STOPTHEDREAM: OUTCOME-GROUNDED CANDIDATE COMPARISON FOR ADAPTIVE SEARCH IN WORLD-MODEL PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

A world-model planner can always select the best candidate it has imagined, even when its predictions provide weak evidence that the selected action will achieve the goal. Generating more candidates can help, but adds computation and can amplify the same scoring error. We introduce StopTheDream, a framework that connects candidate actions to their predicted consequences and uses this evidence for candidate comparison and bounded search. The procedure makes generation, comparison, and computation allocation explicit, grounding its scoring objectives in observed actions and outcomes. On 120 hierarchical world model (HWM) maze tasks, the policy solves 117 tasks versus CoVO-MPC's 113, using 48.58% fewer latent transitions and 24.38% less planning time. The original HWM planner solves 105 tasks with the same world model. On 1,000 CAST visual trajectories, it attains 1.3629 absolute trajectory error, 4.18% below a locally trained PiJEPA baseline, with 54.11% fewer predictor calls. These results show higher observed planning quality with substantially less predictive computation, supporting joint evaluation of decision quality and computational cost.

1 INTRODUCTION

In embodied visual control and trajectory planning, world models enable agents to simulate the consequences of prospective action sequences before committing to physical execution (Ha & Schmidhuber, 2018; Hafner et al., 2023; Hansen et al., 2024). Because generating such imagined trajectories is computationally costly, practical sampling based planners typically evaluate only a finite set of candidate action sequences and select the candidate with the highest predicted value. However, this ranking is inherently relative: identifying the best candidate among those considered does not establish that the candidate is sufficiently good to execute. When all available candidates are poor, a planner can therefore make a confident selection without having found a viable plan.

This introduces a second planning decision: whether to execute the current best candidate or spend additional computation searching for a better one. As illustrated in Figure 1, the appropriate choice depends on the evidence available for the current candidates. When the initial rollouts are uniformly unpromising, additional search may reveal a viable alternative. In contrast, when the available evidence already strongly supports one candidate, further search may consume computation without improving the eventual decision. Thus, effective planning requires not only ranking candidates precisely, but also determining whether the current evidence is sufficient to justify execution.

However, more predictive computation does not necessarily yield better decisions. As shown in Figure 2, across HWM, CAST, and Target-Bench, performance initially improves as the planner predicts farther ahead, but deteriorates as the horizon continues to grow. These results suggest that predictive computation should be treated as a limited planning resource whose value depends on the current decision. Rather than increasing imagination uniformly, a planner should allocate additional computation only when the available evidence does not yet support a confident action choice.

To address this problem, we introduce StopTheDream, a planning framework that directly couples candidate evaluation with search allocation. At each planning step, StopTheDream proceeds in three stages: it generates candidate actions together with their imagined trajectories, evaluates these candidates using task relevant evidence, and determines whether the available evidence is sufficient for

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: **The best available plan may still be a bad plan.** In this conceptual example of conditional search with two candidates, a promising plan supports execution (left). If both predicted success scores are low and budget permits, one additional search supplies a candidate to compare with the retained choice (right). Dashed paths represent imagined trajectories, while the solid teal segment indicates the next executed action. Crosses indicate paths with low predicted success scores.

execution or whether additional search is warranted. Rather than evaluating an imagined future in isolation, our formulation preserves the connection between each candidate action and its predicted consequence, since a favorable imagined outcome does not by itself establish that the proposed action can reliably produce it. We further ground candidate evaluation in observed execution outcomes, providing the scoring function with a behavioral target for assessing candidate quality. The resulting scores then guide a bounded decision rule that commits to the current candidate when the evidence is sufficient and allocates additional computation when the available candidates remain unpromising.

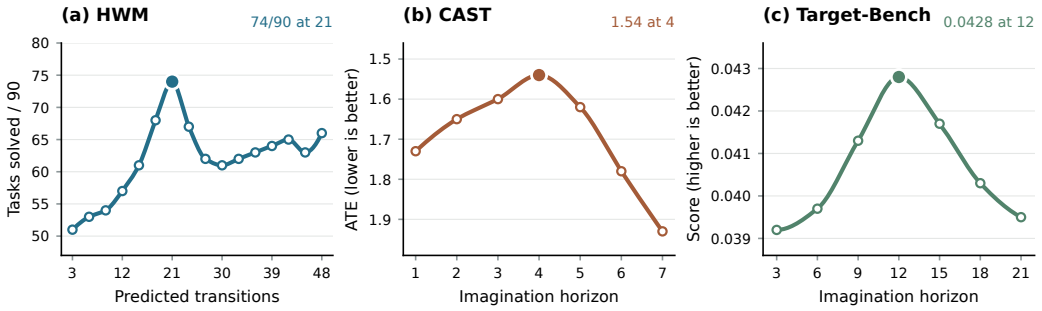

Figure 2: **Observed performance by imagination horizon.** (a) HWM tasks solved out of 90; (b) CAST absolute trajectory error (lower is better); (c) Target-Bench score (higher is better). Filled circles mark the best observed depth in each panel. Open circles are reported aggregates; shape-preserving interpolation connects them only as a visual guide. Metrics and horizon scales are setting-specific. Tables 9 and 10 list all plotted coordinates in detail.

Our evaluation reports complete-policy benchmarks against competitive planners together with their measured computation costs and a controlled sensitivity analysis of the additional-search rule. On the 120 task HWM maze benchmark, StopTheDream solves 117 tasks compared to 113 for CoVO-MPC (Yi et al., 2024), while reducing latent transitions by 48.58% and planning latency by 24.38%. On the 1,000 CAST visual trajectory tracking tasks, it achieves an absolute trajectory error (ATE) of 1.3629 and relative pose error (RPE) of 0.4080. Our contributions are summarized as follows:

- **We propose an adaptive world-model planning framework (StopTheDream)** that dynamically adjusts search effort by evaluating candidate actions alongside their imagined consequences, committing early when a viable plan is found and allocating bounded search only when initial rollouts are unpromising.
- **We introduce outcome-grounded scoring objectives** that evaluate full action-trajectory pairs against observed execution outcomes, connecting candidate preference to realized task success.

- **We improve the quality–computation trade-off in world-model planning**, and the same stop-or-acquire decision transfers to feedback timing and exploration.

2 Related Work

We review prior work on world models and sampling-based planning, followed by approaches to adaptive computation, verification, feedback, and reuse. This review provides the context for our joint treatment of candidate selection and search allocation.

World models and candidate evaluation. World models predict consequences for planning or policy learning (Ha & Schmidhuber, 2018; Hafner et al., 2023). Latent and video models extend prediction to visual observations (Assran et al., 2025; Zhang et al., 2026). TD-MPC2 combines finite latent rollouts with a learned terminal value (Hansen et al., 2024); PiJEPA initializes search from policy proposals (Chahe & Zhou, 2026). Our focus is comparing executable candidates while preserving the connection between actions, predicted consequences, and the task objective.

Sampling-based planning. Model predictive path integral (MPPI) control updates an action sequence using perturbations weighted by predicted trajectory cost (Williams et al., 2017). More samples improve coverage without necessarily correcting a deficient scoring signal. StopTheDream couples evidence-based candidate comparison with the allocation of further search.

