BEYOND TERMINAL MATCHING: ARRIVAL-AWARE PLANNING WITH WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Planning with a learned world model involves deciding not only how to reach a goal, but also when to reach it. Matching the goal only at a fixed terminal step does not explicitly favor earlier arrival. We introduce an arrival-aware planning objective for learned world models that requires neither model fine-tuning nor training of auxiliary components. The objective treats each prediction step as a candidate arrival time and assigns it a cost combining elapsed time with a penalty for predicted goal mismatch. These costs are aggregated through a smooth minimum, allowing action optimization to favor earlier goal attainment without prescribing a fixed arrival step. We analyze how this objective adaptively weights gradient contributions from different candidate arrival times. Experiments on visual goal-reaching tasks show reductions in budgeted physical interaction steps under a shared planning configuration. Controlled comparisons separate flexible endpoints, which improve across four LeWM tasks, from explicit time preference, which provides further gains on TwoRoom and Reacher. These findings highlight arrival-aware objectives as a means of improving physical-interaction efficiency without additional training.

1 INTRODUCTION

A successful prediction is not yet an efficient plan. Consider two predicted trajectories that end at the same goal: one reaches it early and stays there, while the other arrives only at the final step. For a task that ends at first success, the first plan would require fewer actions. Yet terminal matching assigns both plans the same cost. This ambiguity exists even with an exact dynamics model: a terminal score distinguishes where a trajectory ends, but not when it first reaches the goal.

The rollout already contains more temporal information than its terminal score uses. Visual model-predictive control optimizes actions through these predictions, executes a short prefix, and replans (Finn & Levine, 2017; Ebert et al., 2018; Hafner et al., 2019). Recent visual world models support goal-reaching with reusable representations (Zhou et al., 2025; Maes et al., 2026b). Their intermediate predictions offer an opportunity to favor earlier arrival without shortening the prediction window. The prediction horizon specifies how far to look ahead; it does not specify when to arrive.

Scoring these intermediate predictions is a natural first step, but allowing arrival at different times is not the same as preferring an earlier one. A smooth minimum over goal-mismatch penalties can favor a good match anywhere in the rollout, yet swapping the same penalties between earlier and later steps leaves its value unchanged. It captures endpoint flexibility, not an explicit preference over arrival order. To distinguish these cases, the objective must account for both how well a predicted state matches the goal and when that match occurs.

We introduce *Temporal Integration of Mismatch and Elapsed time* (TIME), a planning objective that brings these two considerations together. TIME treats every predicted step as a candidate arrival time and combines its normalized goal-mismatch penalty with an elapsed-time cost. A smooth minimum aggregates these costs, allowing arrival timing to influence action optimization without prescribing a fixed arrival step. The time term changes how different endpoints contribute to the action gradient, while the full prediction window is still evaluated. TIME requires neither model fine-tuning nor auxiliary training and integrates into receding-horizon planning (Figure 1).

We make the following contributions:

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

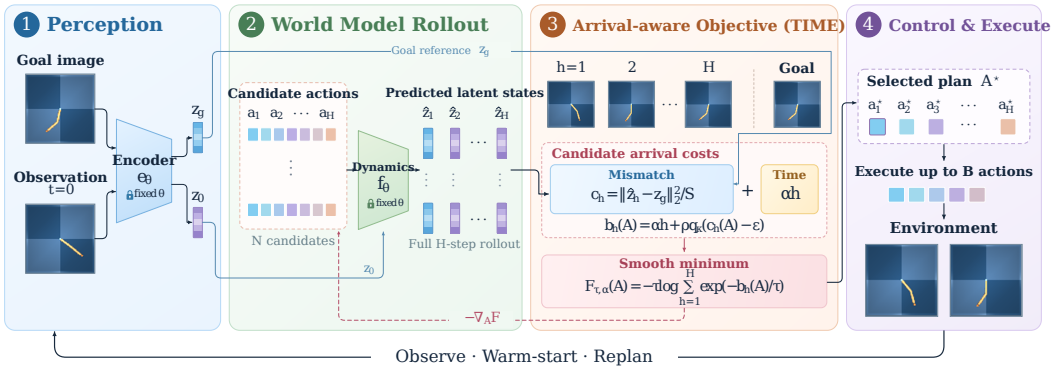

Figure 1: **TIME planning through a learned world model.** The encoder maps observation and goal images to latent representations. Candidate actions generate a full H-step prediction window; each endpoint receives a cost combining goal mismatch and elapsed time. A smooth minimum allocates gradients across endpoints. Dashed arrows indicate action optimization through fixed model parameters. The best encountered plan $A^\star$ supplies at most B physical actions before replanning; execution stops at native success or the budget limit. Observation, goal, and environment thumbnails show PushT inputs; intermediate predicted-state thumbnails are schematic.

- **An arrival-aware objective.** We formulate a differentiable objective for planning through learned world models whose parameters are held fixed during planning. Its zero-time-weight variant, Free, retains endpoint flexibility and enables a controlled comparison of planning with and without explicit time preference.
- **Temporal gradient analysis.** We characterize how candidate arrival costs redistribute gradient contributions across endpoints and establish conditional minimum-time approximation and perturbation bounds on the predicted latent grid.
- **Controlled empirical evaluation.** Under a shared planning configuration, Free reduces mean budgeted physical steps relative to Terminal on all four LeWM tasks; adding time preference further improves TwoRoom and Reacher. On Reacher, TIME reduces mean steps from 19.19 to 14.46 relative to Free, with both succeeding in all evaluated executions.

2 Related Work

Planning with learned visual dynamics. World Models learns compact recurrent representations for control (Ha & Schmidhuber, 2018); visual foresight and PlaNet use learned predictions for online MPC (Finn & Levine, 2017; Ebert et al., 2018; Hafner et al., 2019). TD-MPC and TD-MPC2 combine latent planning with learned value estimation (Hansen et al., 2022; 2024), whereas Dreamer learns policies from imagined trajectories (Hafner et al., 2025). DINO-WM and LeWorldModel provide the frozen visual backbones used in our study (Zhou et al., 2025; Maes et al., 2026b). Recent approaches learn planning components: Traj-LeWM learns a trajectory cost, ProWorld learns goal-progress structure, and Reinforced Planning learns additional planning components (Huang et al., 2026; Liu et al., 2026; Sommer & Schilling, 2026). ProWorld combines terminal, best-intermediate and mean goal distances at planning time. TIME instead studies flexible endpoint aggregation and elapsed-time costs with the released model held fixed. LEAP also differentiates frozen LeWM rollouts, but adds a decoder, action proposal and numerical goal descriptor (Pham & Bera, 2026). Our primary LeWM study uses image-only inputs without auxiliary training; the DINO-WM extension retains its native visual and proprioceptive inputs.

The planning objective and numerical solver play distinct roles. CEM, iCEM, MPPI and GRASP provide alternative optimization strategies (de Boer et al., 2005; Pinneri et al., 2020; Williams et al., 2017; Psenka et al., 2026). We hold the solver fixed for objective comparisons and use disclosed, development-selected configurations for complete planning systems.

Goal conditioning and temporal aggregation. Universal value function approximators represent value jointly over states and goals, while hindsight experience replay makes sparse goal feedback more usable (Schaul et al., 2015; Andrychowicz et al., 2017). These methods learn goal-conditioned values or relabel experience. TIME changes the deployment-time objective applied to a fixed predictor. LePlanner trains an arrival-aware controller indexed by an offline goal's temporal offset (Bansal et al., 2026). TIME instead optimizes actions at deployment and evaluates first physical success without subsequent holding. Appendix B compares the objectives.

Horizon-conditioned learning, final-time optimization and horizon optimization provide further precedents (Pong et al., 2018; Sun et al., 2015; Stachowicz & Theodorou, 2021). Weighted state-norm sums serve as minimum-time surrogates in linear control and particle MPC (Bako et al., 2011; Echigo et al., 2024). Differentiable temporal logic uses smooth temporal extrema, weighted temporal logic expresses satisfaction-time preferences, and model-predictive control with signal temporal logic makes temporal specifications explicit (Raman et al., 2014; Leung et al., 2023; Mehdipour et al., 2020). These operators are established temporal ingredients. TIME instantiates them as a concrete deployment-time composition for visual planning: a normalized latent-mismatch penalty is evaluated at every predicted endpoint, elapsed model-block cost is added before smooth aggregation, and the shared Terminal–Free–TIME controls separate endpoint flexibility from time preference under one learned model and solver.

3 Arrival-aware planning with a learned world model

During planning, the released encoder and predictor are held fixed; TIME changes only the action objective. Figure 1 shows the architecture, and Algorithm 1 specifies the planning loop.

Planning interface. We first describe the image-only interface used in the primary LeWM study. Let e_θ denote the frozen image encoder and f_θ the frozen autoregressive predictor. Given an observation o and goal image o_g, the planner optimizes $A = (a_1, \dots, a_H)$ over the bounded action domain $\mathcal{A}$. Each a_h contains a block of B physical actions. All experiments use $B = 5$; the primary objective comparison uses $H = 8$. A rollout therefore predicts BH physical actions, whereas each MPC call executes at most one block. External planners retain their development-selected horizons (Section 5). The rollout yields latent predictions $\widehat{z}_h(A)$. With $z_g = e_\theta(o_g)$, define

$$C_h(A) = \|\widehat{z}_h(A) - z_g\|_2^2, \qquad S = \max\{\|e_\theta(o) - z_g\|_2^2, 10^{-6}\}, \qquad c_h(A) = C_h(A)/S. \quad (1)$$

The scale S is fixed within each planning call and recomputed at the next observation. Thus $c_h \le \epsilon$ describes relative latent-error reduction, not a calibrated physical goal region. In the primary study, simulator states are used only to evaluate native success and never enter the planner. The DINO-WM extension uses its native input interface, as detailed in Section 5.

An objective over candidate arrival times. For tolerance $\epsilon \ge 0$, penalty scale $\rho > 0$ and smoothing width $\kappa > 0$, define

$$q_\kappa(x) = \kappa \log(1 + e^{x/\kappa}), \quad b_h(A) = \alpha h + \rho q_\kappa(c_h(A) - \epsilon), \quad F_{\tau,\alpha}(A) = -\tau \log \sum_{h=1}^{H} e^{-b_h(A)/\tau}. \quad (2)$$

Each branch b_h combines elapsed time with a smooth penalty on goal mismatch. The temperature $\tau > 0$ controls how strongly the aggregate concentrates on low-cost branches. Setting $\alpha = 0$ gives Free; a positive α additionally favors earlier endpoints, with $\alpha = 1$ used for the primary TIME configuration. Here α is measured in surrogate-cost units per model block: $\alpha = 1$ assigns one unit to each predicted block and is a reference configuration, not a calibrated physical-time claim. Multiplying every branch cost and τ by B changes only the cost units and leaves weights and minimizers unchanged. The horizon H bounds the prediction window, while the objective scores arrivals throughout that window. The mismatch penalty is soft and does not impose hard feasibility. We use stable softplus and log-sum-exp implementations with fixed reference values $\rho = 30$, $\epsilon = .05$, $\kappa = .01$ and $\tau = .25$. These values are not selected on the test cohort.

Algorithm 1 TIME: arrival-aware receding-horizon planning

Require: Learned encoder/predictor e_θ, f_θ (parameters held fixed during planning); goal image o_g; horizon H, block size B, action box $\mathcal{A}$; fixed solver settings; $\alpha = 0$ for Free or $\alpha = 1$ for TIME.

