period.

The shaped latent remains stable under this change. Its success rate changes by at most 2.7 points in every setting except PushT at offset 50, where it drops from 79.3% to 73.3%, and the ordering in Tables 1 and 10 is unchanged. The LeWM latent is more sensitive to the check interval, including a gain on TwoRoom at offset 100, but it still does not close the gap to the shaped latent in the hard long-horizon settings.

D.5 High-level search budget

Table 6 gives HWM's outer CEM the same search budget as the low-level planner: 300 sampled sequences for 30 CEM iterations. This keeps the main comparison compute-matched, but leaves

Table 15: Success rate (%) when only HWM's outer CEM sample count is increased. The 300 column matches Table 6; TREKWM uses adaptive subgoals with the standard 300-sample planner. Each cell uses 150 evaluation tasks.

		HWM, outer CEM samples			
Setting	Latent	300	1000	3000	TREKWM
Cube, offset 50	LeWM	0.0	1.3	2.7	**67.3**
	Shaped	0.7	4.0	1.3	
PushT, offset 100	LeWM	23.3	24.0	33.3	**60.7**
	Shaped	17.3	21.3	27.3	

open a simpler explanation for HWM's weaker long-horizon results: perhaps the high-level planner only needs a larger search. Table 15 tests that explanation on the two settings where HWM is farthest from adaptive subgoals, increasing only the number of outer CEM samples to 1000 and 3000. The low-level planner, high-level checkpoint, horizon, final-goal handoff and scoring rule are unchanged.

The larger search does not change the qualitative picture. Cube remains near zero on both latents even with 3000 outer samples. PushT improves modestly as the outer budget grows, but the best HWM cells, 33.3% on the LeWM latent and 27.3% on the shaped latent, are still well below adaptive subgoals at 60.7% with the standard 300-sample planner. Thus the gap in these settings is not explained by too small a high-level CEM budget.

D.6 CHAINS ON THE TEMPORAL-JEPA LATENT

We also train the same chain proposer on Temporal-JEPA's latent cache while leaving Temporal-JEPA's planning cost unchanged. The result is mixed (Table 16): the chain helps on TwoRoom and PushT, is nearly neutral on Reacher and Diverse Maze, and hurts Cube at offset 25. As with the LeWM-latent chain in Table 10 and HWM in Table 6, the chain is not a latent-agnostic wrapper. It helps when the latent gives the proposer and planner a useful geometry for intermediate targets.

Table 16: Success rate (%) of Temporal-JEPA with and without our chain, next to our model. Temporal-JEPA plans with its own temporal-distance cost in both columns; the chain proposer is trained on its latent.

Setting	Temporal-JEPA direct	Temporal-JEPA + chain	Ours direct	Ours + chain
Cube 25	40.7	13.3	58.7	**87.3**
Cube 50	0.0	4.7	0.7	**67.3**
PushT 50	32.7	42.0	50.0	**79.3**
PushT 100	12.7	29.3	8.7	**60.7**
TwoRoom 50	85.3	97.3	**100.0**	**100.0**
TwoRoom 100	54.7	97.3	86.7	**100.0**
Reacher 50	**90.7**	**90.7**	89.3	86.0
Reacher 100	93.3	94.0	**97.3**	96.0
Diverse Maze near	52.7	50.7	78.7	**80.7**
Diverse Maze mid	28.7	29.3	46.0	**59.3**
Diverse Maze far	14.7	18.7	20.0	**34.7**

D.7 A FIXED-INTERVAL SUBGOAL BASELINE

HIQL-style hierarchies (Park et al., 2023) repeatedly predict a subgoal a fixed number of steps ahead. To isolate this design choice, we train a baseline with the same backbone as our proposer, but with a single regression target: the demonstration frame 25 steps ahead, or the goal if it is nearer. At test time the subgoal is re-predicted at every replanning cycle and pursued by the same CEM planner; the final cycles target the goal as in our chain.