Adaptive computation. Value-of-computation methods ask whether further deliberation improves a decision enough to justify its cost (Callaway et al., 2018; Sezener & Dayan, 2020). Adaptive imagination controls whether and how much to predict (Yu et al., 2026); ELASTIC allocates denoising steps and parallel samples for generative control (Li et al., 2026). Our bounded search rule uses evidence about current candidates to decide whether to execute an action or obtain further predictions.

Verification, feedback, and reuse. Future verifiers assess completed predictions (Wang et al., 2026a;b), while efficient world–action models retain or transfer useful representations (Yuan et al., 2026b; Cohen et al., 2026). Our comparison retains the candidate action alongside its evidence and connects the resulting score to the next computation decision.

3 Preliminaries: Prediction and Action Search

We first introduce the task setting and notation for candidate actions, world-model imaginations, and execution. We then describe prediction-based planning and identify the two decisions addressed by our method: which candidate to execute and whether to perform further search.

Task and available information. We consider goal-conditioned sequential decision making. At physical time t, the environment has state s_t, the controller receives observation o_t, and a_t denotes its executed action. The observation encoder f_{enc} maps history $h_t = (o_0, a_0, \ldots, a_{t-1}, o_t)$ to context $z_t = f_{\mathrm{enc}}(h_t)$. The task supplies a goal g and an outcome criterion. Decisions use available observations; future execution outcomes can supervise offline training.

Imagination and execution. A candidate action sequence is $\mathbf{a}_b = (a_{b,0}, \ldots, a_{b,N_b-1})$, where b uniquely identifies a candidate within a decision, N_b counts its physical actions, and $a_{b,j}$ is its proposed action at relative step j. World model P_θ produces an *imagined trajectory*:

$$\widehat{z}_h^{(b)} = P_\theta^{(h)}(z_t, \mathbf{a}_b), \qquad \widehat{\mathbf{z}}^{(b)} = (\widehat{z}_1^{(b)}, \ldots, \widehat{z}_{H_b}^{(b)}). \tag{1}$$

Here θ denotes model parameters, $P_\theta^{(h)}$ the h-step prediction from the current context, and H_b the prediction horizon, with $h = 1, \ldots, H_b$. A hat marks a prediction and bold type a sequence. Temporal abstraction can make H_b differ from N_b. Predicting leaves the environment unchanged. The *action prefix* $\mathbf{a}_{b,0:L-1}$ contains the first L actions, where $1 \le L \le N_b$; execution ends earlier if the task terminates. Prediction extent, search effort, and executed length are distinct.

Common methods for prediction-based action planning. Standard model predictive control (MPC) optimizes a finite action sequence under predicted dynamics and task costs, executes its initial prefix, and replans from feedback (Rawlings et al., 2017).

Numerical optimization and sampling provide different solvers: real-time iteration uses Newton-type updates (Diehl et al., 2005), MPPI uses cost-weighted samples (Williams et al., 2017), and TD-MPC combines latent rollouts with a learned terminal value (Hansen et al., 2022; 2024).

Predictive planning interfaces. Temporal hierarchies change the abstraction of candidate actions (Zhang et al., 2026), while policy priors initialize their search distribution (Chahe & Zhou, 2026). In either case, a planner proposes actions, predicts their consequences, evaluates a task cost, and updates or selects proposals. This provides the candidate-generation interface used below; representation choices and optimizer configurations belong to the task implementation.

The decision left to the controller. These mechanisms return a preferred action under a predictive objective and allocated search effort. Relative preference alone does not determine whether the available option is good enough or further search is worthwhile. We couple *which candidate to execute* with *whether to obtain further predictive evidence*. The search configuration controls sampling effort and prediction extent; the execution protocol specifies when to replan.

4 METHODOLOGY

StopTheDream couples candidate selection with the computation used to obtain candidates. We define generation, comparison, search allocation, supervision, and execution through common interfaces. Figure 3 and Algorithm 1 illustrate the schedule and specify the inference procedure.

4.1 GENERATING CANDIDATES

Let r index search rounds before physical execution, B_r specify the requested sampling effort and prediction extent, and $\mathcal{D}_{r-1}$ record earlier candidates, computed scores, and measured costs, initially $\mathcal{D}_0 = \varnothing$. The generator combines action search with world-model prediction:

$$\mathcal{P}_r = \mathrm{Generate}_{P_\theta}(h_t, g; B_r, \mathcal{D}_{r-1}), \qquad \xi_b = (\mathbf{a}_b, \widehat{\mathbf{z}}^{(b)}, c_b) \in \mathcal{P}_r. \tag{2}$$

Here $\mathcal{P}_r$ is a nonempty finite set, and each record ξ_b contains executable actions with $N_b \geq L$, their predictions, and an optional computation descriptor c_b. The planner proposes, evaluates, and updates actions using P_θ under B_r; several internal samples can produce one returned record. A permitted direct-policy branch has $H_b = 0$ and an empty predicted trajectory, while still incurring policy and observation-processing costs. Features such as realized prediction counts and generation time enter c_b only after generation. In the full/split schedule of Figure 3, one full search uses K_f sampled proposals, while each of two independent smaller searches uses $K_\mathrm{s} \approx K_\mathrm{f}/2$ and returns one candidate. These labels specify sampling allocation; the predictive model, action interface, and prediction extent are specified separately by the search configuration.

4.2 COMPARING ACTIONS THROUGH THEIR PREDICTED CONSEQUENCES

The scorer evaluates actions together with their predicted consequences. Let $\mathcal{I}_{r-1}$ be the retained records, with $\mathcal{I}_0 = \varnothing$, and $\mathcal{C}_r = \mathcal{I}_{r-1} \cup \mathcal{P}_r$ the comparison set. Scorer S_ϑ assigns preference $q_{r,b}$ to candidate b; the selected index b_r^* satisfies

$$q_{r,b} = S_\vartheta(h_t, g, \xi_b; \mathcal{C}_r), \qquad b_r^* \in \arg\max_{b:\xi_b \in \mathcal{C}_r} q_{r,b}. \tag{3}$$

The configuration ϑ comprises scoring parameters and fixed transformations. Larger scores are preferred; a minimized task cost J_b gives $q_{r,b} = -J_b$. Ties follow the declared selection rule. Context-dependent scores are recomputed when the comparison set changes. A singleton needs no ranking, but the stopping rule can still request its score. Figure 3 illustrates paired encoding and comparison, with architecture and training details in Appendix B.1.

