HORIZON-ALIGNED SCORING: A LONGER VIEW FOR SHORT-HORIZON WORLD MODEL PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

LeWorldModel (LeWM) learns visual latent dynamics from reward-free trajectories and uses the cross-entropy method (CEM) to plan toward image goals. It ranks candidate action sequences by their predicted endpoint-to-goal distances, so short rollouts can undervalue preparatory actions whose benefits emerge later. We introduce Horizon-Aligned Scoring (HAS), which uses an action-free latent predictor to extend each endpoint toward the goal's time scale before scoring. Trained on LeWM's existing trajectories, the predictor adapts its recursion depth to the remaining temporal gap, given the goal offset. HAS thereby extends candidate evaluation without lengthening the action-conditioned search. Across PushT, TwoRoom, and Reacher, HAS improves mean success when the goal image is sampled beyond LeWM's default planning span of 25 environment steps. For goal images sampled 50 environment steps after the initial observation, PushT success rises from 42.8% to 67.8%; at a 100-step goal offset in TwoRoom, HAS reaches 70.4%, compared with 55.4% for the matched hybrid variant of Trajectory Reachability Metrics (TRM-Hybrid) and 16.4% for LeWM. The gains persist across three independently trained checkpoints per task. HAS also outperforms a LeWM control with twice the action horizon at every tested longer goal offset under the same replanning interval. Together, the results show that HAS improves LeWM's long-range planning without additional training data or a longer action-conditioned planning horizon.

1 INTRODUCTION

Planning toward a distant visual goal requires evaluating actions whose benefits may emerge after the current rollout ends. LeWorldModel (LeWM) (Maes et al., 2026) jointly learns an encoder and an action-conditioned latent predictor from reward-free offline trajectories. At test time, LeWM uses the cross-entropy method (CEM) (Rubinstein, 1999) to plan toward image goals: sampled action sequences are rolled out through the latent predictor and ranked by the distance between their predicted endpoints and the goal embedding. This simple interface supports goals specified after training and makes LeWM an effective local planner. However, it also ties the temporal reach of candidate evaluation to the action-conditioned rollout span.

When the goal lies beyond this span, a predicted endpoint represents only an intermediate stage of the task. An action can therefore enable useful later progress while leaving its immediate endpoint far from the goal. Moving toward a passage, for example, can be necessary for reaching another room even though the resulting short-rollout endpoint remains distant from the final goal. Direct endpoint-to-goal distance can consequently undervalue actions whose benefits emerge only through subsequent execution. This temporal mismatch motivates a scoring rule that accounts for progress beyond the action-conditioned rollout.

Horizon-Aligned Scoring (HAS) changes the representation used to score CEM candidates (Figure 1). The action-conditioned predictor first produces each candidate's fixed-span endpoint. An action-free latent predictor, learned from local latent transitions in the same offline trajectories, then recursively advances that endpoint before it is compared with the goal. One-step and recursive supervision from those trajectories support repeated application; no additional data collection or supervision at the evaluated goal distances is required. Our planning query includes the goal image and its temporal offset in a reference trajectory. This offset supplies the time scale for a recursion

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.

schedule that decreases as execution advances. HAS thereby extends the temporal reach of candidate scoring while leaving the CEM sampling and update rules, as well as the candidate action horizon, unchanged.

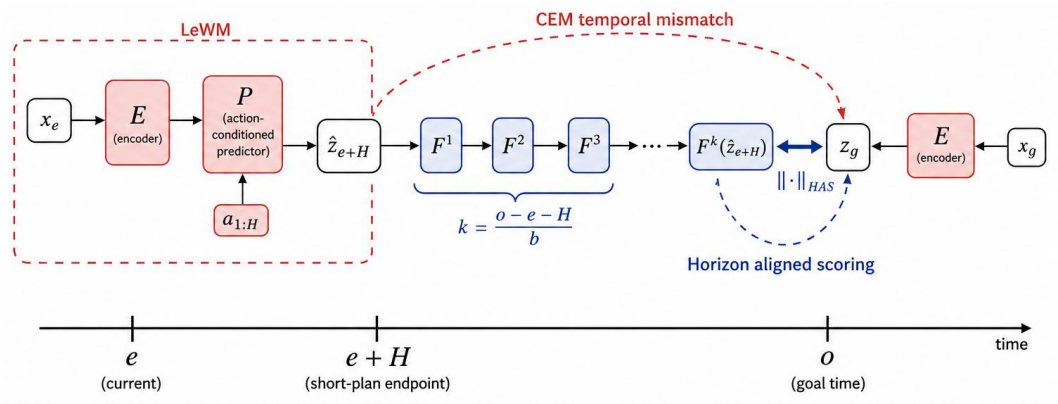

Figure 1: **Horizon-aligned scoring within LeWM's CEM planner.** The encoder E and action-conditioned predictor P produce $\hat{z}_{e+H}$ from h action blocks ($H = hb$ environment steps). HAS applies the shared action-free predictor F before comparing with z_g, bridging the nominal gap to a later goal (red dashed arc). Numbered F boxes denote successive calls; F^k denotes their composition. At $k = 0$, HAS recovers LeWM's endpoint score.

Our contributions are threefold. First, HAS extends the representation used for terminal scoring while preserving LeWM's action-conditioned planning span. Second, local and recursive supervision from the existing training trajectories produces an action-free predictor whose evaluation depth adapts during execution. Third, matched experiments across three visual-control tasks demonstrate long-offset gains, supported by longer-horizon LeWM controls, a TwoRoom TRM comparison, depth and offset sensitivity analyses, and independently trained checkpoints.

2 Related Work

Learned latent dynamics can support planning toward image goals specified after training. PLDM learns these dynamics from offline trajectories (Sobal et al., 2025), DINO-WM predicts pretrained visual features (Zhou et al., 2025), and LeWM jointly trains an encoder and action-conditioned predictor (Maes et al., 2026). HAS builds on LeWM's endpoint-based CEM planner.

Longer-range planning can extend predictions or guide action proposals. Variable-length world models predict over different action durations (Du et al., 2026), and hierarchical models use coarse predictions to guide local planning (Zhang et al., 2026). FF-JEPA proposes subgoals through action-free latent prediction (Masip et al., 2026), while SAGE conditions action generation on predicted subgoals (Cheng et al., 2026). HAS uses action-free prediction after the short action-conditioned rollout to rescore CEM candidates, without lengthening their action span or generating subgoals.

Other work changes how predicted futures are compared with goals. Temporal straightening regularizes trajectory curvature to shape latent distance (Wang et al., 2026), and RC-aux adds multi-horizon prediction and budget-conditioned reachability correction (Li et al., 2026b). Traj-LeWM adds a goal-conditioned trajectory cost to endpoint distance (Huang et al., 2026), while TRM learns a temporal pairwise cost that can replace or augment latent endpoint distance for candidate ranking (Li et al., 2026a). HAS also targets candidate evaluation. It applies the same squared Euclidean distance to a temporally extended endpoint representation, with extension depth set by the remaining goal gap. The distance operator and CEM update procedure are retained; the representation entering that distance changes.