Table 17: Success rate (%) on our latent and the LeWM latent. "Direct" plans to the final goal, while "subgoal" uses an HIQL-style target 25 steps ahead that is re-proposed every replanning cycle. Under our latent, TREKWM is the full method: our shaped latent with our generated subgoal chain. Under the LeWM latent, "our subgoals" applies the same chain proposer, trained on LeWM's latent cache, without latent shaping. The planner, tasks and budgets match Table 1.

	Our latent			LeWM latent		
Setting	direct	subgoal	TREKWM	direct	subgoal	our subgoals
Cube 25	58.7	65.3	**87.3**	26.7	28.0	84.0
Cube 50	0.7	2.0	**67.3**	0.0	0.0	32.0
PushT 50	50.0	**79.3**	**79.3**	36.0	66.7	68.0
PushT 100	8.7	60.0	60.7	11.3	**70.7**	58.7
TwoRoom 50	**100.0**	**100.0**	**100.0**	24.0	34.7	36.7
TwoRoom 100	86.7	**100.0**	**100.0**	16.7	33.3	52.0
Reacher 50	**89.3**	84.0	86.0	50.7	72.0	74.0
Reacher 100	**97.3**	94.7	96.0	53.3	78.0	80.0
Diverse Maze near	78.7	77.3	**80.7**	9.3	18.0	18.0
Diverse Maze mid	46.0	42.7	**59.3**	6.0	4.7	6.0
Diverse Maze far	20.0	28.0	**34.7**	3.3	5.3	4.7

This receding subgoal is strong on tasks where the next local stretch is already informative, such as PushT and Reacher (Table 17). It is much weaker on Cube and on longer navigation settings, where success depends on the whole route: reach–grasp–carry for Cube, or a long sequence of turns in Maze and TwoRoom. In these settings, predicting the full chain gives the planner intermediate targets that are organized by the endpoint rather than only by the next fixed interval.

Table 18: Open-loop prediction error on held-out episodes, driven by the demonstration's actions for h predictor steps of five environment steps each. "rel." is the latent error divided by the true displacement from the current frame; "centred" subtracts the mean prediction offset estimated on training windows; "probe" is the median error of a linear position probe on the predicted latent at $h = 10$, with the probe's error on the true latent in parentheses; "top-1" is the fraction of windows whose prediction is nearest to the correct frame among the window's frames. Diverse Maze uses held-out episodes on the training layouts, since the test layouts have no demonstrations.

Environment	Model	one-step rel.	rel. h=1/5/10	centred rel. h=1/5/10	probe h=10 (floor)	top-1 h=1/10
Cube	LeWM	0.12	0.27 / 0.10 / 0.10	0.25 / 0.10 / 0.10	2.84 (2.69) cm	**1.00 / 1.00**
Cube	Ours	**0.08**	**0.08 / 0.05 / 0.06**	**0.06 / 0.05 / 0.05**	**2.53 (2.47)** cm	**1.00 / 1.00**
PushT	LeWM	0.18	0.18 / 0.12 / 0.20	0.18 / 0.12 / 0.20	**14.28 (11.94)** px	0.98 / 0.79
PushT	Ours	**0.11**	**0.10 / 0.09 / 0.11**	**0.09 / 0.09 / 0.11**	16.72 (15.49) px	**0.99 / 0.89**
TwoRoom	LeWM	**0.03**	**0.03 / 0.05 / 0.09**	**0.03 / 0.05 / 0.09**	**3.11** (2.81) px	**1.00 / 0.96**
TwoRoom	Ours	1.53	1.50 / 0.53 / 0.30	0.53 / 0.16 / 0.11	8.34 (**2.05**) px	0.90 / 0.73
Reacher	LeWM	**0.05**	**0.05 / 0.05 / 0.06**	**0.04 / 0.05 / 0.06**	**0.63 (0.53)** cm	**0.98 / 0.94**
Reacher	Ours	0.11	0.11 / 0.11 / 0.11	0.11 / 0.10 / 0.11	3.04 (2.95) cm	**0.98** / 0.93
Diverse Maze	LeWM	1.07	1.08 / 1.32 / 1.43	1.05 / 1.18 / 1.29	3.09 (3.04) cells	0.13 / 0.10
Diverse Maze	Ours	**0.18**	**0.17 / 0.11 / 0.09**	**0.17 / 0.11 / 0.09**	**0.23 (0.14)** cells	**1.00 / 1.00**

D.8 PREDICTION QUALITY OF THE SHAPED MODEL

Finally, we check that shaping the latent has not simply traded planning performance for unusable dynamics. We evaluate open-loop prediction on held-out episodes by rolling the predictor for ten model steps under the demonstration's actions and comparing each prediction with the latent actually reached (Table 18).