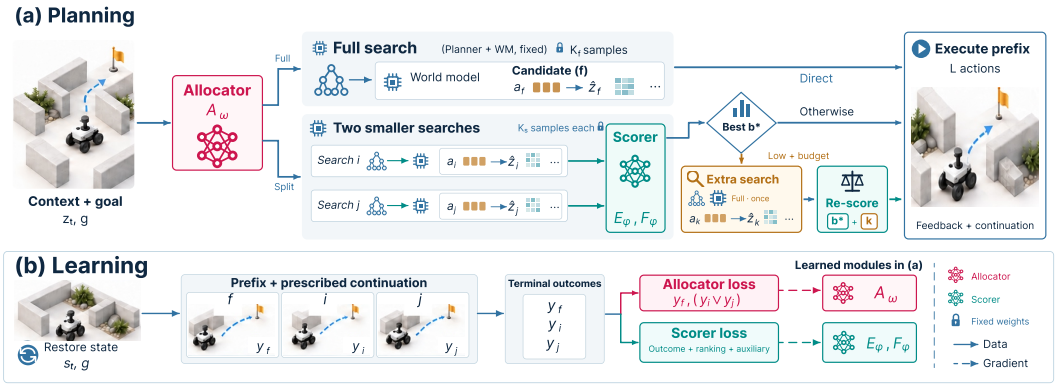

Figure 3: **Planning and learning within a bounded search schedule.** (a) Allocator A_ω uses observed context and goal g to choose one full search (budget B_f) or two smaller searches (budget B_s each). Full and smaller searches use K_f and $K_\mathrm{s} \approx K_\mathrm{f}/2$ internal proposals, respectively. The full route selects candidate f directly. The split route encodes actions $\mathbf{a}_b$ and predictions $\widehat{\mathbf{z}}^{(b)}$ with E_ϕ, then compares candidates i, j with F_ϕ. Low scores p_i, p_j and available budget permit one additional candidate k; both k and the retained incumbent b^* are rescored before executing L actions. (b) Same-state executions provide terminal outcomes y_b for fitting the allocator and scorer. Solid arrows carry data; dashed arrows carry gradients; locks mark fixed model weights. Scenes are schematic. Algorithm 1 specifies inference; Appendix B.6 details retention and supervision.

4.3 Allocating Search Before Execution

The rule κ allocates computation and retains candidates. Let $\mathcal{B}$ contain configurations with bounded search effort and prediction extent, and d_r denote the decision after round r:

$$B_1 = \kappa(h_t, g, \varnothing) \in \mathcal{B}, \qquad d_r = \kappa(h_t, g, \mathcal{D}_r) \in \{\mathrm{stop}\} \cup \mathcal{B}. \tag{4}$$

A non-stop decision sets $B_{r+1} = d_r$ and retains an executable subset $\mathcal{I}_r \subseteq \mathcal{C}_r$ according to κ. The rule requests any singleton score it needs and uses only observed context, acquired predictions, computed scores, and consumed resources. Search ends at a stop decision or the positive integer round limit $R_{\max}$. Configurations, retention, and stopping rules are fixed before evaluation; their decisions depend on the currently available candidate evidence.

4.4 Supervision and Execution Targets

Execution supervision measures the consequence of selecting a candidate. Matched development executions share a restored start, execute each candidate prefix, and finish under the same continuation policy π_{cont}. Task outcomes determine the scoring target and loss; related states or layouts remain in the same data split. Online decisions use observations and predictions rather than these future labels. A predictive task cost can alternatively supply the score directly. Appendix A.2 maps each task to the candidate, evidence, and computation interfaces used by its controller.

4.5 Feedback and Computation Cost

Let R be the realized number of search rounds. Execute the selected prefix $\mathbf{a}_{b^*_R, 0:L-1}$ until it finishes or the task terminates, and let $m \leq L$ count the actions actually executed. Real observations update h_{t+m}; if the task continues, π_{cont} resumes from this history. Feedback initiates another invocation only at the decision points specified by the task protocol. Search rounds change predictive evidence before acting; execution changes the physical state and supplies new observations.

Costs include preparation, candidate generation, scoring, allocation, and continuation; shared operations are counted once. Time, predictor calls, and transitions are recorded separately (Appendix A.3).

Algorithm 1 Candidate Comparison Under a Bounded Search Rule

Require: History h_t at a nonterminal decision, goal g, model P_θ, scorer S_ϑ, rule κ
Require: Round limit $R_{\max} \geq 1$, prefix length $L \geq 1$, continuation π_{cont}
Ensure: Selected action prefix and updated information for continuation

Initialize resource counters before preparation and search
$\mathcal{D}_0 \leftarrow \varnothing$; $\mathcal{I}_0 \leftarrow \varnothing$; $B_1 \leftarrow \kappa(h_t, g, \varnothing)$
for $r = 1, \ldots, R_{\max}$ **do** ▷ Search without executing actions
  $\mathcal{P}_r \leftarrow \text{Generate}_{P_\theta}(h_t, g; B_r, \mathcal{D}_{r-1})$
  $\mathcal{C}_r \leftarrow \mathcal{I}_{r-1} \cup \mathcal{P}_r$
  if $|\mathcal{C}_r| > 1$ or κ requests a singleton score **then**
    Recompute scores for $\mathcal{C}_r$; select b_r^* by Equation 3
  else
    Select the sole candidate b_r^* without scoring
  end if
  Append records, computed scores (if any), and measured costs to form $\mathcal{D}_r$
  $d_r \leftarrow \kappa(h_t, g, \mathcal{D}_r)$
  if $d_r = \text{stop}$ or $r = R_{\max}$ **then**
    $R \leftarrow r$; **break**
  end if
  Retain $\mathcal{I}_r \subseteq \mathcal{C}_r$ as specified by κ; set $B_{r+1} \leftarrow d_r$
end for
Execute $\mathbf{a}_{b_R^*, 0:L-1}$ until completion or task termination; record executed count m
Update h_{t+m} with the executed actions and real observations
if the task has not terminated **then**
  Continue with π_{cont} from h_{t+m} under the task protocol
end if

4.6 Instantiation

For HWM, allocator A_ω reads z_t, g, and pre-search conditions c_{init} to predict success from one full search and the probability that at least one of two smaller searches succeeds:

$$(q_{\text{full}}, q_{\text{split}}) = A_\omega(z_t, g, c_{\text{init}}), \qquad \text{split if } q_{\text{split}} - q_{\text{full}} \geq \delta. \tag{5}$$

Development states calibrate δ. A full search executes directly; a split yields candidates i, j, whose pair-conditioned success scores use encoder E_ϕ and comparator F_ϕ:

$$u_b = E_\phi(z_t, g, \mathbf{a}_{b,0:L-1}, \widehat{\mathbf{z}}^{(b)}, c_b), \qquad (p_i, p_j) = \sigma(F_\phi(u_i, u_j, g)), \tag{6}$$

Here u_b is a candidate embedding, σ the elementwise sigmoid, and ω, ϕ separate learned parameters. If $\max(p_i, p_j) < \eta$ ($\eta = 0.5$), generate one full-budget addition and jointly rescore it with the retained winner; otherwise execute the winner. Only one addition is allowed. Executing $L = 4$ actions and the common continuation from a restored development state gives the terminal label $y_b = \text{Success}(\text{Execute}(s_t, \mathbf{a}_{b,0:L-1}; \pi_{\text{cont}})) \in \{0, 1\}$, used in

$$\mathcal{L}_\phi = \tfrac{1}{2} \sum_{b \in \{i,j\}} \text{BCE}(p_b, y_b) + 2\mathcal{L}_{\text{pair}} + \mathcal{L}_{\text{aux}}. \tag{7}$$

Binary cross-entropy (BCE) predicts success, $\mathcal{L}_{\text{pair}}$ ranks unequal outcomes, and $\mathcal{L}_{\text{aux}}$ fits gain, first-success, endpoint, and probability-error targets (Appendix B.5). The allocator uses this objective with full-search label y_{f} and split-union label $y_i \vee y_j$; the union describes an available opportunity. The predictive model and planner stay frozen while the two decision modules are trained. Equation 7 shows one fully valid pair. During fitting, the outcome-stratified sampler supplies importance weights, and each loss uses its own validity mask and normalization. Failed executions supply negative labels but no first-success target; Appendix B.5 defines loss masks and reductions.