Initialize an empty warm-start suffix; observe o.
while native success is false and the physical-action budget remains **do**
  Encode $z_0 = e_\theta(o)$, $z_g = e_\theta(o_g)$; fix $S = \max\{\|z_0 - z_g\|^2, 10^{-6}\}$.
  Initialize bounded action candidates using the fixed seed and warm start.
  for each optimization iteration **do**
    Expand current coordinates into full action sequences $A \in \mathcal{A}$.
    Predict $\widehat{z}_{1:H}(A)$ with f_θ; compute $c_h = \|\widehat{z}_h - z_g\|^2 / S$.
    Score $b_h = \alpha h + \rho q_\kappa(c_h - \epsilon)$ and $F(A) = -\tau \log \sum_h e^{-b_h/\tau}$.
    Retain the lowest-F full plan seen so far.
    Backpropagate F; take a projected Adam step under the fixed coordinate schedule.
  end for
  Rescore the final population; apply bounded L-BFGS-B within the polishing budget.
  Retain the best full plan $A^\star$ across all evaluated candidates and trials.
  Execute its first block; stop on native success or the action budget, checking each action.
  Shift $A^\star$ to form the next warm start; observe the new image o.
end while

Framework variants and the terminal reference. Writing $p_h = \rho q_\kappa(c_h - \epsilon)$ gives the three objectives

$$J_{\text{terminal}} = p_H, \qquad J_{\text{free}} = -\tau \log \sum_h e^{-p_h/\tau}, \qquad J_{\text{TIME}} = -\tau \log \sum_h e^{-(h+p_h)/\tau}. \quad (3)$$

Free and TIME are members of the same family: $J_{\text{free}} = F_{\tau,0}$ and $J_{\text{TIME}} = F_{\tau,1}$. The Terminal–Free comparison isolates flexible endpoint aggregation; the Free–TIME comparison isolates the added elapsed-time cost. All three share the same normalized, smoothed penalty p_h. Free can reward several low-penalty endpoints through the entropy of the smooth minimum, so it does more than select a single endpoint. It is also invariant to permutations of endpoint costs. The analytic example in Figure 5c shows how the time term supplies ordering information that Free omits.

Shared numerical optimization. The three objectives use the same bounded Gaussian initialization, with a reproducible seed for each case and planning decision. Their first populations are identical and include a projected standardized-zero candidate representing the projected dataset action mean. Subsequent calls warm-start from each condition's own plan, allowing their populations and executed trajectories to diverge. We select among projected Adam, a coarse-to-fine Adam schedule, and that schedule followed by bounded L-BFGS-B refinement on development cases. Selection uses TIME performance, and the chosen solver is shared across objectives. Free selects the same configuration under the same rule (Appendix D). The comparison therefore evaluates objective changes within a common solver, rather than independent tuning of each objective. Appendix C gives the initialization, schedule and refinement procedure.

Each condition retains the best candidate encountered under its own objective and receives the same refinement budget. Every returned plan spans the full H-block window, so endpoint preference does not alter its warm-start length. The controls use no separate endpoint-search stage.

Receding-horizon execution. The planner executes only the first action block, then observes the environment and replans. Evaluation checks native physical success before acting and after every action, stopping on success or exhaustion of the execution budget. Predicted latent feasibility does not determine reported success or terminate execution. The model scores endpoints every B physical actions, whereas evaluation can detect success within a block. A goal visit between these endpoints can therefore be missed even with exact predictions (Appendix A.5).

4 Analysis of endpoint selection and time preference

We analyze the endpoint allocation induced by this deployment-time objective on the predicted latent grid: elapsed block cost tilts endpoint weights, those weights enter causal action gradients, and a

nonsmoothed penalty can recover the earliest latent-threshold endpoint. Native physical success is evaluated separately.

Endpoint weights and causal action gradients. Write $p_h(A) = \rho q_\kappa(c_h(A) - \epsilon)$ and $b_h(A) = \alpha h + p_h(A)$, where $\alpha \geq 0$ is the time weight. The Gibbs variational identity gives

$$F_{\tau,\alpha}(A) = \min_{w \in \Delta_H} \sum_{h=1}^{H} w_h b_h(A) + \tau \sum_{h=1}^{H} w_h \log w_h, \qquad w_h^*(A) = \frac{e^{-b_h(A)/\tau}}{\sum_j e^{-b_j(A)/\tau}}. \tag{4}$$

The weights express a soft allocation across endpoints, not calibrated success probabilities. Autoregressive causality makes c_h independent of actions after h, yielding

$$\nabla_{a_j} F_{\tau,\alpha}(A) = \frac{\rho}{S} \sum_{h=j}^{H} w_h^*(A)\ \mathrm{sigmoid}\left(\frac{c_h(A) - \epsilon}{\kappa}\right) \nabla_{a_j} C_h(A). \tag{5}$$

Terminal matching concentrates endpoint weight at H, whereas the arrival-aware objective distributes it according to candidate arrival costs. When an early branch dominates the weights, later endpoint gradients are downweighted. All finite-cost branches retain positive weight in exact arithmetic, although small weights can underflow in the implementation (Appendix C). The full prediction window is still evaluated, so this redistribution does not itself reduce rollout computation.

Proposition 1 (Time preference at a fixed trajectory)**.** *For fixed A, the time-aware and free-endpoint weights satisfy $w_h^\alpha \propto w_h^0 e^{-\alpha h/\tau}$. Moreover,*

$$\frac{\partial}{\partial \alpha} \sum_h h w_h^\alpha = -\frac{1}{\tau} \mathrm{Var}_{w^\alpha}(h) \leq 0. \tag{6}$$

Increasing the time weight shifts the weighted endpoint index earlier along a fixed trajectory. Exact global reoptimization preserves this ordering of weighted indices (Appendix A.2). Neither result guarantees earlier physical success for a finite local solver. The same concentration can also suppress useful gradients from longer branches. If m branches cost at least Δ more than the best branch, their total weight is at most $me^{-\Delta/\tau}$. Temperature therefore controls both the intended endpoint concentration and its potential optimization cost; its practical choice remains empirical.

Discounted-return interpretation (not a separate objective). Put $\gamma = e^{-\alpha/\tau}$ and $r(c) = \exp[-\rho q_\kappa(c - \epsilon)/\tau]$. Then

$$F_{\tau,\alpha}(A) = -\tau \log R(A), \qquad R(A) = \sum_{h=1}^{H} \gamma^h r(c_h(A)). \tag{7}$$

The transformed return has Free as its undiscounted special case, and minimizing $F_{\tau,\alpha}$ or maximizing R gives the same exact optima. The identity connects the time term to discounted aggregation; finite optimizer steps can still differ because descent is scaled by $\tau/R(A) > 0$.

Connection to a discrete minimum-time problem. Let the action box $\mathcal{A}$ be compact and each c_h continuous. Suppose the normalized latent-mismatch threshold is attainable at at least one predicted endpoint, and define $h^* = \min\{h : \min_{A \in \mathcal{A}} c_h(A) \leq \epsilon\}$. For $h < h^*$ define $\delta_h = \min_A(c_h(A) - \epsilon) > 0$. For unit time weight, let $P(A) = \min_h\{h + \rho[c_h(A) - \epsilon]_+\}$.

Proposition 2 (Penalty and smoothing bounds)**.** *If $\rho > \max_{h<h^*}(h^* - h)/\delta_h$ (vacuous when $h^* = 1$), then $\min_A P(A) = h^*$. For every A,*

$$P(A) - \tau \log H \ \leq\ F_{\tau,1}(A)\ \leq\ P(A) + \rho\kappa \log 2. \tag{8}$$

The bound separates the nonsmoothed penalty's exact value from the smoothing approximation, without requiring feasibility to be monotone in time. Its penalty threshold and global optimization gap are unknown in our experiments. Smoothing can also yield an infeasible minimizer (Appendix A). Thus the result does not certify minimum-time physical arrival: the implemented threshold measures relative latent mismatch.

Model-error boundary. The bounds above concern the surrogate predicted costs; Appendix A gives the corresponding perturbation bounds, while model error and finite optimization govern transfer to native physical success.

5 EXPERIMENTS

Evaluation design. We evaluate whether arrival-aware objectives reduce physical interaction steps, which components drive the gains, and how those gains depend on the planning configuration. The primary study uses frozen LeWM checkpoints for TwoRoom, PushT, Reacher and Cube, with 64 episodes held out from planner development per task and three planning seeds: 6,144 executions across the eight reported conditions. Goals are drawn 50 physical actions ahead in recorded trajectories; execution is capped at 100 actions. Methods share episode identities, image-only inputs, action standardization and native success criteria. Sixteen separate development episodes per task determine the solver configuration before testing (Appendix D). Terminal, Free and TIME share $H = 8$ and $B = 5$, predicting 40 physical actions and executing at most five before replanning. The five literature baselines (CEM, iCEM, MPPI*, GD and GRASP) use per-task selections from $H \in \{3, 5, 8\}$ and two compute budgets. These system comparisons complement the controlled objective comparison; their settings are reported separately in Table 6.

Outcome and inference. Our primary outcome is the mean number of *budgeted physical steps*: first-success time capped at 100, with zero assigned to initial success. Keeping unsuccessful executions at the cap captures both arrival speed and success coverage. We also report success rates and results restricted to cases requiring action. Estimates first average planning seeds within each episode, then average episodes within each task. Bootstrap intervals resample episodes and preserve pairing for contrasts. We use paired tests with Holm correction for primary comparisons; Appendix F reports the effects, marginal intervals and adjusted p-values. Unresolved comparisons are not evidence of equivalence. Figure 2 shows the success-by-step curves, and Table 1 reports the corresponding means and success rates.

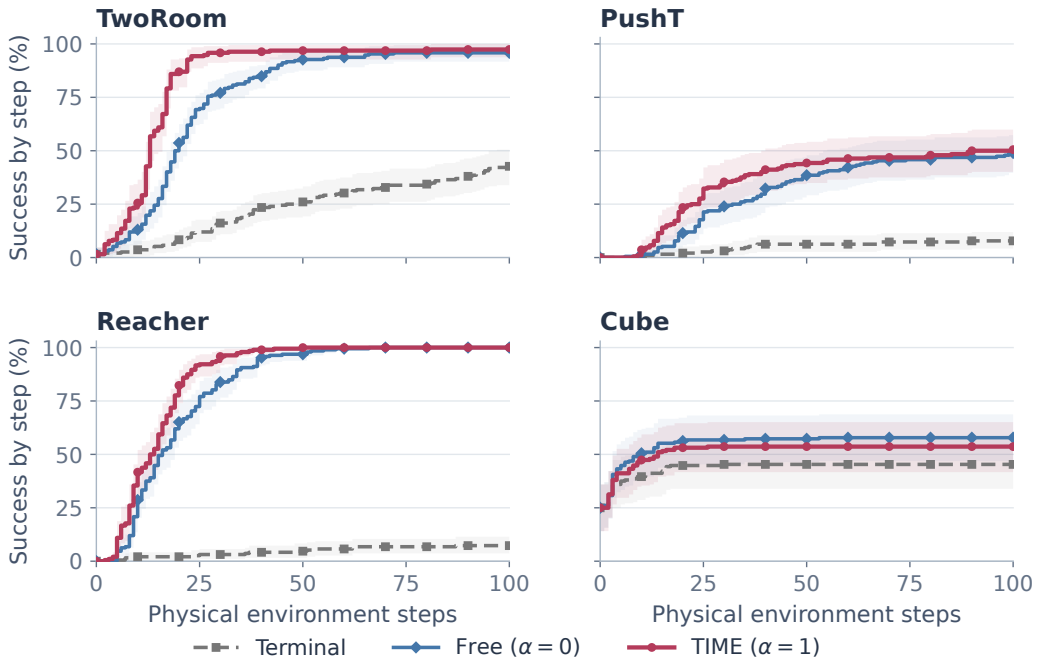

Figure 2: **Flexible endpoint objectives improve the observed success-by-step curves.** Free is TIME's zero-time-weight special case. All three objectives share $H = 8$, the solver, 64 test episodes per task and three planning seeds. Curves show unsmoothed empirical success accumulation, including initial successes at zero. Shading denotes pointwise 95% episode-bootstrap intervals, not simultaneous bands across time or methods.