3 PROBLEM SETUP

Let x_t be an image observation at environment step t, with embedding $z_t = E(x_t) \in \mathbb{R}^d$. An action block contains b consecutive environment actions. The predictor P uses recent latent observations and action blocks to estimate the next latent, and a candidate sequence of h blocks covers a planning span $H = hb$. We use $b = 5$ and default $h = 5$, giving $H = 25$ environment steps. The replanning interval R is the number of environment steps executed before the next planning call; the default is $R = 25$. Each latent prediction step spans one action block.

A reference trajectory supplies the initial state and a goal image x_g sampled o environment steps later, with $z_g = E(x_g)$. The planning query provides both x_g and o. This sampling offset is a temporal-scale cue, not a shortest-path time or an execution deadline; evaluation permits up to $2o$ interaction steps. After e steps of online execution, the current observation x_e may differ from the reference trajectory. The planner evaluates candidate action-block sequences $a^{(n)}_{0:h-1}$. Their predicted endpoints are

$$\hat{z}^{(n)}_{e+H} = P^{(h)}(z_e, a^{(n)}_{0:h-1}), \tag{1}$$

where $P^{(h)}$ denotes autoregressive prediction with the recent latent and action context suppressed for clarity. LeWM assigns the terminal cost

$$C^{(n)}_{\mathrm{LeWM}} = \left\| \hat{z}^{(n)}_{e+H} - z_g \right\|_2^2. \tag{2}$$

When $o - e > H$, HAS uses the nominal gap $o - e - H$ to extend the candidate's representation for scoring. The action-conditioned rollout still ends at $e + H$; the extension supplies a later-time scoring view without specifying further actions or assuming that the online trajectory will reach the goal at step o.

4 HORIZON-ALIGNED SCORING

4.1 SCORING BEYOND THE PLANNING SPAN

HAS learns an action-free latent predictor $F : \mathbb{R}^d \to \mathbb{R}^d$, where one application of F represents one action block of latent evolution. At elapsed time e, it selects depth

$$k(e, o) = \max\left(\frac{o - e - H}{b}, 0 \right). \tag{3}$$

The evaluated offsets and replanning times make every positive gap divisible by b. With $F^0(z) = z$ and $F^{j+1}(z) = F(F^j(z))$, the HAS cost is

$$C^{(n)}_{\mathrm{HAS}} = \left\| F^{k(e,o)}\left(\hat{z}^{(n)}_{e+H} \right) - z_g \right\|_2^2. \tag{4}$$

Thus, each candidate is evaluated through a temporally extended representation of its short-rollout endpoint.

The action-conditioned rollout predicts consequences of the candidate actions; F supplies a continuation learned from local offline transitions. With a single latent as input, F conditions on neither the goal nor further actions. Its output is a scoring representation, and subsequent actions are selected by replanning from fresh observations.

The remaining-gap rule adapts this view as execution advances. For $o = 100$ and $H = 25$, replans at $e = 0, 25, 50, 75$ use $k = 15, 10, 5, 0$, respectively. Once $e + H \geq o$, the extension is the identity and Equation 4 becomes the LeWM terminal cost. HAS therefore joins extended scoring for distant goals with direct endpoint scoring near the goal horizon.

4.2 LEARNING THE ACTION-FREE LATENT PREDICTOR

The predictor F learns from local transitions already available to the world model (Figure 2). Each training sample contains four encoded observations $z_{0:3} = E(x_{0:3})$ separated by one action block; here the subscripts index blocks within the sample. LeWM produces $p_{0:2}$, with p_i predicting z_{i+1}.

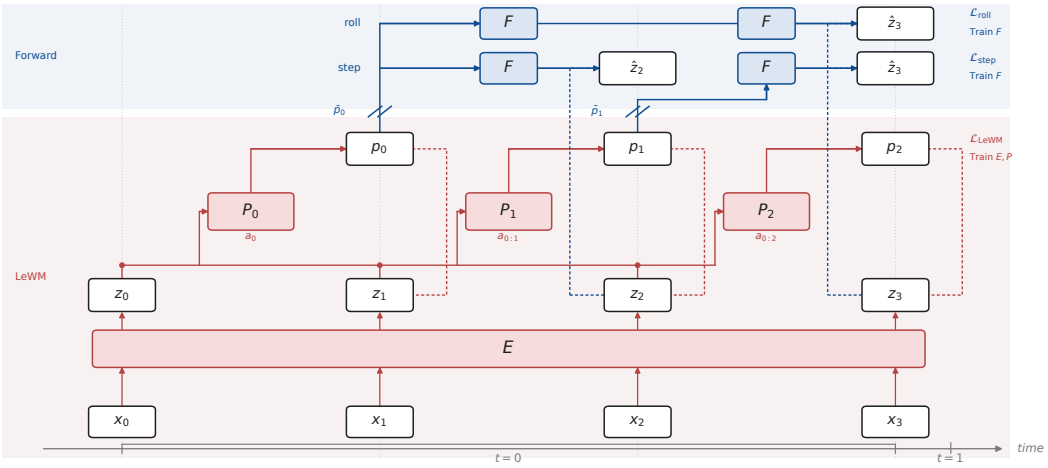

Figure 2: Local and recursive supervision for the action-free latent predictor F. LeWM predicts $p_i \approx z_{i+1}$ from the four-observation training window. The shared F consumes detached predictor outputs and is supervised by detached encoder targets through $F(p_0) \to z_2$, $F(p_1) \to z_3$, and $F(F(p_0)) \to z_3$. The schematic P_i instances share the action-conditioned predictor parameters; each observation interval is one five-step action block.

Define the mean squared latent error as $D(z, z') = \frac{1}{d}\|z - z'\|_2^2$. The world model objective is

$$\mathcal{L}_{\text{LeWM}} = \frac{1}{3}\sum_{i=0}^{2} D(p_i, z_{i+1}) + 0.09\,\mathcal{L}_{\text{SIGReg}}(z_{0:3}), \tag{5}$$

with batch averaging implicit. SIGReg regularizes the embedding distribution (Maes et al., 2026), and both predicted and target embeddings receive gradients from the prediction term.