Algorithm 2 Learning from Matched Candidate Executions

Require: Development roots (s, h, g), calibration roots, fixed planner and world model
Require: Budgets $B_{\mathrm{f}}, B_{\mathrm{s}}$, prefix L, common continuation π_{cont}
Initialize paired records $\mathcal{T}_{\mathrm{pair}}$ and allocation records $\mathcal{T}_{\mathrm{alloc}}$
for each development root (s, h, g) **do**
    Generate $\xi_{\mathrm{f}}, \xi_i, \xi_j$ from that root with budgets $B_{\mathrm{f}}, B_{\mathrm{s}}, B_{\mathrm{s}}$
    Restore the root for each candidate; execute its prefix and common continuation
    Record terminal success y_b and auxiliary outcomes for $b \in \{\mathrm{f}, i, j\}$
    Add labeled pairs $(\mathrm{f}, i), (\mathrm{f}, j), (i, j)$ to $\mathcal{T}_{\mathrm{pair}}$
    Add full/split targets $(y_{\mathrm{f}}, y_i \vee y_j)$ to $\mathcal{T}_{\mathrm{alloc}}$
end for
Fit scorer and allocator separately using Eq. 7; mask allocator actions/futures
Calibrate thresholds on development/calibration roots; freeze both fits before evaluation

5 Experiments

We test whether outcome-grounded candidate comparison improves planning, and whether the same stop-or-acquire decision transfers to feedback timing and exploration. Main comparisons prioritize shared predictive models or matched evidence-acquisition paths.

5.1 Tasks and Evaluation

HWM tests terminal success on 120 maze tasks, and CAST measures continuous trajectory error on 1,000 observation-conditioned trajectories. Planners share the predictive model within each setting. The learned-feedback study uses 1,000 paired LIBERO-Long paths with the same policy and initial states; A-EQA compares selective and full exploration on 184 questions using shared initial observations and the same explorer. Figure 4 also retains the four-suite LIBERO and ROOM extensions.

Each protocol compares quality and cost on every evaluated case. Charges include discarded proposals, scoring, and continuation. Figure 4 reports native units; Appendix A gives the protocols.

Planning implementations. Section 4.6 specifies the HWM allocator and paired success scorer. The reported HWM protocol applies this decision initially, then executes the selected prefix and its prescribed continuation. CAST uses separately trained continuous scores and permits a direct-policy branch; Appendix C specifies its candidate interface, scoring procedure, and planner comparisons.

Across planning, feedback, and exploration, candidates are action plans, execution chunks, and answers; evidence comprises predicted trajectories or observed views; additional computation obtains another search, visual feedback, or exploration, respectively (Appendix A.2, Table 4).

5.2 Choosing Actions and Requesting Feedback

Task success beyond the predicted endpoint. A short predicted trajectory can look favorable while its executed continuation fails. HWM therefore tests whether scoring candidates against execution outcomes improves the complete planning decision. Table 1 shows 117/120 successes, versus CoVO-MPC's 113/120, with 48.58% fewer latent transitions and 24.38% less planning time. Appendix J reports the statistical summaries determined by these aggregate counts. Smooth MPPI is faster (17,358.6 versus 18,934 seconds) but solves fewer tasks (109/120). These operating points favor coupling candidate selection to computation rather than maximizing search volume alone.

Continuous trajectory quality. CAST tests the same decision structure with continuous trajectory error, separately trained scores, and a direct-policy branch. It reaches ATE 1.3629 and RPE 0.4080 at 12.85 predictor calls per trajectory (Table 1). Relative to PiJEPA, error falls by 4.18% and calls by 54.11%; relative to DIAL-MPC, they fall by 9.64% and 69.40%. Thus the improvement extends beyond binary terminal-success supervision to continuous trajectory quality.

How long to execute before observing again. Executing long action chunks saves policy calls but delays visual correction. On LIBERO-Long, the learned feedback rule reaches 92.83% success with

Table 1: **Candidate-based planning under shared predictive models.** Both panels compare planners using the same predictive model within each setting; CAST also includes a locally trained PiJEPA baseline. Underlining marks the best external primary metric (success or ATE).

(a) HWM: 120 maze tasks

Method	Success	Trans. (M)	Time (s)
SVG-MPPI	53/120	3,081.74	119,816.8
MPOPIS	93/120	7,413.54	245,294.2
DIAL-MPC	82/120	1,620.72	58,012
CoVO-MPC	<u>113/120</u>	747.48	25,036.9
Feedback-MPPI	111/120	741.66	28,041.5
CSC-MPPI	105/120	1,088.99	37,941
Smooth MPPI	109/120	741.66	17,358.6
HWM	105/120	741.66	31,765.1
StopTheDream	**117/120**	**384.39**	**18,934**

(b) CAST: 1,000 trajectories

Method	ATE	RPE	Calls/traj.
SVG-MPPI	1.5733	0.5100	35.0
MPOPIS	1.9446	0.7433	42.0
DIAL-MPC	1.5083	0.5412	42.0
CoVO-MPC	1.7149	0.6164	49.0
Biased-MPPI	1.5518	0.5370	42.0
Smooth MPPI	2.0944	0.6602	42.0
PiJEPA (local)	<u>1.4224</u>	0.4484	28.0
StopTheDream	**1.3629**	**0.4080**	**12.85**

Success is higher-is-better; errors and computation are lower-is-better. M denotes millions of latent transitions. CAST calls are means per trajectory, with a batch of candidates processed per call. Models, evaluation records, and timing scopes: Appendices B and C.

20.853 calls per path, versus 92.40% and 22.252 for the training-fold task prior: 0.43 percentage points higher success and 6.29% fewer calls. Fixed ten-action feedback reaches 91.60% at 27.619 calls; sixteen-action execution is cheaper (18.670 calls) but less successful (88.10%). Adaptive feedback improves the trade-off without simply choosing the longest chunk.

Figure 4(d,e) additionally reports suite-calibrated LIBERO control (1,981/2,000 successes) and ROOM observation reuse (100/100), whose distinct protocols are detailed in Appendices E and G.

5.3 Acquiring Evidence for Answers

Selective exploration with a shared explorer. A-EQA asks whether the initial observations suffice or real exploration is useful. With shared initial views, the same LLaVA backbone, explorer, and judge, direct answering scores 42.935 LLM-Match in 918.75 seconds; full exploration scores 48.913 after opening 184 branches in 31,626.23 seconds. Selecting 156 branches retains 48.913 in 26,120 seconds, removing 28 explorations (15.22%) and reducing process time by 17.41% at unchanged aggregate quality (Figure 4(f); Table 19). See Appendix H for fitting and evaluation details.