The shaped model predicts at least as accurately as LeWM on Cube, PushT and Diverse Maze. Reacher is the clearest case where LeWM predicts more precisely, yet the shaped model plans better; for planning, the relevant quantity is the ordering of candidate endpoints, not raw metric calibration. In TwoRoom, the shaped model has a nearly constant prediction bias. After subtracting this mean bias, its relative error is low, and its predictions are usually closest to the correct frame

within the rollout window. The main results are therefore not a consequence of sacrificing world-model dynamics.

Table 19: Rank correlation between latent distance to the goal and true time-to-go. ρ and τ are averaged over goal frames; the binned columns give Spearman's ρ within each time-to-go range.

				Spearman ρ by true time-to-go				
Environment	Latent	ρ	τ	bin 1	bin 2	bin 3	bin 4	bin 5
Cube	LeWM	+0.445	+0.318	+0.68	+0.37	+0.14	-0.02	-0.04
	TREKWM	**+0.997**	**+0.988**	**+0.85**	**+0.65**	**+0.50**	**+0.41**	**+0.43**
PushT	LeWM	+0.696	+0.559	+0.69	+0.33	+0.04	-0.01	-0.06
	TREKWM	**+0.980**	**+0.945**	**+0.73**	**+0.38**	**+0.21**	**+0.15**	**+0.15**
TwoRoom	LeWM	+0.460	+0.332	+0.37	+0.10	+0.09	+0.03	
	TREKWM	**+0.926**	**+0.822**	**+0.64**	**+0.32**	**+0.22**	**+0.18**	
Reacher	LeWM	+0.617	+0.494	**+0.75**	+0.19	+0.03	+0.00	-0.05
	TREKWM	**+0.928**	**+0.841**	+0.69	**+0.30**	**+0.19**	**+0.15**	**+0.04**
Diverse Maze	LeWM	+0.204	+0.147	+0.26	+0.03	+0.06	+0.01	
	TREKWM	**+0.969**	**+0.919**	**+0.90**	**+0.69**	**+0.39**	**+0.10**	

Time-to-go bins in environment steps: Cube and Reacher 0–10 / 10–20 / 20–30 / 30–40 / 40–60; PushT 0–25 / 25–50 / 50–75 / 75–100 / 100–150; TwoRoom 0–25 / 25–50 / 50–75 / 75–100; Diverse Maze 0–20 / 20–40 / 40–60 / 60–80.

D.9 RANK CORRELATION BETWEEN LATENT DISTANCE AND TIME-TO-GO

The planner uses Euclidean distance in latent space as its cost, so a useful latent should rank states by progress toward the goal. Table 19 tests this directly by correlating latent distance to a future goal with the true number of demonstration steps remaining to that goal. A positive correlation means that states farther from the goal in time also tend to be farther away in latent space.

For each held-out demonstration, we choose three goal frames from the final 30% of the episode and compare that goal to every earlier frame. We use the 5% held-out episodes for all environments except Diverse Maze, where the evaluation layouts have no demonstrations, so we use expert episodes from the training layouts. Reacher episodes are split into single-target segments before goals are drawn. Both LeWM and our shaped latent are evaluated on the same frames and goal frames. The first two numeric columns report Spearman's ρ and Kendall's τ averaged over goals; the binned columns report Spearman's ρ within different time-to-go ranges.

The shaped latent has substantially higher overall correlation in every environment. The binned view shows the same pattern locally: LeWM's ordering often collapses once the goal is no longer nearby, while the shaped latent keeps a positive rank signal in every reported bin.