Table 1: **Contributions of endpoint flexibility and time preference.** Mean budgeted physical steps and success rate under the shared solver, using 64 episodes per task and three seeds. All objectives use $H = 8$; Free is the $\alpha = 0$ special case of the TIME framework. Bold denotes the best point estimate for each task and metric, without implying statistical significance.

	TwoRoom		PushT		Reacher		Cube	
Objective	Steps ↓	Success ↑	Steps ↓	Success ↑	Steps ↓	Success ↑	Steps ↓	Success ↑
Terminal	76.73	42.7	94.99	7.8	95.36	7.3	56.10	45.3
Free endpoint	25.08	95.8	69.27	49.0	19.19	**100.0**	**44.59**	**57.8**
TIME	**15.78**	**97.4**	**64.20**	**50.5**	**14.46**	**100.0**	48.30	53.6

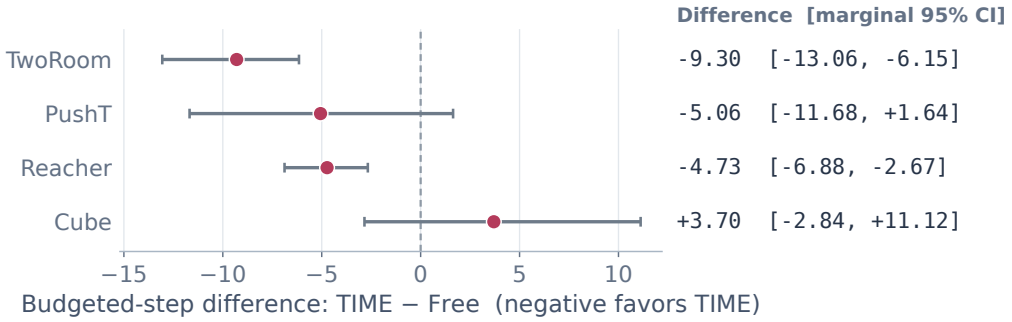

Figure 3: **Explicit time preference further reduces steps on TwoRoom and Reacher.** Points and whiskers show TIME–Free differences in mean budgeted steps and marginal paired 95% intervals. Negative values favor TIME. PushT and Cube intervals include zero; Cube's point estimate favors Free. Intervals are unadjusted, while primary inference uses the paired tests and Holm correction in Appendix F.

Endpoint flexibility and time preference. Both evaluated TIME variants achieved lower mean budgeted steps than Terminal on all four tasks (Table 1). Free reduced steps on every task; adding time preference further reduced TwoRoom steps by 9.30 and Reacher steps by 4.73 (Figure 3). Within the eight prespecified comparisons—Free versus Terminal and TIME versus Free on each task—all four Free–Terminal contrasts and the TwoRoom and Reacher TIME–Free contrasts remained significant after Holm correction ($p < 0.05$). The PushT difference was -5.06 steps ($[-11.68, 1.64]$), while Cube favored Free by 3.70 steps ($[-2.84, 11.12]$ for TIME–Free). Neither difference was resolved by the paired tests. These contrasts identify flexible endpoints as a broadly effective component and explicit time preference as a further, task-dependent refinement.

Arrival speed and success coverage. On Reacher, all 192 executions succeeded for each of Free and TIME, so the -4.73-step difference reflects earlier arrival rather than greater success coverage. Cube illustrates the separate role of coverage: TIME succeeded in 53.6% of all executions and 38.2% of active cases. Free succeeded in 43.8% of active cases (Appendix F).

Sensitivity to prediction horizon. We examined four horizons on 32 matched test episodes per task with seed 42 (Appendix D): 1,536 executions across three objectives, including 384 reused primary executions at $H = 8$. On Reacher, Terminal decreased from 92.66 steps at $H = 8$ to 70.78 at $H = 3$, while both framework variants had lower means at every tested horizon (TIME: 14.25–15.59). Cube showed a different ordering at $H = 3$: Terminal reached 28.56 steps, compared with 39.38 for Free and 33.25 for TIME. Objective choice and prediction horizon interact: flexible aggregation retains lower means across the tested Reacher horizons, whereas Cube favors Terminal at $H = 3$.

Comparison with literature planners. TIME achieved lower mean budgeted steps than every evaluated literature baseline on each of the four tasks (Table 2). Sixteen of the 20 prespecified comparisons were significant after Holm correction at $p < 0.05$. The remaining four comparisons,

Table 2: **Comparison with literature planners.** Mean budgeted physical steps and success rate, using 64 episodes per task and three seeds (192 executions per cell). Free and TIME share $H = 8$ and a solver; literature baselines use development-selected horizons and budgets. This table compares planning systems rather than isolating objective changes. Bold denotes the best point estimate for each task and metric, without implying statistical significance.

	TwoRoom		PushT		Reacher		Cube	
Method	Steps ↓	Success ↑	Steps ↓	Success ↑	Steps ↓	Success ↑	Steps ↓	Success ↑
CEM	78.27	25.5	85.10	21.9	39.95	83.9	52.73	49.0
iCEM	68.52	45.3	84.67	24.0	30.45	93.8	52.70	49.0
MPPI*	64.06	45.3	80.62	31.8	46.22	88.5	63.23	37.5
GD	60.83	54.7	75.09	41.1	32.87	93.2	50.90	51.0
GRASP	90.29	13.0	84.06	21.9	58.14	50.5	50.20	52.1
Free ($\alpha = 0$)	25.08	95.8	69.27	49.0	19.19	**100.0**	**44.59**	**57.8**
TIME	**15.78**	**97.4**	**64.20**	**50.5**	**14.46**	**100.0**	48.30	53.6

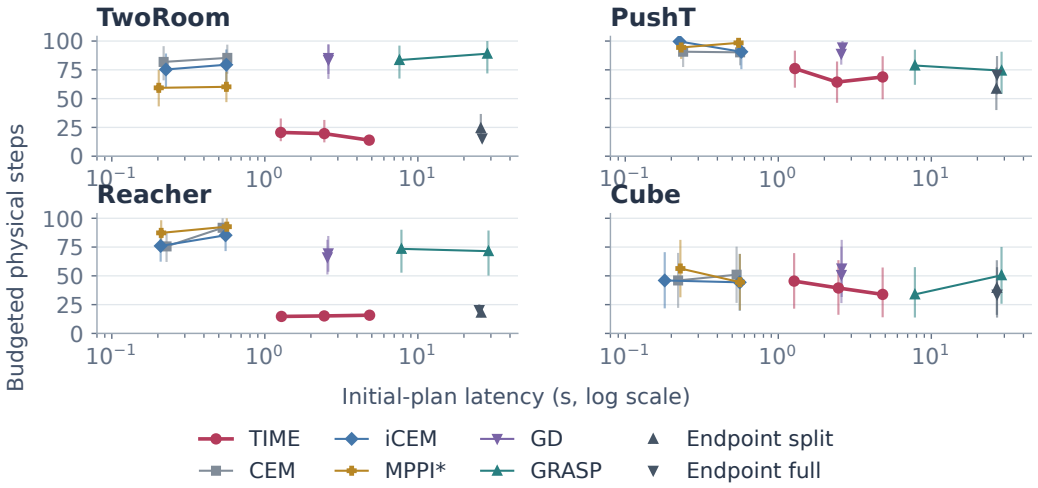

Figure 4: **Physical interaction steps and initial-plan latency at $H = 8$.** Whiskers denote marginal 95% episode-bootstrap intervals on 16 test cases. Latency is the median of nine initial solves on three development inputs, after warmup and without warm-start plans. Lines connect tested budgets, not an interpolated frontier. Latency excludes subsequent replanning; Appendix D gives the timing protocol and success rates.

all on Cube, were unresolved. Appendix F provides the paired contrasts and external success-by-step curves. These complete-system comparisons use different horizons, losses and search procedures. For example, Reacher GD uses $H = 3$, raw terminal latent error and AdamW, whereas Terminal uses $H = 8$, a normalized soft penalty and coarse-to-fine Adam with L-BFGS-B refinement; Table 6 maps these choices.

Interaction steps and planning latency. We compared budgeted steps and initial-plan latency at $H = 8$ on 16 matched test episodes per task with seed 42 (Figure 4). These declared compute-budget settings differ from the development-selected primary configurations. Timing uses identical development observations and exclusive GPU access; Appendix D specifies the scope and reports corresponding success rates. Default TIME required 4.81–4.84 seconds per initial solve, compared with 0.18–0.57 for the sampling presets and 25.18–26.88 for discrete endpoint search. Relative to sampling, TIME traded additional planning latency for fewer physical steps. Reduced optimization budgets produced task-dependent outcomes (Appendix D). Initial-plan latency excludes subsequent replanning and therefore does not establish an advantage in total completion time.

Cross-model evaluation with DINO-WM. We tested the objective with released DINO-WM checkpoints on PushT and Wall to examine applicability beyond LeWM (Zhou et al., 2025). The extension comprises 224 executions: 32 paired cases per task and three methods on both tasks, plus a Wall-only Free check, using one seed, $H = 8$, $B = 5$ and a 100-action cap. All methods receive DINO-WM's native visual and proprioceptive inputs and use prespecified reduced planning budgets. This is a system comparison under prespecified reduced budgets, rather than a reproduction of the original DINO-WM benchmark or an equal-compute study. PushT's 32 segments come from 21 trajectories, which form the resampling clusters for its intervals. Table 3 reports both tasks and all three methods.

On Wall, TIME reached all 32 goals in 19.00 mean steps, compared with 53.84 for CEM and 66.22 for GD. The paired descriptive 95% intervals excluded zero for both differences: TIME–CEM, -34.84 $[-43.22, -26.69]$; TIME–GD, -47.22 $[-57.72, -36.22]$. On PushT, TIME and GD each achieved 21.9% success, versus 18.8% for CEM. The step differences remained uncertain: TIME–CEM, -8.78 $[-23.92, 3.44]$; TIME–GD, -6.22 $[-17.79, 3.37]$. To separate the two components, we additionally reran Free with the same 32 Wall cases and native protocol. Free also succeeded on all cases but required 26.06 steps versus 19.00 for TIME; the paired Free–TIME difference was $+7.06$ $[1.44, 12.72]$. This Wall-only component check indicates that the cross-model reduction includes an explicit timing contribution, while PushT remains unresolved.