To match the inference interface, F is trained on outputs of P. Since p_i predicts block $i+1$, its one-block continuation targets z_{i+2}. The four-observation window provides such targets for p_0 and p_1. With bars denoting stop-gradient copies, we define

$$\mathcal{L}_{\text{step}} = \frac{1}{2}\sum_{i=0}^{1} D(F(\bar{p}_i), \bar{z}_{i+2})\,. \tag{6}$$

HAS applies F recursively during planning, so we additionally supervise a two-block extension from p_0:

$$\mathcal{L}_{\text{roll}} = D(F(F(\bar{p}_0)), \bar{z}_3). \tag{7}$$

The complete action-free predictor objective is

$$\mathcal{L}_F = \mathcal{L}_{\text{step}} + \mathcal{L}_{\text{roll}}. \tag{8}$$

The one-step term trains local continuation from LeWM-predicted endpoints, while the recursive term exposes F to its own output and constrains the two-block composition used during planning. Gradients pass through both applications of F, but the detached inputs and targets prevent $\mathcal{L}_F$ from directly updating P or E. Both terms use the same four-observation training window as LeWM, requiring no additional trajectories or targets at the evaluated goal offsets.

The $\mathcal{L}_F$ objective has unit weight in training. We implement F as a small residual MLP, with architectural and optimization details in Appendix A. Training supervises one- and two-block extensions. Evaluation tests longer compositions of this shared map, up to $k = 15$ in the nominal schedule and $k = 19$ in the offset-perturbation study.

4.3 Receding-horizon integration

HAS replaces the candidate cost while retaining CEM's sampling and elite-refitting procedure. Algorithm 1 describes one planning call. The controller executes the first R environment steps of the resulting mean sequence, then replans from the latest observation.

Algorithm 1 Horizon-Aligned CEM at elapsed time e

Require: current latent z_e and recent context; elapsed steps e
Require: goal z_g; reference-trajectory offset o
Require: predictors P (action-conditioned) and F (action-free)
Require: block size b; span $H = hb$
Require: candidates N; elites M; updates J

$k \leftarrow \max((o - e - H)/b, 0)$
$(\mu_0, \sigma_0) \leftarrow \text{INITIALIZECEM}(h)$
for $j = 1, \ldots, J$ **do**
  $A^{(1:N)} \leftarrow \text{SAMPLE}(\mu_{j-1}, \sigma_{j-1}, N)$
  for all $n \in \{1, \ldots, N\}$ **do**
    $\hat{z}^{(n)}_{e+H} \leftarrow P^{(h)}(z_e, A^{(n)})$
    $\tilde{z}^{(n)} \leftarrow F^k(\hat{z}^{(n)}_{e+H})$
    $c_n \leftarrow \|\tilde{z}^{(n)} - z_g\|_2^2$
  end for
  $\mathcal{I}_j \leftarrow \text{ELITE}(c_{1:N}, M)$
  $(\mu_j, \sigma_j) \leftarrow \text{MOMENTS}(\{A^{(n)} : n \in \mathcal{I}_j\})$
end for
return μ_J

Within each CEM update, F is applied k times to the batch of candidate endpoints. These batched residual-MLP stages decrease in number as the gap closes, while the action sequence remains h blocks long. Appendix E measures the resulting closed-loop latency.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We test long-offset goal reaching, alternative planning and scoring strategies, depth sensitivity, and consistency across trained models. PushT (Florence et al., 2022), TwoRoom (Sobal et al., 2025), and Reacher (Maes et al., 2026) provide visual control settings involving object manipulation, navigation, and reaching. For the primary paired comparison, we use one epoch-10 checkpoint per task and compare the LeWM terminal score with HAS on the same world model. Ten evaluation groups, indexed by seeds 42–51, contain 50 episodes each. Methods share the sampled starts and goals within each task, offset, and group. Reported means and sample standard deviations summarize these ten group-level success rates and measure evaluation variability for the trained model.

Goal offsets are $o \in \{25, 50, 75, 100\}$ environment steps, with an interaction budget of $2o$ and the success criterion defined by each environment. Default LeWM and HAS use $H = R = 25$, with five actions per block. To compare the scoring extension with a longer action-conditioned plan, we evaluate LeWM with $h = 10$, giving $H = 50$. One control retains the default replanning interval $R = 25$; a second executes the full plan with $R = 50$. All configurations use 300 CEM candidates, 30 elites, and 30 updates. The $H = 50$ controls' $o = 25$ condition extends beyond the sampled goal time. Appendices A and B give training and evaluation details.

5.2 LONG-OFFSET GOAL REACHING AND ACTION-HORIZON CONTROLS

HAS improves mean success across all three tasks at every evaluated offset beyond the default planning span (Table 1, Figure 3). The improvement is particularly pronounced in TwoRoom, where success at $o = 100$ rises from 16.4% to 70.4%. PushT at $o = 50$ improves from 42.8% to 67.8%. These gains show that a temporal extension of candidate endpoints can make the same short action-conditioned planner more effective at reaching distant visual goals.

With the replanning interval fixed at $R = 25$, LeWM with $H = 50$ remains below HAS in all nine task–offset conditions beyond the default planning span. For example, TwoRoom at $o = 100$ reaches 24.4% under this same-cadence control, compared with 70.4% under HAS. Under the standard CEM candidate and update budget, the tested doubling of the action-conditioned search span does not reproduce the gains from extending the scored representation. The $H = 50, R = 50$ control

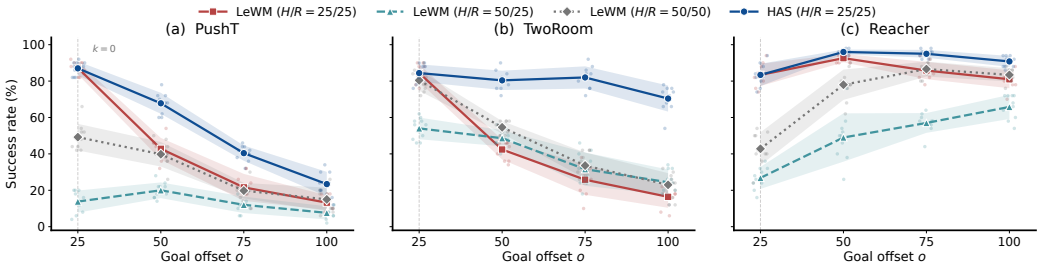

Figure 3: Goal reaching across temporal offsets. Points are success rates for evaluation groups of 50 episodes; lines and shaded bands give the mean and sample standard deviation over ten groups using one checkpoint per task. Configurations are identified by their action-conditioned rollout span H and replanning interval R. The dashed vertical line marks the default planning span $H = 25$, where HAS uses $k = 0$.

Table 1: Success rate (%; mean ± sample SD over 10 evaluation groups). H/R gives the action-conditioned rollout span and replanning interval in environment steps. Bold marks the highest mean for each task and offset. †The o=25 column is a non-primary overshoot condition for $H = 50$.