More reasoning over the same images. On SAT, the same Gemma backbone answers and compares candidates from the original images. Selective reasoning reaches 3,698/4,001 correct (92.43%) versus the combined baseline's 3,205/4,001 (80.10%), with 9,853 versus 8,543 requests: 493 additional correct answers at 15.33% more requests. This is a quality gain with increased computation; the released question type triggers the extra reasoning (Appendix I).

Appendix A.5 reports WBench video quality, Target-Bench pool selection, and external SAT comparisons under their respective generation, fitting, and backbone configurations.

5.4 When Does Further Prediction Help?

In the controlled 90-task horizon study (Figure 2), success rises from 51 at depth 3 to 74 at depth 21, then falls to 66 at depth 48. Success thus varies non-monotonically with prediction depth. The sweep changes prediction depth while keeping starts, goals, the model, and the sampling-noise prefix fixed.

CAST and Target-Bench likewise peak at intermediate depths under their own metrics (Appendix D). These scans motivate adaptive computation: longer predictions alone do not ensure better decisions. They characterize the underlying prediction process rather than the proposed controller.

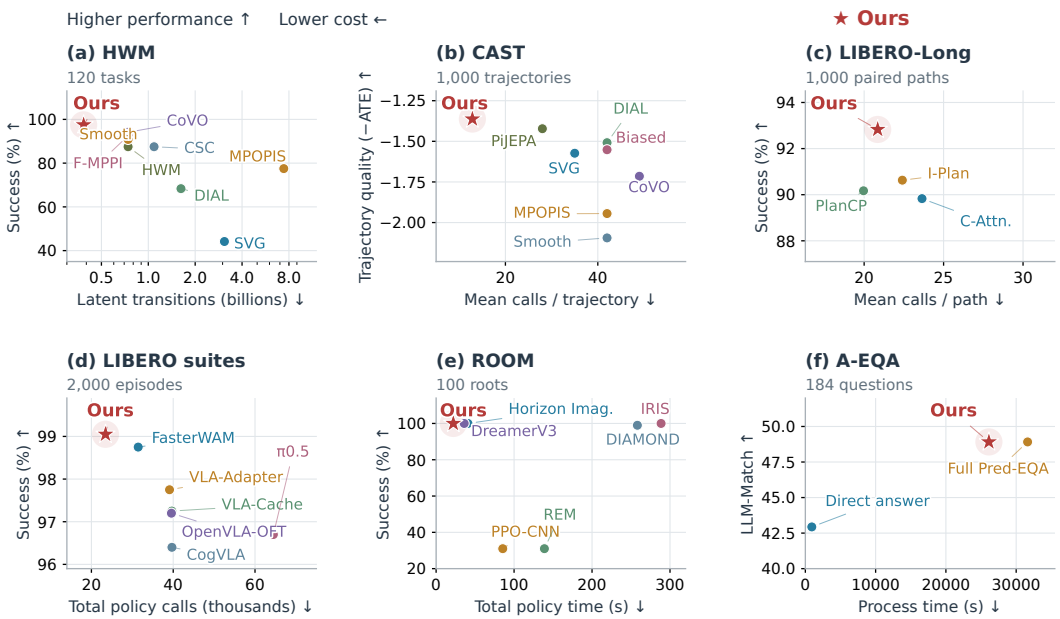

Figure 4: **Task performance versus computation.** Upward means better performance and leftward means lower cost in every panel; CAST uses $-$ATE, with the original errors in Table 1. Red stars mark StopTheDream. Panels (a)–(f) cover 120 HWM tasks, 1,000 CAST trajectories, 1,000 paired LIBERO-Long paths, 2,000 LIBERO episodes, 100 ROOM roots, and 184 A-EQA questions. HWM uses a logarithmic cost axis. Calls count requests within the indicated model; ROOM time sums policy decisions over complete episodes, and A-EQA time covers the complete answer and exploration process. Each panel retains its own units and protocol.

5.5 Sensitivity of Additional Search

Table 2: HWM sensitivity on the same 120 tasks with learned weights fixed. Additions are extra candidates generated; selected additions are those chosen for execution.

η	Success	Latent transitions	Additions	Selected additions
0	116/120	383,251,360	0	0
0.25	116/120	384,331,292	10	6
0.4	117/120	384,386,362	11	7
0.5	117/120	384,386,362	11	7
0.6	117/120	384,516,522	13	7
0.75	116/120	384,591,612	14	6
1	115/120	393,981,398	99	36

Table 2 shows 117 successes throughout $\eta \in \{0.4, 0.5, 0.6\}$, using 11–13 additions. At $\eta = 0.5$, one more task is solved than with addition disabled, at 0.30% more transitions. Raising η to one generates 99 additions but solves 115 tasks; only 36 additions are selected. The observed trade-off favors selective acquisition over maximizing search volume.

6 Conclusion

StopTheDream couples outcome-grounded candidate comparison with bounded search. Shared-world-model evaluations improve observed planning quality while reducing predictive computation. Feedback and exploration studies extend the stop-or-acquire decision to obtaining new observations, supporting computation allocation according to its value for the resulting task decision.

AI USE STATEMENT

The research idea originated with the authors. Generative AI tools, including OpenAI ChatGPT/Codex, assisted with code writing, discussion and clarification of the mathematical formulation, feedback on experimental designs, interpretation and presentation of author-supplied results, literature discovery, and manuscript drafting and revision. Image-generation tools assisted with conceptual illustrations and figure-design exploration. The quantitative plots were rendered from the reported numerical values rather than taken from generated chart images. AI assistants were not used to generate, clean, or otherwise process the experimental datasets. The authors are responsible for the final text, scientific claims, code, measurements, citations, and illustrations.

ETHICS STATEMENT

This work studies computation allocation using existing benchmarks and simulated or recorded environments. Predicted outcomes and model-based answer judgments can be incorrect or biased; reduced computation does not itself establish safe behavior in physical environments. Deployment would require application-specific validation, appropriate monitoring, and safeguards for failures of perception and prediction. Use and redistribution of the underlying models and datasets remain subject to their respective licenses and access conditions. Generated scene illustrations explain the method and are not experimental observations.

REPRODUCIBILITY STATEMENT

Section 4 defines the candidate, comparison, and computation interfaces, and Algorithm 1 gives the inference procedure. Appendix A specifies task interfaces and computation accounting. Appendix B details HWM training and planning; Appendix C describes CAST task interfaces, planner comparisons, and evaluation settings; Appendices D–I provide the remaining task protocols and aggregate results. Figure 2 is accompanied by its complete plotted coordinates in Tables 9 and 10, and the task-specific tables give the values and resource units underlying Figures 4 and 5.

REFERENCES

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

Dylan M. Asmar, Ransalu Senanayake, Shawn Manuel, and Mykel J. Kochenderfer. Model predictive optimized path integral strategies. In *2023 IEEE International Conference on Robotics and Automation*, pp. 3182–3188, 2023. doi: 10.1109/ICRA48891.2023.10160929. URL https://doi.org/10.1109/ICRA48891.2023.10160929.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. URL https://arxiv.org/abs/2506.09985.