Table 3: **Cross-model evaluation with DINO-WM.** Mean budgeted steps (marginal 95% interval) and success rates among 32 paired cases per task. Failures receive 100 steps. PushT intervals resample 21 source-trajectory clusters; Wall intervals resample cases. The Wall Free row is a paired component check and reports its paired interval in the text. Bold denotes the best point estimate within each task and metric, without implying statistical significance.

Task	Method	Steps (95% interval)	Success rate ↑
PushT	**TIME**	**84.28 [71.56, 94.43]**	**21.9%**
	CEM	93.06 [87.85, 97.90]	18.8%
	GD	90.50 [82.52, 96.94]	**21.9%**
Wall	**TIME**	**19.00 [15.75, 22.91]**	**100.0%**
	CEM	53.84 [46.53, 61.78]	87.5%
	GD	66.22 [56.25, 76.38]	65.6%
	Free	26.06	100.0%

6 Discussion and conclusion

TIME makes the distinction between prediction horizon and preferred arrival time explicit in the planning objective. By aggregating candidate arrival costs, it uses intermediate predictions to guide action optimization without additional training. Free, its zero-time-weight special case, separates endpoint flexibility from explicit time preference; the analysis characterizes how both choices redistribute action gradients.

The experiments identify their complementary roles. Under the shared solver and horizon, Free reduces budgeted physical interaction steps relative to Terminal on all four LeWM tasks. TIME further improves TwoRoom and Reacher; on Reacher, both variants succeed in every evaluated execution, so the reduction reflects earlier arrival. The horizon study shows that objective choice and prediction horizon interact, while the DINO-WM extension provides preliminary evidence beyond LeWM: on Wall, TIME reaches success in 19.0 steps versus 26.1 for the same-case Free check, whereas PushT remains a close comparison.

Together, these results support a deployment-time objective that makes fuller use of a world model's predicted trajectory: the horizon defines what can be considered, while the objective determines which arrivals matter. The evidence concerns physical interaction steps; latent mismatch, model error, finite local optimization and planning latency leave minimum-time physical arrival and total wall-clock completion time outside the current evaluation.

AI USE STATEMENT

Generative AI tools were used solely for language editing and polishing of the Introduction, Related Work, and Experiments sections, with the purpose of improving grammar, clarity, and readability. The scientific content and contributions of the paper remain the work of the authors. All AI-assisted edits were reviewed and verified by the authors. The authors take full responsibility for the final content of the paper.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems 30*, 2017. URL https://papers.nips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html.

Laurent Bako, Dulin Chen, and Stéphane Lecoeuche. A numerical solution to the minimum-time control problem for linear discrete-time systems, 2011. URL https://arxiv.org/abs/1109.3772v1.

Saksham Bansal, Om Naphade, Chayan Aggarwal, and Vrishin M. LePlanner: An iterative amortized controller for world models. *arXiv preprint arXiv:2609.13845*, 2026. URL https://arxiv.org/abs/2609.13845.

Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134:19–67, 2005. doi: 10.1007/s10479-005-5724-z. URL https://link.springer.com/article/10.1007/s10479-005-5724-z.

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

Kazuya Echigo, Abhishek Cauligi, and Behçet Açıkmeşe. Expected time-optimal control: a particle MPC-based approach via sequential convex programming, 2024. URL https://arxiv.org/abs/2404.16269v1.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion, 2017. URL https://arxiv.org/abs/1610.00696. ICRA 2017.

David Ha and J"urgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, 2018. URL https://papers.nips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2555–2565, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

Nick Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/cf73d57b6dcda32b293df7c2d5341f49-Abstract-Conference.html.

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

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, 2026. URL https://arxiv.org/abs/2608.14125.

Yewon Lee, Philip Huang, Krishna Murthy Jatavallabhula, Andrew Z. Li, Fabian Damken, Eric Heiden, Kevin Smith, Derek Nowrouzezahrai, Fabio Ramos, and Florian Shkurti. STAMP: Differentiable task and motion planning via stein variational gradient descent, 2023. URL https://arxiv.org/abs/2310.01775v1. Version 1, Appendix A-D.

Karen Leung, Nikos Aréchiga, and Marco Pavone. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *The International Journal of Robotics Research*, 42(6), 2023. doi: 10.1177/02783649221082115. URL https://arxiv.org/abs/2008.00097.

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, 2026. URL https://arxiv.org/abs/2608.01926.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017. URL https://arxiv.org/abs/1711.05101.

Lucas Maes, Quentin Le Lidec, Luiz Facury, Nassim Massaudi, Ayush Chaurasia, Francesco Capuano, Richard Gao, Taj Gillin, Dan Haramati, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel: A platform for reproducible world modeling research and evaluation, 2026a. URL https://arxiv.org/abs/2605.21800v1.

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

Noushin Mehdipour, Cristian-Ioan Vasile, and Calin Belta. Specifying user preferences using weighted signal temporal logic, 2020. URL https://arxiv.org/abs/2010.00752.

Phu Pham and Aniket Bera. Latent energy action planning with world models, 2026. URL https://arxiv.org/abs/2609.03294v1.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning, 2020. URL https://arxiv.org/abs/2008.06389.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal Difference Models: Model-Free Deep RL for Model-Based Control, 2018. URL https://arxiv.org/abs/1802.09081.

Michael Psenka, Michael Rabbat, Aditi Krishnapriyan, Yann LeCun, and Amir Bar. Parallel stochastic gradient-based planning for world models, 2026. URL https://arxiv.org/abs/2602.00475.

Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M. Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A. Seshia. Model predictive control with signal temporal logic specifications. In *Proceedings of the 53rd IEEE Conference on Decision and Control*, pp. 81–87, 2014. URL https://people.eecs.berkeley.edu/~sseshia/pubs/b2hd-raman-cdc14.html.

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

Armin Sommer and Jannik Schilling. Reinforced Planning with Latent World Models, 2026. URL https://arxiv.org/abs/2608.18669.

Kyle Stachowicz and Evangelos A. Theodorou. Optimal-horizon model-predictive control with differential dynamic programming, 2021. URL https://arxiv.org/abs/2111.09207.

Wei Sun, Evangelos Theodorou, and Panagiotis Tsiotras. Model based reinforcement learning with final time horizon optimization, 2015. URL https://arxiv.org/abs/1509.01186.

Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information theoretic model predictive control: Theory and applications to autonomous driving, 2017. URL https://arxiv.org/abs/1707.02342.

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, pp. 79115–79135, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

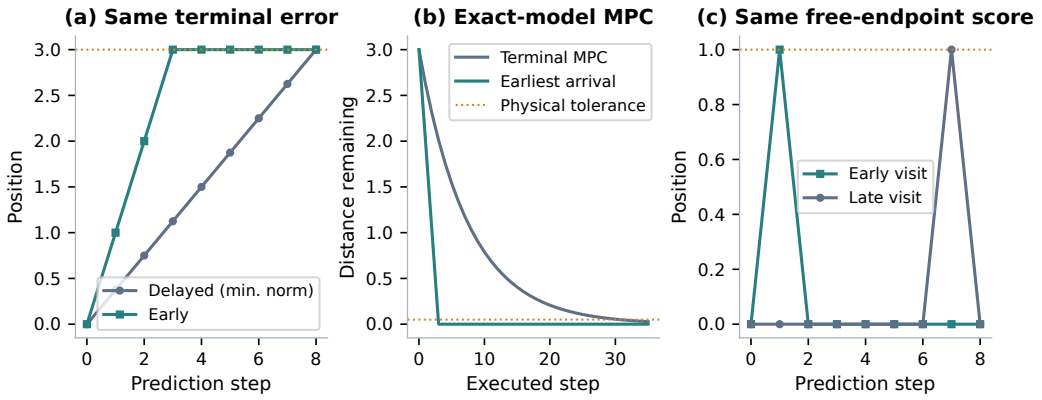

Figure 5: **Why arrival order matters.** Analytic integrator examples, not visual-task results. (a) Plans can share zero terminal error and arrive at different times. (b) Repeated minimum-norm terminal planning postpones arrival under exact dynamics; other terminal solution rules can behave differently. (c) Two prescribed plans share terminal and free-endpoint scores but visit the goal at steps 1 and 7; TIME distinguishes their order. The constructions are specified below.

A Derivations and proof boundaries

A.1 Entropy representation and action derivatives

For $Z = \sum_h e^{-b_h/\tau}$ and $w_h^* = e^{-b_h/\tau}/Z$, any $w \in \Delta_H$ satisfies

$$\sum_h w_h b_h + \tau \sum_h w_h \log w_h = -\tau \log Z + \tau D_{\mathrm{KL}}(w \| w^*).$$

All entries of w^* are positive when costs are finite, so the nonnegative KL divergence is well defined with $0 \log 0 = 0$. Equality holds at $w = w^*$. Differentiating log-sum-exp gives $\nabla F = \sum_h w_h^* \nabla b_h$; no separate derivative of the minimizing weights is needed. Since $q'_\kappa(x) = \mathrm{sigmoid}(x/\kappa)$ and S is held constant within a call, the chain rule yields Equation (5). Causality supplies $\nabla_{a_j} C_h = 0$ for $h < j$. For a terminal-only cost $J(A) = \phi(c_H(A))$, the same chain rule gives $\nabla_{a_j} J = \phi'(c_H) \nabla_{a_j} c_H$ for every $j \le H$; terminal matching therefore provides no endpoint-wise attenuation of actions after an earlier predicted match. The arrival weights alter these coefficients through the sum in Equation (5); they do not remove later actions from the plan or from the rollout. These derivatives refer to the full action sequence. During coarse optimization, if $A = Lu + d$ is the affine interpolation from knot coordinates u to bounded action coordinates before projection, the chain rule gives $\nabla_u F = L^\top \nabla_A F$. A knot can influence multiple earlier and later actions, so the endpoint–action triangular structure is not a triangular statement about knot indices.

A.2 Time tilt and gradient concentration

Proof of Proposition 1. Dividing $e^{-(p_h + \alpha h)/\tau}$ by its sum gives the tilt identity. Let $\bar{h} = \sum_h h w_h^\alpha$. Direct differentiation gives $\partial_\alpha w_h^\alpha = -(h - \bar{h}) w_h^\alpha / \tau$. Multiplication by h and summation give $\partial_\alpha \bar{h} = -(\sum_h h^2 w_h^\alpha - \bar{h}^2)/\tau$. □

If $b_h - b_{\min} \ge \Delta$ for each h in a set J of size m, then

$$\sum_{h \in J} w_h = \frac{\sum_{h \in J} e^{-(b_h - b_{\min})/\tau}}{\sum_i e^{-(b_i - b_{\min})/\tau}} \le m e^{-\Delta/\tau},$$

because the denominator is at least one. If each corresponding branch gradient has norm at most G, the norm of their weighted contribution is at most $G m e^{-\Delta/\tau}$. A small temperature can therefore remove useful gradient influence before those branches have been optimized. This is a statement

about weights and bounded gradients, not about which physical plans are attainable. For twice differentiable branches,

$$\nabla^2 F = \sum_h w_h \nabla^2 b_h - \frac{1}{\tau}\left(\sum_h w_h \nabla b_h \nabla b_h^\top - \nabla F \nabla F^\top\right).$$

The negative covariance term shows that smooth aggregation does not make the planning problem convex, even when individual branches are locally convex.