Task	Method (H/R)	o=25	o=50	o=75	o=100
PushT	LeWM (25/25)	**87.2±3.6**	42.8±6.0	21.6±7.4	13.2±4.0
	LeWM (50/25)	13.8±6.1†	20.0±4.3	12.0±4.5	7.6±3.6
	LeWM (50/50)	49.2±7.3†	39.8±6.8	19.8±4.3	15.0±6.4
	HAS (25/25)	87.0±3.8	**67.8±5.8**	**40.4±4.2**	**23.4±6.7**
TwoRoom	LeWM (25/25)	**84.4±4.8**	42.4±5.5	25.8±8.3	16.4±6.0
	LeWM (50/25)	54.0±6.0†	48.6±4.4	31.6±9.1	24.4±7.0
	LeWM (50/50)	80.4±5.7†	54.6±3.4	33.6±7.2	23.0±6.4
	HAS (25/25)	**84.4±4.8**	**80.4±5.3**	**82.0±6.1**	**70.4±7.2**
Reacher	LeWM (25/25)	**83.4±6.1**	92.6±2.3	85.8±4.8	81.0±4.9
	LeWM (50/25)	26.8±6.1†	49.0±13.3	57.0±5.4	65.8±6.6
	LeWM (50/50)	42.8±9.1†	78.0±8.3	86.6±4.1	83.4±3.7
	HAS (25/25)	**83.4±6.1**	**96.0±1.3**	**95.0±2.2**	**90.8±3.0**

also remains below HAS in all nine longer-offset conditions. HAS thus improves long-offset control beyond both tested ways of using a longer action-conditioned plan. In the single-environment timing study at $o = 100$, mean task-wise replanning costs are $1.04\times$ LeWM for HAS and $2.12\times$ for the $H = 50, R = 25$ control (Appendix E). We next compare HAS with a learned terminal-scoring alternative.

5.3 Comparison with trajectory reachability scoring

We compare HAS with Trajectory Reachability Metrics (TRM) (Li et al., 2026a), which learns a temporal pairwise cost for terminal candidate ranking. Under the shared TwoRoom protocol, both TRM variants reuse the frozen LeWM checkpoint, start manifests, CEM budget, and $H = R = 25$; their temporal head is trained on evaluation-disjoint pairs from the same trajectories. TRM-Replace uses the learned cost alone, whereas TRM-Hybrid combines candidate-standardized latent and TRM costs with $\lambda = 1$, fixed before evaluation. Appendix D gives the complete reimplementation details.

HAS exceeds TRM-Hybrid by 13.0 and 15.0 percentage points at $o = 75$ and $o = 100$, respectively, while the methods are close at $o = 50$ (80.4% versus 81.0%). At $o = 25$, HAS uses $k = 0$ and matches LeWM; TRM-Hybrid improves the short-offset score. The advantage at the two longest offsets shows that HAS remains effective against a competitive learned scoring baseline as the temporal separation between start and goal increases.

Table 2: TwoRoom success rate (%; mean ± sample standard deviation over ten evaluation groups of 50 episodes) under the shared $H = R = 25$ protocol. TRM-Replace and TRM-Hybrid are matched reimplementations using one frozen temporal head. Bold marks the highest mean at each offset.

Method	o=25	o=50	o=75	o=100
LeWM	84.4±4.8	42.4±5.5	25.8±8.3	16.4±6.0
TRM-Replace	69.6±6.2	45.6±6.9	31.2±2.5	14.4±5.8
TRM-Hybrid (λ=1)	**95.4±3.4**	**81.0±5.7**	69.0±4.6	55.4±4.7
HAS	84.4±4.8	80.4±5.3	**82.0±6.1**	**70.4±7.2**

5.4 The role of alignment depth

The usefulness of recursive latent extension depends on its depth. We compare the remaining-gap rule with fixed depths $k \in \{0, 5, 10, 15\}$ on the same ten evaluation groups at $o \in \{75, 100\}$ (Table 3). A fixed depth is applied at every replan, including late in execution when the dynamic rule has reached zero. This comparison tests the full depth schedule: the initial extension and its reduction as the controller approaches the goal horizon.

Table 3: Success rate (%; mean ± sample SD over ten evaluation groups) under different recursion-depth schedules. HAS uses the remaining-gap rule; fixed k is reapplied at every replan. The world model checkpoint, starts, and CEM configuration are shared within each task and evaluation group. Bold marks the highest mean in each row.

Task	o	k=0	k=5	k=10	k=15	HAS
PushT	75	21.6±7.4	4.0±2.7	0.8±1.0	0.4±0.8	**40.4±4.2**
PushT	100	13.2±4.0	2.6±1.9	0.4±0.8	0.6±1.0	**23.4±6.7**
TwoRoom	75	25.8±8.3	66.6±3.1	29.0±6.1	13.6±3.7	**82.0±6.1**
TwoRoom	100	16.4±6.0	**76.2±5.7**	31.6±5.5	12.8±2.1	70.4±7.2
Reacher	75	85.8±4.8	83.6±4.5	67.6±5.1	45.6±6.4	**95.0±2.2**
Reacher	100	81.0±4.9	80.6±5.5	71.2±6.7	47.4±7.4	**90.8±3.0**

Dynamic alignment achieves the highest mean in five of the six task–offset conditions, making it a strong default across the tested settings. Fixed deep recursion is particularly unfavorable in PushT: at $o = 75$, HAS reaches 40.4%, while fixed depths of 5 or more remain at 4.0% or below. The comparison supports reducing extension depth during execution, so that later decisions return to direct endpoint scoring as the nominal temporal gap closes.

Sensitivity to the supplied goal offset. We scale only the offset used by the recursion schedule to $0.8o$ or $1.2o$, keeping the goal image and interaction budget unchanged. All 18 perturbed task–offset–scale means remain above matched LeWM. Relative to the nominal schedule, PushT success decreases by at most 9.6 percentage points, Reacher changes by at most 1.6 points, and TwoRoom improves under both perturbations. These results support tolerance to the tested offset errors and reveal task-dependent depth preferences (Appendix F).

5.5 Consistency across matched training runs

We repeat the paired LeWM–HAS comparison using three independently trained epoch-10 checkpoints per task (seeds 3072–3074). Within each training seed, both scoring rules share the encoder, action-conditioned predictor, $H = 25$ CEM configuration, and the same ten 50-episode start-and-goal groups. We first average success over the ten groups for each checkpoint. Table 4 reports the mean and sample standard deviation across the three checkpoint means, together with the paired gain for each training seed.

All 27 paired training-run comparisons beyond $H = 25$ are positive (2.8–64.8 percentage points), showing consistent HAS gains across tasks, offsets, and independent checkpoints.

Table 4: Matched three-seed results. LeWM/HAS: success rate (%; mean ± SD); Δ: paired gain (percentage points). Each checkpoint averages ten 50-episode groups.