Tommaso Belvedere, Michael Ziegltrum, Giulio Turrisi, and Valerio Modugno. Feedback-MPPI: Fast sampling-based MPC via rollout differentiation – adios low-level controllers. *IEEE Robotics and Automation Letters*, 11(1):1–8, 2026. doi: 10.1109/LRA.2025.3630871. URL https://doi.org/10.1109/LRA.2025.3630871.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Frederick Callaway, Sayan Gul, Paul M. Krueger, Thomas L. Griffiths, and Falk Lieder. Learning to select computations. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*, pp. 776–785, 2018. URL https://auai.org/uai2018/proceedings/papers/269.pdf.

Amirhosein Chahe and Lifeng Zhou. Policy-guided world model planning for language-conditioned visual navigation. *arXiv preprint arXiv:2603.25981*, 2026. URL https://arxiv.org/abs/2603.25981.

An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. SpatialRGPT: Grounded Spatial Reasoning in Vision Language Models. *arXiv preprint arXiv:2406.01584*, 2024. URL https://arxiv.org/abs/2406.01584.

Lior Cohen, Kaixin Wang, Bingyi Kang, and Shie Mannor. Improving Token-Based World Models with Parallel Observation Prediction. *arXiv preprint arXiv:2402.05643*, 2024a. URL https://arxiv.org/abs/2402.05643.

Lior Cohen, Ofir Nabati, Kaixin Wang, Navdeep Kumar, and Shie Mannor. Horizon imagination: Efficient on-policy training in diffusion world models. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2602.08032.

Nathaniel Cohen, Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Slicedit: Zero-Shot Video Editing With Text-to-Image Diffusion Models Using Spatio-Temporal Slices. *arXiv preprint arXiv:2405.12211*, 2024b. URL https://arxiv.org/abs/2405.12211.

Yuren Cong, Mengmeng Xu, Christian Simon, Shoufa Chen, Jiawei Ren, Yanping Xie, Juan-Manuel Perez-Rua, Bodo Rosenhahn, Tao Xiang, and Sen He. FLATTEN: optical FLow-guided ATTENtion for consistent text-to-video editing. *arXiv preprint arXiv:2310.05922*, 2023. URL https://arxiv.org/abs/2310.05922.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and

PixMo: Open weights and open data for state-of-the-art vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Deitke_Molmo_and_PixMo_Open_Weights_and_Open_Data_for_State-of-the-Art_CVPR_2025_paper.html.

Moritz Diehl, Hans Georg Bock, and Johannes P. Schlöder. Real-time iterations for nonlinear optimal feedback control. In *44th IEEE Conference on Decision and Control and European Control Conference*, pp. 5871–5876, 2005. URL https://cdn.syscop.de/publications/Diehl2005c.pdf.

Michal Geyer, Omer Bar-Tal, Shai Bagon, and Tali Dekel. TokenFlow: Consistent Diffusion Features for Consistent Video Editing. *arXiv preprint arXiv:2307.10373*, 2023. URL https://arxiv.org/abs/2307.10373.

Google DeepMind. Gemma 4 Model Card. Official model documentation, 2026. URL https://ai.google.dev/gemma/docs/core/model_card_4.

Yuwei Guo, Ceyuan Yang, Anyi Rao, Maneesh Agrawala, Dahua Lin, and Bo Dai. SparseCtrl: Adding Sparse Controls to Text-to-Video Diffusion Models. *arXiv preprint arXiv:2311.16933*, 2023. URL https://arxiv.org/abs/2311.16933.

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

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023. URL https://arxiv.org/abs/2301.04104.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

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

Kohei Honda, Naoki Akai, Kosuke Suzuki, Mizuho Aoki, Hirotaka Hosogaya, Hiroyuki Okuda, and Tatsuya Suzuki. Stein variational guided model predictive path integral control: Proposal and experiments with fast maneuvering vehicles. In *2024 IEEE International Conference on Robotics and Automation*, pp. 7020–7026, 2024. doi: 10.1109/ICRA57147.2024.10611021. URL https://doi.org/10.1109/ICRA57147.2024.10611021.

Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. VACE: All-in-One Video Creation and Editing. *arXiv preprint arXiv:2503.07598*, 2025. URL https://arxiv.org/abs/2503.07598.

Ozgur Kara, Bariscan Kurtkaya, Hidir Yesiltepe, James M. Rehg, and Pinar Yanardag. RAVE: Randomized Noise Shuffling for Fast and Consistent Video Editing with Diffusion Models. *arXiv preprint arXiv:2312.04524*, 2023. URL https://arxiv.org/abs/2312.04524.

Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2Video-Zero: Text-to-Image Diffusion Models are Zero-Shot Video Generators. *arXiv preprint arXiv:2303.13439*, 2023. URL https://arxiv.org/abs/2303.13439.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.017. URL https://doi.org/10.15607/RSS.2025.XXI.017.

Taekyung Kim, Gyuhyun Park, Kiho Kwak, Jihwan Bae, and Wonsuk Lee. Smooth model predictive path integral control without smoothing. *IEEE Robotics and Automation Letters*, 7(4): 10406–10413, 2022. doi: 10.1109/LRA.2022.3192800. URL https://doi.org/10.1109/LRA.2022.3192800.

Andrew Zou Li, Gokul Swamy, Yonatan Bisk, and Andrea Bajcsy. ELASTIC: Efficiently learning to adaptively scale test-time compute for generative control policies. *arXiv preprint arXiv:2606.31132*, 2026. URL https://arxiv.org/abs/2606.31132.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024. URL https://arxiv.org/abs/2408.03326.

Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. CogVLA: Cognition-aligned vision-language-action models via instruction-driven routing & sparsification. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c9028f7874df04843e7bf435ee4cd3c3-Abstract-Conference.html.

Han Lin, Jaemin Cho, Abhay Zala, and Mohit Bansal. Ctrl-Adapter: An Efficient and Versatile Framework for Adapting Diverse Controls to Any Diffusion Model. *arXiv preprint arXiv:2404.09967*, 2024. URL https://arxiv.org/abs/2404.09967.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023. URL https://arxiv.org/abs/2306.03310.

Youwei Liu, Jian Wang, Hanlin Wang, Beichen Guo, and Wenjie Li. Imagine-then-plan: Agent learning from adaptive lookahead with world models. *arXiv preprint arXiv:2601.08955*, 2026. URL https://arxiv.org/abs/2601.08955.

Wufei Ma, Yu-Cheng Chou, Qihao Liu, Xingrui Wang, Celso de Melo, Jianwen Xie, and Alan Yuille. SpatialReasoner: Towards Explicit and Generalizable 3D Spatial Reasoning. *arXiv preprint arXiv:2504.20024*, 2025. URL https://arxiv.org/abs/2504.20024.

Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, Karmesh Yadav, Qiyang Li, Ben Newman, Mohit Sharma, Vincent Berges, Shiqi Zhang, Pulkit Agrawal, Yonatan Bisk, Dhruv Batra, Mrinal Kalakrishnan, Franziska Meier, Chris Paxton, Alexander Sax, and Aravind Rajeswaran. OpenEQA: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16488–16498, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Majumdar_OpenEQA_Embodied_Question_Answering_in_the_Era_of_Foundation_Models_CVPR_2024_paper.html.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996. URL https://doi.org/10.1007/BF02295996.

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

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024. URL https://arxiv.org/abs/2405.12213.

Maxime Oquab, Timothee Darcet, Theo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. URL https://arxiv.org/abs/2304.07193.

Leesai Park, Keunwoo Jang, and Sanghyun Kim. CSC-MPPI: A novel constrained MPPI framework with DBSCAN for reliable obstacle avoidance. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 19645–19652, 2025. doi: 10.1109/IROS60139.2025.11246839. URL https://doi.org/10.1109/IROS60139.2025.11246839.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Santhosh K. Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, et al. Habitat-matterport 3d dataset (HM3D): 1000 large-scale 3d environments for embodied ai. In *Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/34173cb38f07f89ddbebc2ac9128303f-Abstract-round2.html.

James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2 edition, 2017. URL https://sites.engineering.ucsb.edu/~jbraw/mpc/.

Arijit Ray, Jiafei Duan, Ellis Brown, Reuben Tan, Dina Bashkirova, Rose Hendrix, Kiana Ehsani, Aniruddha Kembhavi, Bryan A. Plummer, Ranjay Krishna, Kuo-Hao Zeng, and Kate Saenko. SAT: Dynamic spatial aptitude training for multimodal language models. *arXiv preprint arXiv:2412.07755*, 2024. URL https://arxiv.org/abs/2412.07755.

Weiming Ren, Huan Yang, Ge Zhang, Cong Wei, Xinrun Du, Wenhao Huang, and Wenhu Chen. ConsistI2V: Enhancing Visual Consistency for Image-to-Video Generation. *arXiv preprint arXiv:2402.04324*, 2024. URL https://arxiv.org/abs/2402.04324.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Eren Sezener and Peter Dayan. Static and dynamic values of computation in MCTS. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence*, 2020. URL https://proceedings.mlr.press/v124/sezener20a.html.

Xiaoyu Shi, Zhaoyang Huang, Fu-Yun Wang, Weikang Bian, Dasong Li, Yi Zhang, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Motion-I2V: Consistent and Controllable Image-to-Video Generation with Explicit Motion Modeling. *arXiv preprint arXiv:2401.15977*, 2024. URL https://arxiv.org/abs/2401.15977.

Jiankai Sun, Yiqi Jiang, Jianing Qiu, Parth Talpur Nobel, Mykel J. Kochenderfer, and Mac Schwager. Conformal prediction for uncertainty-aware planning with diffusion dynamics model. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/fe318a2b6c699808019a456b706cd845-Abstract-Conference.html.

Elia Trevisan and Javier Alonso-Mora. Biased-MPPI: Informing sampling-based model predictive control by fusing ancillary controllers. *IEEE Robotics and Automation Letters*, 9(6):5871–5878, 2024. doi: 10.1109/LRA.2024.3397083. URL https://doi.org/10.1109/LRA.2024.3397083.

Dingrui Wang, Zhihao Liang, Hongyuan Ye, Zhexiao Sun, Zhaowei Lu, Yuchen Zhang, Yuyu Zhao, Yuan Gao, et al. Target-bench: Can video world models achieve mapless path planning with semantic targets? *arXiv preprint arXiv:2511.17792*, 2025. URL https://arxiv.org/abs/2511.17792.

Kailin Wang, Haoxiang Jie, Yaoyuan Yan, Jiacheng Zhou, and Zhiyou Heng. EV-WM: Event-verified world models for long-horizon robotic manipulation. *arXiv preprint arXiv:2606.13053*, 2026a. URL https://arxiv.org/abs/2606.13053.

Rui Wang, Yue Zhang, Jiehong Lin, Kuncheng Luo, Jianan Wang, Zhongrui Wang, and Xiaojuan Qi. When to trust imagination: Adaptive action execution for world action models. *arXiv preprint arXiv:2605.06222*, 2026b. URL https://arxiv.org/abs/2605.06222.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. VLA-Adapter: An effective paradigm for tiny-scale vision-language-action model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, 2026c. doi: 10.1609/aaai.v40i22.38931. URL https://doi.org/10.1609/aaai.v40i22.38931.

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

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953. URL https://doi.org/10.1080/01621459.1927.10502953.

Diankun Wu, Fangfu Liu, Yi-Hsin Hung, and Yueqi Duan. Spatial-MLLM: Boosting MLLM Capabilities in Visual-based Spatial Intelligence. *arXiv preprint arXiv:2505.23747*, 2025. URL https://arxiv.org/abs/2505.23747.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general chinese embeddings. *arXiv preprint arXiv:2309.07597*, 2023. URL https://arxiv.org/abs/2309.07597.

Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Xintao Wang, Tien-Tsin Wong, and Ying Shan. DynamiCrafter: Animating Open-domain Images with Video Diffusion Priors. *arXiv preprint arXiv:2310.12190*, 2023. URL https://arxiv.org/abs/2310.12190.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-Cache: Efficient vision-language-action manipulation via adaptive token caching. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f062da1973ac9ac61fc6d44dd7fa309f-Abstract-Conference.html.

Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-order sampling-based MPC for torque-level locomotion control via diffusion-style annealing. In *2025 IEEE International Conference on Robotics and Automation*, pp. 4974–4981, 2025. doi: 10.1109/ICRA55743.2025.11127320. URL https://doi.org/10.1109/ICRA55743.2025.11127320.

Zeji Yi, Chaoyi Pan, Guanqi He, Guannan Qu, and Guanya Shi. CoVO-MPC: Theoretical analysis of sampling-based MPC and optimal covariance design. In *Proceedings of the 6th Annual Learning for Dynamics and Control Conference*, volume 242 of *Proceedings of Machine Learning Research*, pp. 1122–1135, 2024. URL https://proceedings.mlr.press/v242/yi24b.html.

Kaining Ying, Hengrui Hu, Siyu Ren, Jiamu Li, Fengjiao Chen, Ziwen Wang, Xuezhi Cao, Xunliang Cai, and Henghui Ding. WBench: A comprehensive multi-turn benchmark for interactive video world model evaluation. *arXiv preprint arXiv:2605.25874*, 2026. URL https://arxiv.org/abs/2605.25874.

Shoubin Yu, Yue Zhang, Zun Wang, Jaehong Yoon, Huaxiu Yao, Mingyu Ding, and Mohit Bansal. When and how much to imagine: Adaptive test-time scaling with world models for visual spatial reasoning. *arXiv preprint arXiv:2602.08236*, 2026. URL https://arxiv.org/abs/2602.08236.

Bowen Yuan, Sisi You, and Bing-Kun Bao. Predict before you explore: Predictive planning with specialized memory for embodied question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 29610–29619, 2026a. URL https://github.com/yuanrr/Pred-EQA.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026b. URL https://arxiv.org/abs/2603.16666.