Reoptimizing actions as the time weight changes. Fix the input, action domain, horizon, normalization scale, penalty and smoothing parameters for one planning call. For $\alpha_2 > \alpha_1 \geq 0$, suppose A_i satisfies $F_{\tau,\alpha_i}(A_i) \leq \inf_A F_{\tau,\alpha_i}(A) + \eta_i$. Let w_i be its Gibbs weights at α_i, and put $m_i = \sum_h h w_{i,h}$. Then

$$m_2 - m_1 \leq \frac{\eta_1 + \eta_2}{\alpha_2 - \alpha_1}. \tag{9}$$

In particular, exact global solutions have $m_2 \leq m_1$, even if the selected action sequences differ or global solutions are not unique. To see this, write the joint variational objective as $J_\alpha(A, w) = \alpha m(w) + G(A, w)$, where $G = \sum_h w_h p_h(A) + \tau \sum_h w_h \log w_h$ does not depend on α. Equation (4) makes (A_i, w_i) an η_i-optimal joint solution. Comparing it with the other pair gives

$$\alpha_1 m_1 + G_1 \leq \alpha_1 m_2 + G_2 + \eta_1, \qquad \alpha_2 m_2 + G_2 \leq \alpha_2 m_1 + G_1 + \eta_2.$$

Adding and rearranging proves Equation (9). This consequence concerns the weighted predicted index, not the first proxy threshold crossing or native first success. Our local solver supplies no known η_i, and different MPC calls change both observations and normalization, so the bound is not a monotonicity guarantee for the implemented controller.

A.3 Penalty on the discrete prediction grid

Proof of Proposition 2. For each $h < h^*$, continuity on the compact action set implies that the minimum of c_h is attained. By definition of h^*, this minimum is strictly larger than ϵ, establishing $\delta_h > 0$. With the stated penalty, $h + \rho[c_h(A) - \epsilon]_+ \geq h + \rho\delta_h > h^*$ for all A and $h < h^*$. For $h \geq h^*$, the branch value is at least $h \geq h^*$. Some A^* attains $c_{h^*}(A^*) \leq \epsilon$ and hence branch value h^*. Minimizing over both A and the finite endpoint set proves $\min_A P(A) = h^*$.

The inequalities $[x]_+ \leq q_\kappa(x) \leq [x]_+ + \kappa \log 2$ imply that the minimum of the smoothed branch values is between $P(A)$ and $P(A) + \rho\kappa \log 2$. For any vector b, $\min_h b_h - \tau \log H \leq -\tau \log \sum_h e^{-b_h/\tau} \leq \min_h b_h$. Combining the two statements proves Equation (8). □

The penalty result gives the optimal value of the nonsmoothed global problem. If no endpoint is feasible, h^* and this particular penalty threshold are undefined. These cases are included in physical evaluation as usual.

Smoothing can remove exact feasibility. Consider $H = 2$, $A = (u, v) \in [0, 1]^2$, $\rho = 1$, $\epsilon = 0$, $c_1(A) = 3 - u$ and $c_2(A) = u^2$. The second coordinate is unused. Here $h^* = 2$, $\delta_1 = 2$, and the strict penalty condition $1 > 1/2$ holds. The only feasible actions have $u = 0$, where $P = 2$. Nevertheless, for any $\tau, \kappa > 0$, the right derivative of the smoothed objective there is

$$\partial_u F_{\tau,1}|_{u=0} = -w_1^* \,\mathrm{sigmoid}(3/\kappa) < 0,$$

because $\partial_u c_2 = 0$ at zero. Thus every global minimizer of the smooth objective has $u > 0$ and violates both endpoint thresholds. For example, with $\tau = 1$, $\kappa = .1$, the infeasible action $u = 1/8$ has a lower smooth value than every feasible action. This continuous-cost construction is a counterexample to a general feasibility guarantee, not a measured model failure. A small value approximation gap alone does not preserve a hard threshold.

A.4 Uniform perturbations and planning regret

Suppose $\sup_{A,h} |\widetilde{c}_h(A) - c_h(A)| \leq \delta$, and use the same S, ϵ, time weights and smoothing parameters for both objectives. Because $0 < q'_\kappa < 1$, corresponding branches differ by at most $\rho\delta$. If vectors

b and $\widetilde{b}$ differ by at most d in max norm, $e^{-d/\tau}\sum e^{-b_h/\tau} \le \sum e^{-\widetilde{b}_h/\tau} \le e^{d/\tau}\sum e^{-b_h/\tau}$. Taking $-\tau\log$ proves $|\widetilde{F} - F| \le d$. If $\widetilde{F}(\widehat{A}) \le \inf_A \widetilde{F}(A) + \eta$, then

$$F(\widehat{A}) \le \widetilde{F}(\widehat{A}) + \rho\delta \le \inf_A \widetilde{F}(A) + \eta + \rho\delta \le \inf_A F(A) + \eta + 2\rho\delta.$$

Our finite optimizer supplies neither η nor a uniform δ for the learned model. The proposition is therefore used to expose dependence on model error and penalty scaling, never to claim measured real-world regret.

Endpoint allocation can be more sensitive than objective value. At a fixed action sequence, put $e_h = \widetilde{b}_h - b_h$, so $|e_h| \le \rho\delta$. Normalization cancels in pairwise log odds:

$$\log\frac{\widetilde{w}_h}{\widetilde{w}_j} - \log\frac{w_h}{w_j} = -\frac{e_h - e_j}{\tau}, \qquad \left|\log\frac{\widetilde{w}_h}{\widetilde{w}_j} - \log\frac{w_h}{w_j}\right| \le \frac{2\rho\delta}{\tau}.$$

For illustration only, a hypothetical normalized-cost error bound $\delta = .01$ with the primary $\rho = 30$ and $\tau = .25$ gives a value bound of .3 but permits an endpoint-odds change by a factor up to $e^{2.4} \approx 11.0$. This is not an estimated error level or an observed weight change. The expression concerns exact positive weights; numerical underflow is discussed in Appendix C. Moreover, a uniform bound on a differentiable cost does not bound its derivative: the perturbation $\delta\sin(\omega a)$ has amplitude at most δ while its derivative amplitude is $\delta\omega$. Thus the value-regret argument does not establish a gradient-error bound or optimizer stability.

A.5 Why terminal matching can postpone arrival

Consider $x_{t+1} = x_t + a_t$, $|a_t| \le 1$, $x_0 = 0$ and a target $g > 0$ with $g \le H$ and integer $H > 1$. A constant sequence $a_t = g/H$ reaches the exact target at H; a sequence of unit actions followed by a fractional final action and zeros reaches it at $\lceil g\rceil$. Both attain zero terminal mismatch. Thus terminal mismatch alone cannot distinguish their arrival times. It does not force the later plan, which is why this example is not a theorem that every terminal planner is slow. The constant sequence is, however, the unique minimum squared-action-norm sequence among exact terminal solutions: Cauchy–Schwarz gives $\sum a_t^2 \ge (\sum a_t)^2/H = g^2/H$, with equality only for equal actions. If this terminal solution is recomputed after every action, the remaining distance satisfies $d_{k+1} = (1 - 1/H)d_k$. For strict success $d_k < r$, with $0 < r < g$, the first successful step is $\lfloor \log(r/g)/\log(1 - 1/H)\rfloor + 1$. For the nonstrict rule $d_k \le r$, it is instead $\lceil \log(r/g)/\log(1 - 1/H)\rceil$; the rules differ when an iterate lands exactly on the threshold. Figure 5 uses the strict rule with $H = 8$, $g = 3$ and $r = .05$, giving 31 steps. A preference for an early feasible endpoint can distinguish this behavior without changing the dynamics model.

Endpoint flexibility does not determine arrival order. Figure 5c uses the same exact integrator, now with target $g = 1$ and $H = 8$. The action sequences $(1, -1, 0, 0, 0, 0, 0, 0)$ and $(0, 0, 0, 0, 0, 0, 1, -1)$ first visit the goal at steps 1 and 7. Both obey $|a_t| \le 1$ and end at zero. Each has one zero mismatch and seven unit mismatches, with $S = 1$. Consequently terminal and free-endpoint scores agree exactly between them. With the primary objective parameters, TIME scores the early and late plans approximately 1.0020 and 7.0020. These are prescribed feasible plans, not optimizer solutions; suffixes after first arrival illustrate predictions, not observations after a stopping event.

More generally, hold all other endpoint penalties fixed and swap two costs $p_{\mathrm{lo}} < p_{\mathrm{hi}}$ between indices $i < j$. Write Z_E, Z_L for the exponential sums when the lower penalty is earlier or later. Then

$$Z_E - Z_L = \left(e^{-p_{\mathrm{lo}}/\tau} - e^{-p_{\mathrm{hi}}/\tau}\right)\left(e^{-\alpha i/\tau} - e^{-\alpha j/\tau}\right) > 0 \quad \text{for } \alpha > 0.$$

Hence $F_E < F_L$, whereas $\alpha = 0$ gives equality. This isolates temporal ordering from the multiset of mismatches. It does not assert that an arbitrary permutation is dynamically attainable, or that an optimized TIME plan will arrive earlier in a learned environment; the two integrator plans establish attainability only for this example.

Arrival between native prediction endpoints. Endpoint sampling introduces a separate limitation even with an exact model. For the integrator $x_{t+1} = x_t + u_t$, $|u_t| \leq 1$, take $x_0 = 0$, goal $g = 1$ and native predictions every $B = 5$ physical actions. Compare the prescribed sequence $(1, -1, 0, \ldots, 0)$ with an all-zero sequence over $BH = 40$ actions. Both have $x_{Bh} = 0$ for all $h = 1, \ldots, H$, so with an identity encoder and $S = 1$ they have identical sampled mismatch vectors $c_h = 1$ and identical Terminal, Free and TIME scores. Yet for strict physical success $|x_t - 1| < .05$, the first sequence succeeds at action 1 and the second never succeeds. The first sequence's suffix describes the hypothetical unstopped rollout, not execution after success. Thus exact endpoint costs need not identify physical first arrival. The construction holds for any $B \geq 2$; with $B = 1$ the visit is sampled. It demonstrates a sampling limit, not optimizer behavior.

B Relation to existing temporal objectives

LePlanner trains an iterative controller with an arrival-and-hold loss (Bansal et al., 2026),

$$L_q(A) = d_q(A) + \frac{\lambda_h}{H - q} \sum_{j=q+1}^{H} d_j(A),$$

where the second term is zero at $q = H$. The index q is the temporal offset of a goal sampled from an offline trajectory and supplies training supervision. TIME evaluates candidate endpoints during online action optimization, requiring neither that offset nor an additional trained controller. This comparison distinguishes objectives and training requirements; we do not empirically compare performance against LePlanner.

Weighted sums of unsquared state norms, $\sum_h \omega_h \|x_h\|_2$, provide minimum-time surrogates in discrete linear control (Bako et al., 2011) and particle MPC with sequential convex programming (Echigo et al., 2024). Their recovery arguments rely on assumptions about dynamics and target equilibria that do not automatically extend to nonlinear latent models. TIME evaluates first physical success without requiring a feasible holding suffix. Weighted unsquared latent-norm adaptations remain a relevant comparison outside the present experiments.