Task	o	LeWM	HAS	Δ_{3072}	Δ_{3073}	Δ_{3074}
PushT	50	38.9 ± 1.7	68.5 ± 2.7	$+31.0$	$+29.4$	$+28.2$
	75	20.8 ± 0.7	43.1 ± 2.8	$+18.4$	$+24.8$	$+23.8$
	100	15.1 ± 1.0	25.9 ± 3.8	$+9.0$	$+14.8$	$+8.8$
TwoRoom	50	37.5 ± 1.5	79.9 ± 1.2	$+42.4$	$+42.0$	$+42.8$
	75	23.9 ± 1.2	81.7 ± 3.9	$+59.0$	$+53.0$	$+61.2$
	100	15.3 ± 0.6	72.3 ± 8.7	$+59.4$	$+47.0$	$+64.8$
Reacher	50	93.0 ± 0.6	96.9 ± 1.0	$+4.0$	$+4.8$	$+2.8$
	75	87.5 ± 0.7	95.7 ± 0.4	$+8.0$	$+7.8$	$+8.8$
	100	82.0 ± 0.8	92.3 ± 1.9	$+11.4$	$+7.4$	$+12.2$

5.6 How temporal extension changes scoring

On 400 evaluation-disjoint windows per task, we compare the observed endpoint-to-goal distance $\|z_H - z_g\|_2$ with its recursively extended counterpart $\|F^k(z_H) - z_g\|_2$ (Figure 4).

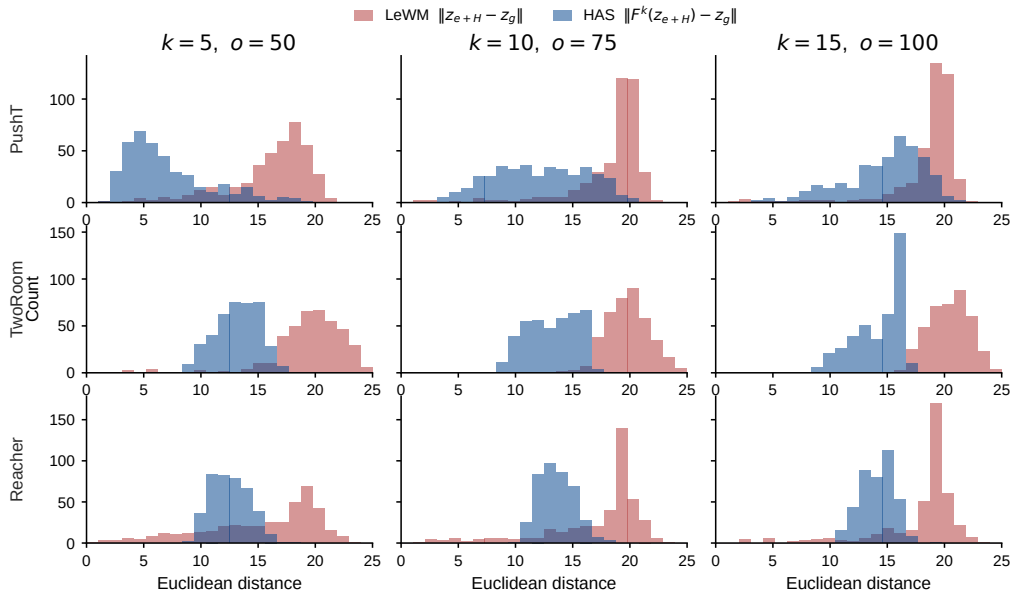

Figure 4: Observed endpoint-to-goal distances before and after temporal extension for 400 evaluation-disjoint windows per task ($H = 25$). Axes show Euclidean distance and window count.

Temporal extension generally shifts these distances toward smaller values, with broader distributions in some conditions (Figure 4). This diagnostic visualizes how F^k changes the endpoint representation before comparison with a later goal. It uses encoded trajectory states, not sampled CEM candidates, and does not by itself measure ranking accuracy. The matched control gains and depth ablation provide the behavioral evidence that this scoring construction benefits planning. Appendix C details the sampling protocol.

5.7 Planning behavior

The paired executions in Figure 5 show how useful actions may first prepare the state before producing visible goal progress. In PushT, LeWM contacts the T-shaped object within the first action block and rotates it past the target orientation by about step 30; subsequent motion does not recover the goal pose. HAS instead repositions the pusher around the object through roughly step 35, with

little change in the object's pose, then aligns the object with the goal between steps 35 and 55. In TwoRoom, LeWM moves downward along the starting side and remains near the lower boundary, while HAS follows a preparatory route and approaches the goal between steps 25 and 60. These selected executions illustrate preparatory movements followed by goal-directed progress, consistent with evaluating intermediate states through their continuation. HAS supports this behavior while retaining the short action-conditioned planning span.

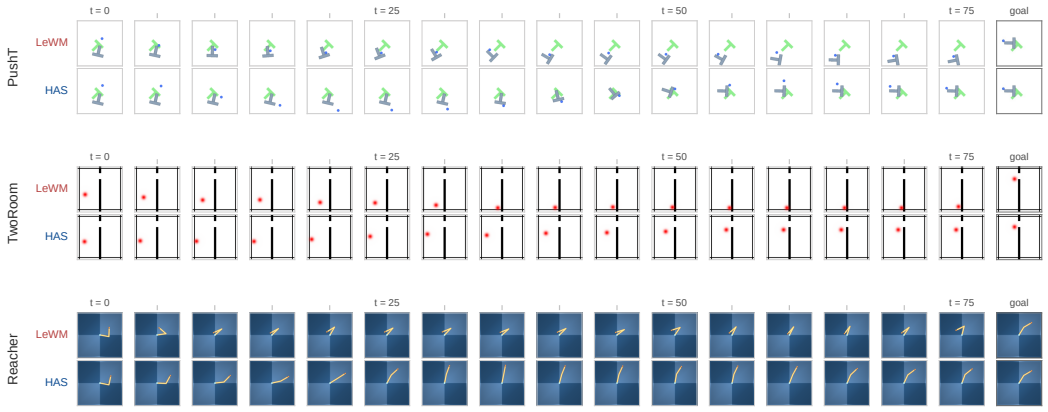

Figure 5: Paired executions from one shared start per task at goal offset $o = 75$, using LeWM and HAS with $H = 25$. Frames are spaced by one action block (5 environment steps); the last column is the visual goal. HAS succeeds on these selected starts, while LeWM fails under the environment success criterion.

6 Limitations

The current formulation treats the temporal offset of the goal image as part of the planning query because it determines the recursion schedule. Tasks that supply only an image goal would therefore pair HAS with a temporal-scale estimator or evaluate multiple candidate depths. The $\pm 20\%$ schedule-offset perturbations in Appendix F preserve the mean advantage over LeWM in every tested condition; obtaining the temporal scale from the goal image remains an extension. In addition, the deterministic predictor represents a single continuation. Longer compositions extrapolate beyond its local training targets and do not explicitly represent uncertainty over alternative futures. When the goal lies within the action-conditioned planning span, the adaptive rule selects $k = 0$ and preserves LeWM's endpoint score. HAS therefore specializes in correcting temporal mismatch beyond the action horizon; the short-offset gains of TRM-Hybrid show that learned metric correction addresses a complementary scoring dimension. Combining these two dimensions is a natural extension. Finally, PushT, TwoRoom, and Reacher cover manipulation, navigation, and reaching, but their action spaces are low-dimensional. Evaluation on higher-dimensional visuomotor control, together with broader matched comparisons to long-horizon planning and scoring methods, would clarify the generality of scoring-time alignment.