For the predicate $\epsilon - c_h(A) \geq 0$, temporal eventuality has robustness $\max_h(\epsilon - c_h) = \epsilon - \min_h c_h$. Free-endpoint scoring is thus closely related to established reachability objectives. STLCG uses smooth temporal extrema (Leung et al., 2023), weighted temporal logic expresses satisfaction-time preferences (Mehdipour et al., 2020), and STAMP applies smooth temporal aggregation to differentiable planning (Lee et al., 2023). These connections, together with Equation (7), place TIME within established temporal aggregation rather than treating its smooth minimum or discounted form as new operators. The contribution is the deployment-time composition and its analysis in the visual-world-model setting: normalized latent mismatch, elapsed model-block cost, smooth endpoint aggregation, and a controlled separation of endpoint flexibility from explicit time preference. The experiments evaluate first physical success under receding-horizon replanning.

C Numerical planning procedure

The objective controls use a shared projected optimizer with optional coarse-to-fine coordinates and local refinement. Let N denote the number of starts, K the Adam iteration budget, and $P_{\max}$ the refinement cap in objective–gradient evaluations. In model-standardized action coordinates, write the bounds as ℓ, u, the center as $m = (\ell + u)/2$, and the radius as $r = (u - \ell)/2$. Optimization uses coordinates $V \in [-1, 1]$; the model receives $A = m + r \odot V$ after any expansion of coarse coordinates.

One planning call. Besides the released model and settings, whose parameters remain fixed during planning, the procedure receives the current and goal images, a case-and-decision seed, and the previous plan's shifted suffix.

1. Encode the current and goal images once, holding both embeddings and the normalization scale S fixed throughout the call.
2. Draw N Gaussian candidates in standardized action coordinates and replace the first by zero. When a previous plan is available, add its shifted suffix to the first half of the population.

Set the first candidate's corresponding prefix exactly to that suffix, with its remaining block initialized at standardized zero. Project the resulting population into the bounded action domain.

3. For a coarse configuration, draw a separate population of four Gaussian physical-action knots per candidate. Map and project them into the same unit box, then interpolate linearly across all HB physical actions. This knot draw follows the full-population draw in the same random-number stream. With a warm start, replace the first half of the knot population by compressed full candidates. Otherwise, the first coarse candidate represents projected standardized zero. Compression can change the trajectory represented by the warm start.
4. At each of K iterations, expand the coordinates, predict every native endpoint, and evaluate the condition's objective for every candidate. Retain the lowest objective value encountered and its full action sequence. Differentiate the sum of candidate objectives, apply an Adam update, and project the coordinates to $[-1, 1]$. At the declared coarse-to-full transition, expand the current coordinates and restart Adam with an empty moment state.
5. Evaluate the population after the final Adam update and update the incumbent. When refinement is enabled, initialize bounded L-BFGS-B from that incumbent and evaluate at most $P_{\max}$ objective–gradient pairs. Every evaluated trial, including a line-search trial, can update the incumbent. Early solver termination can leave part of the evaluation budget unused.
6. Recompute endpoint costs for diagnostics and return the full H-block plan. Execute its first block, checking native success after each physical action. The next call uses the remaining $H - 1$ blocks as its warm-start suffix.

Adam uses learning rate .12, the default coefficients $(\beta_1, \beta_2) = (.9, .999)$, and numerical epsilon 10^{-8}. L-BFGS-B uses function tolerance 10^{-7} and at most eight line-search steps, subject to the stricter gradient-evaluation cap. Both methods perform local optimization and provide no global optimality certificate.

Numerical precision. Checkpoint parameters, encoder outputs, predicted embeddings, costs, action coordinates and gradients use float32. The model parameters remain fixed while gradients propagate through predictions to actions. TensorFloat-32 is disabled for matrix multiplication and cuDNN convolution, and no mixed-precision autocast is used. The SciPy L-BFGS-B interface receives objective values and gradients converted to double precision after the float32 model computation. This conversion preserves interface compatibility without making the underlying optimization a double-precision solve.

Softplus and log-sum-exp use numerically stable library implementations. Very small endpoint weights can nevertheless underflow to zero in float32. Positivity of all finite-cost Gibbs weights is therefore an exact-arithmetic property; individual endpoints can supply numerically zero gradients in the implementation.

D Evaluation protocol and computation

Cases, inputs and success. Each primary task has 16 development episodes and 64 test episodes. Complete episodes used in earlier local planning studies are excluded. Compared methods share recorded starts and goals, a goal offset of 50 physical actions, native action bounds, and a 100-action execution budget. The primary planning seeds are 42, 123 and 2026. The goal image and simulator state come from the same recorded index. Action standardization uses all finite action rows in the released task dataset, including rows outside development; recorded future actions are never planner inputs. Pretraining overlap with evaluation episodes is unknown. Recorded states and stable environment seeds restore each simulation.

The evaluator checks success before acting and after every physical action, using native state predicates outside the image-only planner. TwoRoom requires agent–target distance below 16 pixels. PushT requires a Euclidean difference below 20 pixels between concatenated pusher and block positions, plus wrapped block-angle difference below $\pi/9$. Its target is the restored recorded state, rather than the renderer's separately configured green T marker. Reacher requires every joint coordinate to be within $.05$ radians of its target. Cube tests object-position error below $.04$ metres, without an

Table 4: Development grid for the three core objectives: mean budgeted steps / success percent. Each cell uses 16 development episodes and seed 42. These cases are excluded from testing; selection uses the TIME rows only.

Solver	Objective	TwoRoom	PushT	Reacher	Cube
Full Adam	Terminal	65.44 / 50.0	91.94 / 12.5	100.00 / 0.0	64.06 / 37.5
Full Adam	Free endpoint	27.56 / 93.8	66.62 / 50.0	20.44 / 100.0	62.00 / 43.8
Full Adam	TIME	14.00 / 100.0	42.88 / 75.0	16.31 / 100.0	57.69 / 43.8
Coarse Adam	Terminal	78.69 / 37.5	89.62 / 18.8	89.81 / 18.8	53.06 / 50.0
Coarse Adam	Free endpoint	18.69 / 100.0	59.12 / 56.2	22.12 / 100.0	63.38 / 37.5
Coarse Adam	TIME	11.38 / 100.0	49.19 / 68.8	11.94 / 100.0	57.62 / 43.8
Coarse + polish	Terminal	70.56 / 43.8	90.06 / 18.8	90.81 / 12.5	63.44 / 43.8
Coarse + polish	Free endpoint	18.56 / 100.0	57.88 / 62.5	20.31 / 100.0	46.69 / 56.2
Coarse + polish	TIME	12.31 / 100.0	50.69 / 62.5	14.19 / 100.0	46.00 / 56.2

orientation criterion. Initially successful episode counts are 1, 0, 0 and 16 out of 64, respectively, and retain zero steps. We report active cases separately to distinguish these initial successes. Native predicates are never passed to the optimizer as latent thresholds.

Shared-solver selection. The three candidate configurations use 96 starts and Adam learning rate .12 in box-normalized coordinates. Full Adam uses 60 iterations; coarse Adam uses 60 iterations, with the first 30 optimizing four interpolated action knots. Coarse Adam with refinement uses 40 iterations, the first 20 coarse, followed by at most 16 L-BFGS-B gradient evaluations. Selection minimizes TIME's mean budgeted steps averaged equally across tasks, with ties resolved by higher success and then configuration name. Coarse Adam with refinement is selected (30.80 steps, 79.7% success), ahead of coarse Adam (32.53, 78.1%) and full Adam (32.72, 79.7%). Terminal, Free and TIME then share this configuration.

Table 4 reports all three objectives under every candidate configuration. Under the same equal-task rule, Free yields 44.15625, 40.828125 and 35.859375 steps for full Adam, coarse Adam and coarse Adam with refinement, respectively. Thus Free selects the same configuration among the tested candidates. This descriptive check preserves the frozen selection; it is not exhaustive independent tuning of each objective.

External baselines. Each literature baseline has six development candidates: horizons $3, 5, 8$ and two budgets. Selection is per task and family, ordered by mean budgeted steps, decreasing success, exclusive median solver latency and configuration name. Table 5 gives the frozen choices. CEM and iCEM use the installed sampling implementations; iCEM includes colored temporal noise and elite retention. MPPI* averages only the top-k trajectories at temperature .5, distinguishing it from unrestricted path-integral formulations. GD uses AdamW with learning rate .1 and weight decay .01 (Loshchilov & Hutter, 2017); candidates are projected and rescored before selection. Its rollout objective, also used by CEM, iCEM and MPPI*, is the native terminal squared latent distance C_H from Equation (1), without division by S or a soft threshold. AdamW additionally applies decoupled weight decay to the optimized action coordinates. Table 6 distinguishes this system from the Terminal objective reference on Reacher. GRASP optimizes relaxed state–action trajectories and synchronizes to native rollouts. These systems differ in search mechanics from the shared objective controls. Equal iteration counts do not imply equal work.

Both terminal costs are strictly increasing in C_H for a fixed planning call, so they share the same exact minimizers over the same action domain at the same H. Their gradient scales differ, and finite optimization need not produce the same plan. The present horizon study does not separately vary this loss transform or optimizer. Consequently, the GD–Terminal gap does not isolate either factor or deferred arrival alone.

Prediction-horizon comparison. The declared secondary horizons are $3, 5, 12$. For each objective we combine their 32-episode, seed-42 results with the matching 32-episode seed-42 prefix at $H = 8$. Episode identities, restored starts, goals, $B = 5$, the 100-action budget, 96 starts, 40 Adam iterations,

Table 5: External configurations selected on development data. GRASP's single sample is a relaxed state–action trajectory; iterations have different costs across solvers.

Task	Method	H	Starts/samples	Iterations	Elites
TwoRoom	CEM	3	128	10	16
TwoRoom	iCEM	5	300	30	30
TwoRoom	MPPI*	3	128	10	16
TwoRoom	GD	5	100	30	–
TwoRoom	GRASP	5	1	750	–
PushT	CEM	3	128	10	16
PushT	iCEM	3	128	10	16
PushT	MPPI*	3	128	10	16
PushT	GD	3	100	30	–
PushT	GRASP	5	1	750	–
Reacher	CEM	3	300	30	30
Reacher	iCEM	3	128	10	16
Reacher	MPPI*	3	128	10	16
Reacher	GD	3	16	30	–
Reacher	GRASP	5	1	750	–
Cube	CEM	5	300	30	30
Cube	iCEM	5	128	10	16
Cube	MPPI*	8	128	10	16
Cube	GD	3	100	30	–
Cube	GRASP	3	1	200	–

Table 6: **Reacher GD and Terminal are different configurations.** Both execute one $B = 5$ block, shift the returned full plan by one block, and use the same physical success check and 100-action budget. The table describes the measured systems; it does not attribute their performance gap to one factor.

Field	GD (external system)	Terminal (objective reference)
Objective	C_H	$\rho q_\kappa(C_H/S - \epsilon)$
Prediction	$H = 3$: 15 physical actions	$H = 8$: 40 physical actions
Optimizer	AdamW, learning rate .1, weight decay .01	Adam, learning rate .12; bounded L-BFGS-B polish
Budget	16 starts, 30 iterations	96 starts, 40 Adam iterations (20 coarse); at most 16 polish evaluations
Coordinates	Full standardized actions	Four interpolated knots, then full coordinates, each in $[-1, 1]$ action-box units
Bounds	Project final candidates before rescoring	Project every Adam update; bounded polishing
Initialization	Standardized-zero center plus Gaussian perturbations; first candidate unperturbed	Bounded Gaussian knot population; one projected standardized-zero candidate
Warm start	Shifted suffix plus zero tail; perturb all but the first candidate	Shifted suffix injected into half the population; coarse knots obtained by compression
Retention	Rescore and choose among projected final candidates	Retain best visited candidate under the condition's own objective, including polishing

the coarse schedule and polishing cap remain identical. Only H changes. Larger H increases rollout length and full action dimension; the experiment fixes solver settings, not total computation. No test-horizon outcome is used to revise the primary configuration. Table 9 reports all four tasks and all four horizons, including the different ordering on Cube. Figure 6 shows the exploratory marginal episode-bootstrap intervals from this single-seed cohort.

Physical quality and compute budgets. Figure 4 uses the first 16 fixed test episodes per task, seed 42 and $H = 8$. Half- and quarter-budget TIME configurations jointly scale starts, Adam iterations, coarse iterations and the refinement cap. External reference/fast budgets use 300/128 candidates

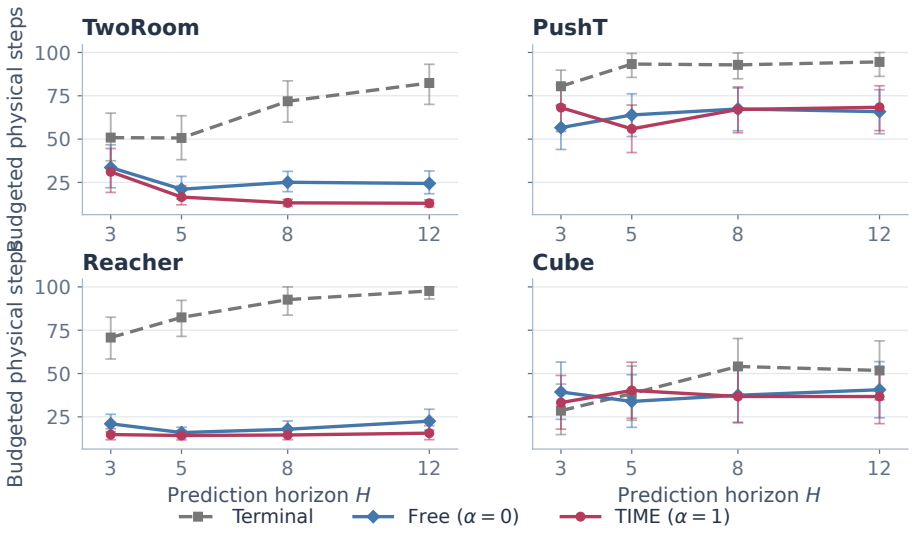

Figure 6: **Objective effects across prediction horizons.** Matched 32 episodes per task and seed 42; each H uses the same solver and $B = 5$ for all three objectives. Points and whiskers show marginal 95% episode-bootstrap intervals. Lines connect tested settings, not an interpolated optimum. A fixed iteration budget costs more model transitions at larger H; this is not an equal-compute comparison.

and 30/10 iterations for CEM, iCEM and MPPI*, with 30/16 elites; GD uses 100/16 starts and 30 iterations; GRASP uses 750/200 iterations. These conditions were declared before primary execution and do not replace the main development selections. Explicit endpoint search solves a terminal-penalty problem separately at every $h \in \{1, \dots, H\}$ and selects the returned candidate minimizing $h + \rho q_\kappa(c_h - \epsilon)$. Its split/full versions divide 96 starts across endpoints or use 96 at each endpoint, with the same iteration/polishing limits. The selected prefix is padded with bounded standardized zero actions to return the full window. This is finite local search, not a global enumeration certificate.

Latency uses three noninitial development inputs in protocol order. Each method/input receives one warmup and three timed initial solves, without a warm-start plan. CUDA synchronization brackets each solve, with no concurrent GPU evaluation. Timing includes encoding and excludes preprocessing, model loading and physics. The pooled median of nine solves is reported; these repetitions do not increase the physical-performance sample size. Initial-plan latency measures the first solve rather than end-to-end closed-loop completion. Figure 7 shows the matching success-rate view for these same declared settings. On Reacher, quarter-budget TIME uses 1.28 seconds with 14.75 mean physical steps, versus 15.81 for default TIME; the paired difference is -1.06 $[-3.69, 1.56]$. On Cube, the quarter budget increases the point estimate by 11.56 steps and reduces success by 12.5 percentage points. These exploratory results illustrate task-dependent sensitivity to the optimization budget.

Statistics. Primary inference first averages three seeds within each episode. Marginal 95% bootstrap intervals use 10,000 episode resamples; contrasts preserve paired episodes. Two-sided paired sign-permutation tests use 100,000 draws. The 20 TIME–external comparisons form one Holm family, and the eight TIME–Free and Free–Terminal comparisons another. The permutation null assumes paired-label exchangeability, not merely equal means. An all-success empirical-bootstrap interval does not certify a population success probability of one. Compute-budget comparisons use paired single-seed episode intervals and remain exploratory. All displayed matrices are complete; failed executions receive 100 steps and are not dropped. Initially successful cases receive zero. For budget K and first-success time T, including $T = \infty$ on failure,

$$\mathbb{E}[\min(T, K)] = \sum_{t=0}^{K-1} \Pr(T > t).$$

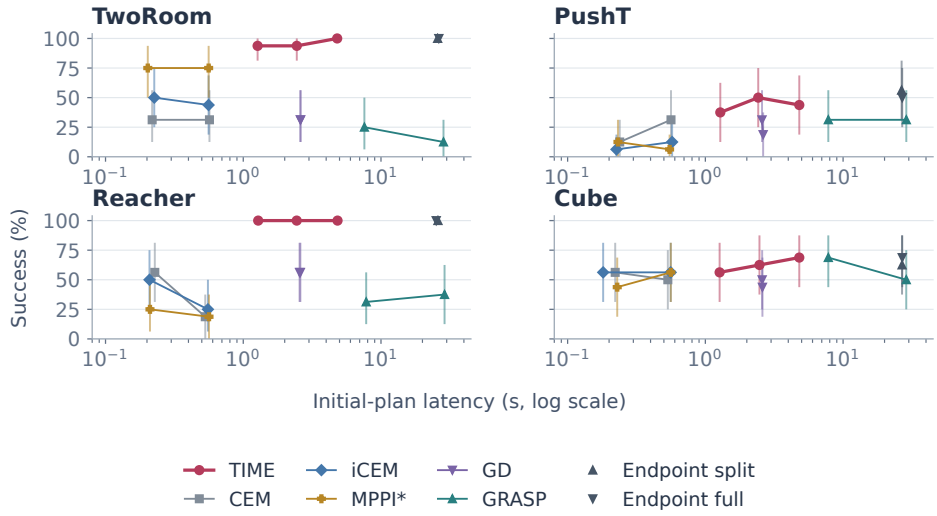

Figure 7: **Success coverage versus initial-plan latency.** Same declared $H = 8$ compute settings, 16 test cases per task and seed 42 as Figure 4. Whiskers are marginal 95% episode-bootstrap intervals; latency is the median of nine initial solves. Initial-plan latency is not closed-loop completion time.

Thus budgeted steps combine success coverage and timing; they are not a conditional mean among successful executions.

E Model and reproducibility details

The four frozen LeWM checkpoints use float32 parameters in evaluation mode. Each encoder is a 12-layer vision transformer with width 192, three attention heads and 14×14 patches. The projected class token provides the 192-dimensional planning representation. The six-layer autoregressive predictor uses at most three embedding frames, starting with the current embedding and then its own predictions. Preprocessing uses RGB 224×224 images, bilinear resizing with antialiasing, and normalization with channel means $(.485, .456, .406)$ and standard deviations $(.229, .224, .225)$. Each model action concatenates five physical actions; the physical action dimensions for TwoRoom, PushT, Reacher and Cube are 2, 2, 2 and 5, respectively.

Experiments run on one NVIDIA GeForce RTX 5090 (32 GB), using Python 3.12, PyTorch 2.8.0 with CUDA 12.8, and SciPy 1.18.1. The archived software includes `stable-worldmodel` 0.1.1, `stable-pretraining` 0.1.7, `dm-control` 1.0.40, MuJoCo 3.8.0, OGBench 1.2.1 and Pymunk 7.3.0 (Maes et al., 2026a). Checkpoint, source and input hashes accompany episode manifests, recorded physical actions and first-success flags. Replaying saved actions verifies physical accounting in the same simulator; it does not assess model accuracy or reproduce the optimization process.

CUDA interpolation-gradient reductions can be nondeterministic. A separate reoptimization check produced 56 rather than 17 actions for one TwoRoom Free case, with success unchanged. This observation identifies possible numerical sensitivity without estimating its frequency or its effect on aggregate rankings. The experiments and checks use one host, so portability to other hardware remains untested. Compiling the document does not rerun planning or physics.

F Primary estimates and paired comparisons

The following tables retain all four tasks and eight displayed methods: five literature baselines and three objective conditions. Each method/task cell uses 64 episodes and three planning seeds. Active-case columns exclude only episodes successful before acting, identically across methods. All intervals are marginal 95% episode-bootstrap intervals. Differences are first-named minus second-named: negative steps and positive success percentage points favor the first method. Holm correction uses

the two separate families specified in Appendix D; positive values below 10^{-4} are displayed as < 0.0001.

Table 7: Core objective contrasts under the shared solver.

Task	Contrast	Δ steps [95% CI]	Δ SR (pp) [95% CI]	Holm p
TwoRoom	TIME − Free	-9.30 [-13.06, -6.15]	+1.6 [-1.6, +5.7]	< 0.0001
TwoRoom	Free − Terminal	-51.65 [-56.85, -46.22]	+53.1 [+45.3, +61.5]	< 0.0001
PushT	TIME − Free	-5.06 [-11.68, +1.64]	+1.6 [-8.3, +10.9]	0.2816
PushT	Free − Terminal	-25.72 [-32.31, -19.33]	+41.1 [+32.8, +49.5]	< 0.0001
Reacher	TIME − Free	-4.73 [-6.88, -2.67]	+0.0 [+0.0, +0.0]	0.0001
Reacher	Free − Terminal	-76.17 [-79.16, -72.98]	+92.7 [+88.5, +96.4]	< 0.0001
Cube	TIME − Free	+3.70 [-2.84, +11.12]	-4.2 [-12.0, +3.1]	0.3094
Cube	Free − Terminal	-11.51 [-18.57, -5.19]	+12.5 [+5.7, +20.3]	0.0020

Table 8: TIME versus each literature baseline on the primary test episodes.