7 Conclusion

Horizon-Aligned Scoring extends short-rollout endpoints before evaluating their distance to a later goal. Its action-free predictor learns from LeWM's existing training trajectories, and its extension depth adapts as execution reduces the remaining temporal gap. The observed long-offset gains, including improvements over the tested longer-horizon LeWM controls and over TRM-Hybrid at the longest TwoRoom offsets, support this approach to evaluating intermediate progress. Depth ablations favor adapting the extension during execution, while paired trajectories illustrate preparatory movements followed by successful goal-directed progress. HAS thus offers a way to account for the value of intermediate states that direct endpoint-to-goal proximity can underestimate.

AI USE STATEMENT

We used generative AI tools to assist with software implementation, literature discovery and checking, manuscript organization, interpretation of experimental results, drafting, and language editing. AI-assisted code was reviewed and tested by the authors, references identified or checked with AI assistance were verified against their original sources, and all reported numerical results were checked against the underlying experiment outputs. Generative AI was not used to generate synthetic datasets or to formulate or prove mathematical claims. The authors take full responsibility for the final manuscript, including its text, code, results, citations, and claims.

ETHICS STATEMENT

This work evaluates visual goal-conditioned control in simulated manipulation, navigation, and reaching tasks. The experiments involve simulated observations and actions, with no human-subject or personal-data collection. Deployment on physical systems would require separate validation of control safety and failure recovery.

REPRODUCIBILITY STATEMENT

Sections 3–4 specify the planning interface, objectives, gradient boundaries, and recursion rule. Appendices A–D document implementation, paired evaluation, the scoring diagnostic, and the TRM reimplementation. Appendices E–F report latency and offset-perturbation protocols and complete results. Table captions identify the statistical units used for aggregation.

REFERENCES

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

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.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A. Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 158–168, 2022.

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*, 2026.

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

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.

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.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999. doi: 10.1023/A:1010091220143.

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1465.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim G. J. Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning. In *International Conference on Machine Learning*, 2026.

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.

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

A TRAINING AND IMPLEMENTATION

Each training window contains four observations sampled every five environment steps. Predictor outputs p_0, p_1, p_2 align with encoder latents z_1, z_2, z_3, respectively. The action-free predictor F uses p_0 and p_1 as one-step inputs with targets z_2 and z_3; its recursive example maps p_0 to z_3 through two applications. Figure 2 summarizes this supervision. All instances of F share parameters, and the training losses average squared errors across latent dimensions and batch examples.

F operates on 192-dimensional latents using two residual MLP blocks. Each block applies LayerNorm, a linear expansion to width 768, GELU, and a linear projection back to 192 dimensions before the residual addition. A final LayerNorm and linear projection produce the output. The input consists of one latent representation. Predictor inputs and encoder targets are detached before the losses associated with F, so their direct gradients update only the parameters of F.

Training uses AdamW with learning rate 5×10^{-5}, weight decay 10^{-3}, batch size 128, bfloat16 precision, and global gradient clipping at norm 1.0. The LeWM SIGReg coefficient is 0.09. Both the step and recursive terms for F have coefficient 1.0, as does the complete $\mathcal{L}_F$ objective in the training loop. The primary paired comparisons use the epoch-10 checkpoint from one training run per task; Section 5.5 additionally evaluates matched LeWM and HAS scoring across three independently trained checkpoints per task.

The checkpoint-producing implementation also includes a parameter-disjoint backward auxiliary head. Its inputs and targets are detached, so its loss supplies direct gradients only to that head. HAS evaluation uses the recursive scoring path described in Section 4.

B EVALUATION PROTOCOL

Task interfaces and preprocessing. PushT and TwoRoom use `swm/PushT-v1` and `swm/TwoRoom-v1`, respectively. Reacher uses `swm/ReacherDMControl-v0` in `qpos_match` mode: the target is the reference joint configuration, rather than only the end-effector position. The corresponding evaluation datasets are `pusht_expert_train`, `tworoom`, and `dmc/reacher_random`. Images are normalized with ImageNet statistics and resized to 224×224. Reference state fields initialize each environment and its goal; planning uses the resulting image observations. For PushT these fields are the dataset state, for TwoRoom they are proprioception, and for Reacher initialization uses joint positions and velocities with target joint positions.

Planner and paired starts. CEM samples 300 candidates per update, retains the 30 lowest-cost candidates, and performs 30 distribution updates with initial variance scale 1.0. The default LeWM and HAS configurations plan $h = 5$ action blocks and execute a prefix of five blocks before replanning, giving $H = 25$ and $R = 25$. The main longer-span LeWM control plans $h = 10$ blocks but executes the first five before replanning, giving $H = 50$ and $R = 25$. The second longer-span control plans and executes ten blocks, giving $H = 50$ and $R = 50$. Both retain the same candidate, elite, and update counts. Table 1 reports the two controls alongside the default LeWM and HAS configurations.

Goals are drawn at offsets $o \in \{25, 50, 75, 100\}$ environment steps, with budget $2o$. For each task and offset, evaluation groups 42–51 each contain 50 episodes. All methods reuse the corresponding start manifests, including initial states and sampled goals. Within a group, the shared-start protocol selects starts eligible for the maximum offset and reuses them across the four offsets. An episode succeeds if the environment reports success termination within the interaction budget; successful episodes stop early. The paired design matches initial states, goals, and planner settings, without requiring identical realized candidate sequences after scores and online trajectories diverge. We compute one success rate per group and configuration, then report the mean and sample standard deviation of these ten rates. The fixed-depth comparison uses the same protocol and the same checkpoint within each task. For the matched training-run comparison in Section 5.5, we first average the ten group-level rates within each checkpoint and then report the mean and sample standard deviation across the three checkpoint means. The resulting 27 paired gains are indexed by task, long offset, and training seed; comparisons at different offsets can share starts.

Depth schedules. The dynamic rule uses $k = \max((o - e - H)/b, 0)$, with every positive remainder divisible by b in the evaluated configurations. When $k = 0$, F is bypassed. For fixed-depth evaluation, the specified depth overrides this rule at every replanning step, including steps after the dynamic extension would have stopped. The ablation therefore evaluates a depth schedule over the whole execution.