Task	Contrast	Δ steps [95% CI]	Δ SR (pp) [95% CI]	Holm p
TwoRoom	TIME − CEM	-62.49 [-70.67, -53.67]	+71.9 [+60.9, +82.3]	0.0002
TwoRoom	TIME − iCEM	-52.74 [-60.58, -44.77]	+52.1 [+40.6, +63.5]	0.0002
TwoRoom	TIME − MPPI*	-48.28 [-56.93, -39.57]	+52.1 [+40.6, +63.5]	0.0002
TwoRoom	TIME − GD	-45.05 [-52.31, -37.88]	+42.7 [+32.3, +53.1]	0.0002
TwoRoom	TIME − GRASP	-74.51 [-80.42, -68.03]	+84.4 [+76.0, +91.7]	0.0002
PushT	TIME − CEM	-20.90 [-28.63, -13.27]	+28.6 [+18.2, +39.1]	0.0002
PushT	TIME − iCEM	-20.47 [-28.00, -12.72]	+26.6 [+16.7, +36.5]	0.0002
PushT	TIME − MPPI*	-16.42 [-24.19, -8.59]	+18.7 [+7.8, +29.2]	0.0004
PushT	TIME − GD	-10.89 [-18.35, -3.46]	+9.4 [-0.5, +19.3]	0.0297
PushT	TIME − GRASP	-19.85 [-27.58, -11.86]	+28.6 [+18.2, +38.5]	0.0002
Reacher	TIME − CEM	-25.49 [-31.94, -19.48]	+16.1 [+8.3, +25.0]	0.0002
Reacher	TIME − iCEM	-15.99 [-20.88, -11.69]	+6.2 [+1.6, +11.5]	0.0002
Reacher	TIME − MPPI*	-31.77 [-36.68, -27.04]	+11.5 [+6.2, +17.2]	0.0002
Reacher	TIME − GD	-18.41 [-22.98, -14.24]	+6.8 [+2.6, +12.0]	0.0002
Reacher	TIME − GRASP	-43.68 [-53.08, -34.18]	+49.5 [+37.5, +61.5]	0.0002
Cube	TIME − CEM	-4.44 [-13.21, +4.81]	+4.7 [-5.2, +14.1]	1.0000
Cube	TIME − iCEM	-4.40 [-12.99, +4.58]	+4.7 [-5.2, +14.1]	1.0000
Cube	TIME − MPPI*	-14.93 [-23.30, -6.72]	+16.1 [+7.3, +25.0]	0.0044
Cube	TIME − GD	-2.60 [-11.12, +5.87]	+2.6 [-6.8, +12.0]	1.0000
Cube	TIME − GRASP	-1.90 [-9.96, +6.57]	+1.6 [-7.8, +10.4]	1.0000

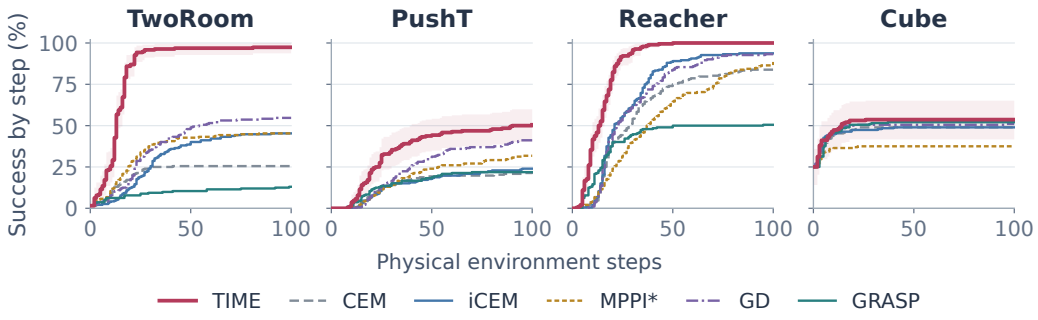

Figure 8: **Arrival curves for the external-system comparison.** The 64 primary episodes per task and three seeds, including initial successes. External methods use development-selected configurations; TIME uses $H = 8$. The band is TIME's pointwise 95% episode-bootstrap interval; other curves are point estimates. Figure 2 provides the fixed-H comparison with the Free variant and Terminal reference.

Table 9: **Matched prediction-horizon results.** Budgeted physical steps / success percent on the same 32 episodes per task and seed 42. The $H = 8$ rows use this subset only, so they differ from the 64-episode, three-seed primary means. All horizons use $B = 5$, a 100-action budget and the shared solver. Results are exploratory; no horizon is chosen from these test outcomes.

Task	H	Terminal	Free	TIME
TwoRoom	3	50.88 / 65.6	33.62 / 78.1	31.12 / 78.1
	5	50.66 / 68.8	21.09 / 96.9	16.56 / 96.9
	8	71.84 / 43.8	25.09 / 100.0	13.22 / 100.0
	12	82.38 / 25.0	24.41 / 96.9	12.97 / 100.0
PushT	3	80.50 / 37.5	56.62 / 65.6	68.19 / 43.8
	5	93.31 / 15.6	63.94 / 53.1	55.91 / 56.2
	8	92.84 / 12.5	67.41 / 50.0	67.09 / 50.0
	12	94.56 / 6.2	65.84 / 50.0	68.38 / 43.8
Reacher	3	70.78 / 50.0	21.00 / 100.0	14.78 / 100.0
	5	82.41 / 28.1	15.94 / 100.0	14.25 / 100.0
	8	92.66 / 9.4	17.84 / 100.0	14.53 / 100.0
	12	97.66 / 3.1	22.50 / 96.9	15.59 / 100.0
Cube	3	28.56 / 75.0	39.38 / 62.5	33.25 / 68.8
	5	38.44 / 65.6	33.91 / 68.8	40.25 / 62.5
	8	54.09 / 46.9	37.47 / 65.6	36.78 / 65.6
	12	51.78 / 50.0	40.66 / 62.5	36.72 / 65.6

Table 10: TwoRoom primary results: mean [95% CI] for budgeted steps, success rate, active-case steps and active-case success rate.

Method	Steps [95% CI]	SR (%) [95% CI]	Active steps [95% CI]	Active SR (%) [95% CI]
CEM	78.27 [69.02, 86.81]	25.5 [15.6, 36.5]	79.51 [70.57, 87.78]	24.3 [14.8, 34.9]
iCEM	68.52 [59.92, 76.80]	45.3 [34.4, 56.8]	69.61 [61.07, 77.94]	44.4 [33.3, 55.6]
MPPI*	64.06 [54.68, 73.30]	45.3 [34.4, 56.8]	65.07 [55.58, 74.28]	44.4 [33.3, 55.6]
GD	60.83 [52.60, 68.97]	54.7 [43.8, 65.1]	61.79 [53.86, 69.85]	54.0 [42.9, 64.6]
GRASP	90.29 [83.42, 96.22]	13.0 [6.2, 20.8]	91.72 [85.33, 97.01]	11.6 [5.3, 19.0]
Terminal	76.73 [71.16, 81.94]	42.7 [34.4, 51.0]	77.95 [72.89, 82.76]	41.8 [33.9, 49.7]
Free endpoint	25.08 [21.40, 29.29]	95.8 [91.7, 99.5]	25.48 [21.83, 29.72]	95.8 [91.5, 99.5]
TIME	15.78 [12.80, 19.59]	97.4 [93.8, 100.0]	16.03 [13.06, 19.84]	97.4 [93.7, 100.0]

Table 11: PushT primary results: mean [95% CI] for budgeted steps, success rate, active-case steps and active-case success rate.

Method	Steps [95% CI]	SR (%) [95% CI]	Active steps [95% CI]	Active SR (%) [95% CI]
CEM	85.10 [78.76, 90.90]	21.9 [14.1, 30.2]	85.10 [78.48, 90.98]	21.9 [14.1, 30.7]
iCEM	84.67 [78.81, 89.89]	24.0 [16.1, 32.3]	84.67 [78.87, 89.96]	24.0 [16.1, 32.3]
MPPI*	80.62 [74.74, 86.06]	31.8 [23.4, 40.6]	80.62 [74.83, 86.08]	31.8 [23.4, 40.1]
GD	75.09 [68.55, 81.38]	41.1 [31.8, 50.5]	75.09 [68.47, 81.38]	41.1 [31.8, 50.5]
GRASP	84.06 [76.67, 90.85]	21.9 [13.0, 31.2]	84.06 [76.58, 90.71]	21.9 [13.0, 31.2]
Terminal	94.99 [91.89, 97.62]	7.8 [4.2, 12.0]	94.99 [91.93, 97.59]	7.8 [4.2, 12.0]
Free endpoint	69.27 [62.21, 76.10]	49.0 [40.1, 57.8]	69.27 [61.96, 76.11]	49.0 [39.6, 58.3]
TIME	64.20 [56.15, 71.91]	50.5 [40.6, 60.4]	64.20 [56.21, 72.02]	50.5 [40.6, 60.4]

Table 12: Reacher primary results: mean [95% CI] for budgeted steps, success rate, active-case steps and active-case success rate.

Method	Steps [95% CI]	SR (%) [95% CI]	Active steps [95% CI]	Active SR (%) [95% CI]
CEM	39.95 [33.58, 46.81]	83.9 [75.0, 91.7]	39.95 [33.64, 46.71]	83.9 [75.0, 91.7]
iCEM	30.45 [26.23, 35.30]	93.8 [88.5, 98.4]	30.45 [26.34, 35.16]	93.8 [88.5, 97.9]
MPPI*	46.22 [41.10, 51.60]	88.5 [82.8, 93.8]	46.22 [41.16, 51.54]	88.5 [82.3, 93.8]
GD	32.87 [28.75, 37.49]	93.2 [88.0, 97.4]	32.87 [28.80, 37.42]	93.2 [88.5, 97.4]
GRASP	58.14 [47.77, 68.34]	50.5 [38.5, 62.5]	58.14 [47.69, 68.76]	50.5 [38.0, 62.5]
Terminal	95.36 [92.47, 97.82]	7.3 [3.6, 11.5]	95.36 [92.54, 97.82]	7.3 [3.6, 11.5]
Free endpoint	19.19 [16.96, 21.53]	100.0 [100.0, 100.0]	19.19 [17.00, 21.53]	100.0 [100.0, 100.0]
TIME	14.46 [12.84, 16.20]	100.0 [100.0, 100.0]	14.46 [12.77, 16.18]	100.0 [100.0, 100.0]

Table 13: Cube primary results: mean [95% CI] for budgeted steps, success rate, active-case steps and active-case success rate.

Method	Steps [95% CI]	SR (%) [95% CI]	Active steps [95% CI]	Active SR (%) [95% CI]
CEM	52.73 [41.11, 64.22]	49.0 [37.0, 60.9]	70.31 [58.17, 81.78]	31.9 [19.4, 45.1]
iCEM	52.70 [41.25, 63.63]	49.0 [37.5, 60.9]	70.26 [58.59, 81.46]	31.9 [20.1, 44.4]
MPPI*	63.23 [52.36, 73.59]	37.5 [27.1, 48.4]	84.31 [76.15, 91.58]	16.7 [9.0, 25.0]
GD	50.90 [39.70, 62.09]	51.0 [39.6, 62.5]	67.87 [55.92, 79.13]	34.7 [22.9, 47.2]
GRASP	50.20 [38.42, 61.76]	52.1 [40.1, 64.1]	66.93 [54.49, 78.84]	36.1 [22.9, 49.3]
Terminal	56.10 [44.73, 67.00]	45.3 [33.9, 56.8]	74.80 [63.99, 84.94]	27.1 [16.0, 38.9]
Free endpoint	44.59 [33.73, 55.44]	57.8 [46.9, 68.8]	59.46 [47.86, 70.94]	43.8 [31.9, 56.2]
TIME	48.30 [37.08, 59.66]	53.6 [41.7, 65.1]	64.40 [52.44, 75.75]	38.2 [26.4, 50.7]