C SCORING DIAGNOSTIC PROTOCOL

Figure 4 uses 400 offline trajectory windows per task, sampled after excluding every episode listed in the ten evaluation-group manifests. Each window covers at least 100 environment steps and is encoded using the same epoch-10 checkpoint as the control experiments. For each window, z_H is the encoder representation at $H = 25$ and z_g corresponds to offset $o \in \{50, 75, 100\}$. The recursion depth is $k = (o - H)/b$ with $b = 5$. The plotted quantities are Euclidean norms; the planner uses their squares. This diagnostic evaluates encoded observations and their recursively extended representations, with no CEM candidate search. Evaluation-disjoint here refers to the closed-loop evaluation episodes; it does not assert that these windows were excluded from world-model training. A change in this marginal distance distribution is descriptive evidence about representations, not a test of candidate ranking fidelity.

D TRM REIMPLEMENTATION UNDER THE SHARED TWOROOM PROTOCOL

Scope and data. The comparison in Section 5.3 transfers the replacement and hybrid terminal costs described by TRM (Li et al., 2026a) to our frozen TwoRoom LeWM checkpoint. It is evaluated entirely under our shared protocol rather than mixing success rates from a different checkpoint or task configuration. The temporal head uses the same offline TwoRoom trajectories as the world model, and the union of the 488 episodes appearing in evaluation-group manifests 42–51 is excluded before pair construction. We sample 100,000 training pairs and 10,000 validation pairs from within individual trajectories, balancing temporal separations $\Delta \in \{1, \ldots, 100\}$. The regression target is $\Delta/224$. Independent pair sampling produces 104 shared unordered endpoint pairs across the training and validation sets; neither set contains an evaluation episode.

Head and optimization. For latent dimension $d = 192$, the pairwise input is

$$[z_i,\ z_j,\ z_i - z_j,\ |z_i - z_j|] \in \mathbb{R}^{4d}. \tag{9}$$

The head contains two 256-unit SiLU layers followed by a scalar Softplus output, for 262,913 parameters. We train it for 20 epochs with Smooth-L1 loss, AdamW, learning rate 10^{-3}, weight decay 10^{-4}, and batch size 1024. The checkpoint is selected by minimum validation loss, which occurs at epoch 13; closed-loop success is not used for model or hyperparameter selection. The selected head has validation Smooth-L1 loss 0.00278657 and an unscaled temporal-distance MAE of 13.29 environment steps. The head-training seed is 3072; training uses float32 and 1,960 optimizer steps, with no learning-rate scheduler or gradient clipping. The validation set selects the checkpoint under the pair-sampling protocol above, which permits shared endpoints and the reported exact-pair overlap.

Planning costs. Let $z_{\text{end}} = P^{(h)}(z_e, a_{0:h-1})$ be the action-conditioned candidate endpoint, z_g the goal representation, and m_ϕ the frozen temporal head. The two TRM variants use

$$C_{\text{TRM}} = m_\phi(z_{\text{end}}, z_g), \tag{10}$$

$$C_{\text{replace}} = C_{\text{TRM}}, \tag{11}$$

$$C_{\text{hybrid}} = \text{zscore}_S(C_{\text{lat}}) + \lambda\, \text{zscore}_S(C_{\text{TRM}}), \qquad \lambda = 1, \tag{12}$$

where $C_{\text{lat}} = \|z_{\text{end}} - z_g\|_2^2$, matching the sum-of-squares planner cost in Equation 2. For each current CEM candidate pool S,

$$\text{zscore}_S(c_n) = \frac{c_n - \bar{c}_S}{\sqrt{|S|^{-1} \sum_{m \in S} (c_m - \bar{c}_S)^2 + 10^{-8}}}. \tag{13}$$

The hybrid weight is fixed before closed-loop evaluation. Both costs retain the frozen encoder and action-conditioned predictor, together with the CEM proposal family, initialization protocol, sample budget, and update rule. Changing the scores changes the selected elites and hence the fitted proposal parameters.

Evaluation. TRM-Replace and TRM-Hybrid reuse the exact TwoRoom start manifests and success criterion from Table 1. Evaluation uses groups 42–51, 50 episodes per group, offsets $o \in \{25, 50, 75, 100\}$, interaction budget $2o$, and the same $H = R = 25$ planner with 300 candidates, 30 elites, and 30 CEM updates. The same frozen temporal-head checkpoint is used for both variants. Training pairs contain encoder representations of observed trajectory states, while planning applies the head to predicted candidate endpoints; Table 2 therefore measures the closed-loop transfer of the learned temporal metric to the planner's endpoint distribution.

E CLOSED-LOOP PLANNING LATENCY

Protocol. We measure the wall-clock cost of the closed-loop replanning branch on one NVIDIA RTX 4000 Ada Generation GPU. The benchmark uses $o = 100$, an interaction budget of 200 environment steps, and the first ten start–goal pairs from evaluation group 42 for each task. Episodes run sequentially with one environment and video disabled. LeWM and HAS use $H = R = 25$, while the same-cadence longer-horizon control uses $H = 50, R = 25$. All configurations reuse the paper checkpoints and CEM settings $N = 300$, $M = 30$, and $J = 30$. We synchronize CUDA immediately before and after each timed interval and exclude the first two replan calls for every task–method pair as warm-up.

Metric. Replan-branch latency begins when a replan is requested and ends after the new action sequence has been written to the receding-horizon buffer. It includes input slicing, the HAS depth calculation, CEM, and buffer update. The CEM-only measurement times the solver call inside this branch. Both exclude environment stepping, reset, dataset and video I/O, buffered-action execution, and result serialization. We do not use full-episode wall time because early success changes the number of replans. Sample counts in Table 5 therefore denote non-warm-up replan calls rather than episodes.

Table 5: CUDA-synchronized replan-branch latency at $o = 100$ in milliseconds per replan. Percentiles and means pool non-warm-up calls within each task and method; n is the resulting number of calls. Relative cost divides the mean by default LeWM for the same task.

Task	Method (H/R)	n	Mean	p50	p95	CEM mean	Relative
PushT	LeWM (25/25)	65	739.5	739.9	746.8	739.0	1.000×
	HAS (25/25)	66	767.1	756.6	804.1	765.8	1.037×
	LeWM (50/25)	74	1588.8	1588.4	1592.1	1588.1	2.148×
TwoRoom	LeWM (25/25)	56	753.3	753.6	756.8	752.9	1.000×
	HAS (25/25)	46	777.1	773.3	808.9	776.4	1.032×
	LeWM (50/25)	68	1581.2	1582.0	1590.3	1580.3	2.099×
Reacher	LeWM (25/25)	16	754.4	753.8	757.4	753.5	1.000×
	HAS (25/25)	24	790.3	791.8	809.5	790.1	1.048×
	LeWM (50/25)	59	1591.1	1590.7	1593.4	1590.2	2.109×

The unweighted mean of the three task-wise relative costs is 1.039× for HAS and 2.119× for LeWM with $H = 50, R = 25$. Under this dynamic-depth, single-environment protocol, HAS adds about 4% to mean replanning time, whereas doubling the action-conditioned CEM horizon approximately doubles it. This average pools the depths actually visited during closed-loop execution; it is not the overhead of holding $k = 15$ fixed. The CEM and full-branch means differ by only 0.5–1.5 ms, showing that the measured branch is solver-dominated. The recursive applications of F are batched over the CEM candidate pool.

Table 6: HAS replan-branch latency as the adaptive depth decreases. Each cell reports mean/p50/p95 milliseconds with the non-warm-up call count n in parentheses.

Task	$k = 15$	$k = 10$	$k = 5$	$k = 0$
PushT	804.6/803.9/813.1 (9)	785.6/785.8/788.0 (9)	768.7/768.9/771.0 (10)	753.5/752.0/761.8 (38)
TwoRoom	808.8/808.0/815.8 (9)	790.2/790.9/791.9 (9)	773.8/774.3/776.0 (9)	757.5/757.5/760.1 (19)
Reacher	808.3/807.6/810.3 (9)	791.5/791.7/792.1 (7)	776.1/776.6/777.5 (5)	757.2/757.3/757.3 (3)

The HAS premium is depth-dependent and temporary: initial $k = 15$ calls take approximately 805–809 ms, and latency falls as the remaining-gap schedule tapers. At $k = 0$, F is bypassed and the

measured latency returns to approximately 753–757 ms. The smaller sample counts at later depths, especially for Reacher, arise because execution stops when the environment reports success. This single-GPU benchmark measures per-replan latency rather than batched throughput; it is separate from the success-rate evaluation in Table 1, and TRM variants are not included.

F SENSITIVITY TO SCHEDULE-OFFSET PERTURBATIONS

Intervention and protocol. We distinguish the reference-trajectory goal offset o from the offset supplied to the recursion schedule, $\tilde{o} = \mathrm{round}(so)$, with $s \in \{0.8, 1.2\}$. At elapsed time e, the perturbed schedule uses

$$k(e, \tilde{o}) = \max\left(\frac{\tilde{o} - e - H}{b}, 0\right), \qquad H = 25, \quad b = 5. \tag{14}$$

All tested assumed offsets are multiples of b, so the positive depths are integers without further rounding. Only the offset used to calculate depth changes: the goal image is still sampled at o, and the interaction budget remains $2o$. Both H and R remain 25 environment steps. Each task uses its original epoch-10 checkpoint, the same start manifests, and the CEM configuration of Table 1: 300 candidates, 30 elites, and 30 updates. No model is retrained.

The experiment covers PushT, TwoRoom, and Reacher at $o \in \{50, 75, 100\}$, with ten evaluation groups (42–51) and 50 episodes per group. The two perturbation scales are specified by the protocol rather than selected by success rate. They yield 180 new evaluation units, or 9,000 episodes, with video disabled and 50 parallel environments per unit. The unperturbed HAS and LeWM references are reused from the original paired evaluation in Table 1; neither reference is rerun here. The experiment does not include $o = 25$ or an image-based temporal-scale estimator.

Depth schedules. Table 7 lists the assumed offsets and scheduled depths at $e = 0, 25, 50, \ldots$, through the first zero. Subsequent depths remain zero. These lists specify the rule rather than the number of planning calls actually executed, since episodes can terminate early. Unlike a fixed-depth override, each perturbed schedule still decreases during execution. For example, at $o = 100$ and $e = 75$, $s = 1.2$ retains $k = 4$ where the unperturbed rule uses $k = 0$.

Table 7: Assumed offsets and depth schedules. Each entry gives $\tilde{o}$ followed by the scheduled depths in brackets. The interaction budget remains $2o$ for every scale.

True o	Budget	$s = 0.8$	$s = 1$	$s = 1.2$
50	100	$40:\ [3, 0]$	$50:\ [5, 0]$	$60:\ [7, 2, 0]$
75	150	$60:\ [7, 2, 0]$	$75:\ [10, 5, 0]$	$90:\ [13, 8, 3, 0]$
100	200	$80:\ [11, 6, 1, 0]$	$100:\ [15, 10, 5, 0]$	$120:\ [19, 14, 9, 4, 0]$

Table 8 reports the unweighted mean and sample standard deviation of the ten group-level success rates for every task, offset, and scale.

Table 8: Success under schedule-offset perturbations (%; mean ± sample standard deviation over ten evaluation groups of 50 episodes). LeWM and HAS at $s = 1$ reuse Table 1. Bold marks the highest mean in each row. All methods share the reference goal image, interaction budget, and starts.

Task	o	LeWM	HAS ($s = 0.8$)	HAS ($s = 1$)	HAS ($s = 1.2$)
PushT	50	42.8 ± 6.0	60.0 ± 5.8	$\mathbf{67.8 \pm 5.8}$	58.2 ± 6.4
	75	21.6 ± 7.4	32.0 ± 7.1	$\mathbf{40.4 \pm 4.2}$	34.2 ± 2.9
	100	13.2 ± 4.0	22.6 ± 6.0	$\mathbf{23.4 \pm 6.7}$	21.0 ± 3.8
TwoRoom	50	42.4 ± 5.5	96.6 ± 1.9	80.4 ± 5.3	$\mathbf{98.4 \pm 1.8}$
	75	25.8 ± 8.3	$\mathbf{95.0 \pm 2.7}$	82.0 ± 6.1	94.6 ± 4.1
	100	16.4 ± 6.0	$\mathbf{78.8 \pm 5.2}$	70.4 ± 7.2	73.0 ± 7.0
Reacher	50	92.6 ± 2.3	95.6 ± 2.5	$\mathbf{96.0 \pm 1.3}$	94.4 ± 2.3
	75	85.8 ± 4.8	94.6 ± 2.8	95.0 ± 2.2	$\mathbf{95.2 \pm 3.0}$
	100	81.0 ± 4.9	90.6 ± 2.1	90.8 ± 3.0	$\mathbf{91.6 \pm 2.5}$

Results. Comparing the HAS columns measures sensitivity to the supplied offset, whereas comparison with LeWM tests whether the advantage over native endpoint scoring persists. PushT achieves its highest mean at $s = 1$ for all three offsets; the largest decrease under perturbation is 9.6 percentage points at $o = 50, s = 1.2$. Reacher varies by at most 1.6 points across the three scales at any offset. Both perturbed scales exceed the unperturbed mean on TwoRoom, with the better scale varying by offset. This task dependence shows that the nominal schedule need not maximize success among nearby schedules, consistent with the depth preferences in Table 3.

Across the 180 paired evaluation-group comparisons with LeWM, perturbed HAS is higher in 169, lower in 7, and tied in 4. At the task–offset–scale mean level, all 18 perturbed conditions exceed LeWM, by 1.8–69.2 percentage points. The results support preservation of HAS's mean advantage under the tested $\pm 20\%$ errors in the temporal scale supplied to the recursion schedule.