Haoyu Zhang, Meng Liu, Zaijing Li, Haokun Wen, Weili Guan, Yaowei Wang, and Liqiang Nie. Spatial understanding from videos: Structured prompts meet simulation data. *arXiv preprint arXiv:2506.03642*, 2025a. URL https://arxiv.org/abs/2506.03642.

Jiahan Zhang, Muqing Jiang, Nanru Dai, Taiming Lu, Arda Uzunoglu, Shunchi Zhang, Yana Wei, Jiahao Wang, et al. World-in-world: World models in a closed-loop world. *arXiv preprint arXiv:2510.18135*, 2025b. URL https://arxiv.org/abs/2510.18135.

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

Yiming Zhang, Zhening Xing, Yanhong Zeng, Youqing Fang, and Kai Chen. PIA: Your Personalized Image Animator via Plug-and-Play Modules in Text-to-Image Models. *arXiv preprint arXiv:2312.13964*, 2023. URL https://arxiv.org/abs/2312.13964.

Duo Zheng, Shijia Huang, Yanyang Li, and Liwei Wang. Learning from Videos for 3D World: Enhancing MLLMs with 3D Vision Geometry Priors. *arXiv preprint arXiv:2505.24625*, 2025. URL https://arxiv.org/abs/2505.24625.

A Experimental Protocols

Table 4 maps each task to its available evidence, comparison, and executed decision. The following benchmark sections give the complete results, models, fitting splits, and resource scopes for that protocol.

Evaluation conventions. The planning comparisons share the observation and action interfaces of each task. WBench uses the Macro-22 evaluator, and the Full-184 A-EQA comparison uses a common Gemma judge; these scores follow the definitions in Appendices F and H, rather than the official leaderboard metrics.

A.1 Common Notation

Table 3 follows one decision from observations to execution. Controller-specific symbols are defined with their equations. Lowercase c_b is a model input; uppercase $C^{(u)}$ is charged computation. The relative prediction index h differs from the observed history h_t, and the search index r differs from physical time t. The score configuration ϑ contains fitted parameters and fixed transformations, while θ parameterizes prediction. For a complete executed prefix, $m = L$.

Abbreviations used in the protocol tables. ATE denotes absolute trajectory error; RPE, relative pose error; RGB, red–green–blue image channels; LLM, large language model; VLM, vision–language model; and VLA, vision–language–action model. LLM-Match denotes LLM-based answer matching; LLM-SPL weights that answer score by path efficiency. HWM denotes the hierarchical world model setting. CAST, LIBERO, WBench, and model names retain their released names; their task interfaces and source references are given in the corresponding sections.

A.2 Task Interfaces

Planning compares candidates before physical execution; active question answering instead acquires real observations through navigation. Task sections specify the candidates eligible for selection and the computation charged to that choice.

Related maps, scenes, or initial states remain grouped during fitting and evaluation. Task-specific split counts and calibration boundaries are reported with each implementation below.

A.3 Measuring Computation

For one resource unit u and R realized search rounds, the complete ledger is

$$C^{(u)}_{\text{total}} = C^{(u)}_{\text{init}} + \sum_{r=1}^{R} \left(C^{(u)}_{r,\text{gen}} + C^{(u)}_{r,\text{score}} + C^{(u)}_{r,\text{rule}} \right) + C^{(u)}_{\text{cont}}. \tag{8}$$

Equation 8 is applied separately to each resource counter. Initial observation encoding and allocation enter $C^{(u)}_{\text{init}}$. Candidate search and all its internal predictive rollouts enter $C^{(u)}_{r,\text{gen}}$; comparisons enter $C^{(u)}_{r,\text{score}}$; the subsequent stop-or-search test enters $C^{(u)}_{r,\text{rule}}$. Computation during prefix execution, feedback processing, and downstream planning enters $C^{(u)}_{\text{cont}}$. An initial encoding shared across candidates is charged once. A scorer call that reuses cached features does not incur another copy of that encoding cost.

For transition or invocation counters, only operations that perform the counted prediction increment the corresponding ledger. For time, the declared measurement scope determines whether simulator execution and waiting are included: an end-to-end timer includes them, while a policy-computation timer can exclude them. Nested component timings partition a total; they are not added again to that enclosing total. Offline fitting and calibration are outside the evaluation-time accounting used here. The benchmark-specific scopes below determine which reported timer is used. Prediction counts depend on batching: one invocation can produce many latent transitions. Each benchmark therefore retains its native counters and timer scope; the detailed result tables report available calls, transitions, time, memory, or physical exploration costs without converting them into a shared cost unit.

Table 3: Shared notation, grouped by role. Dimensions depend on the task interface; d_a is the physical action dimension and d_z the encoded feature dimension.

Symbol	Meaning	Shape / unit
Problem and available information		
s_t, o_t, a_t	Environment state, observed input, and executed action at physical time t. Perception uses observations.	Task-defined
h_t, g	Observed action–observation history and task goal.	History; goal input
f_{enc}, z_t	Perception encoder and encoded context $f_{\mathrm{enc}}(h_t)$.	Feature vector / grid
Prediction and returned candidates		
P_θ, $\widehat{z}_h^{(b)}$	Predictive model and candidate b's predicted feature at relative step h.	d_z features
b, $\mathbf{a}_b$, $a_{b,j}$	Unique candidate index, proposed action sequence, and its action at relative position j.	Index; $N_b \times d_a$; d_a
N_b, H_b	Candidate physical action count and prediction extent; $N_b \geq L$ and $H_b = 0$ for a permitted no-imagination branch.	Integer counts
$\widehat{\mathbf{z}}^{(b)}$	Candidate b's full imagined trajectory; empty on a permitted direct branch.	H_b predicted features
ξ_b, c_b	Returned candidate record and optional computation features.	Record; task-defined vector
Candidate comparison		
$\mathcal{P}_r$, $\mathcal{I}_r$, $\mathcal{C}_r$	New, retained, and currently compared candidate records.	Finite sets
S_ϑ, $q_{r,b}$	Scorer and candidate preference score (larger is preferred).	Function; scalar
J_b, b_r^*	Minimized task cost and selected candidate index; $q = -J$ when the cost supplies the score.	Scalar; index
Bounded search		
r, R, $R_{\max}$	Search round, realized number of rounds, and finite round allowance.	Integer counts
B_r, $\mathcal{B}$	Requested computation configuration, including sampling effort and prediction extent, and its permitted set.	Task-defined configuration
$\mathcal{D}_r$	Search history: candidates, scores, and measured computation.	Recorded history
κ, d_r	Search/retention rule and its stop-or-search decision.	Rule; discrete decision
Execution and feedback		
L, m, π_{cont}	Requested prefix length, actions actually executed before prefix completion or termination ($m \leq L$), and continuation policy.	Action counts; policy
Computation accounting		
u, $C_{\mathrm{total}}^{(u)}$	A chosen resource unit and total computation in that unit.	Seconds, calls, or transitions
$C_{\mathrm{init}}^{(u)}$, $C_{\mathrm{cont}}^{(u)}$	Initial encoding/allocation and execution/feedback/continuation computation.	Resource unit u
$C_{r,\mathrm{gen}}^{(u)}$	Candidate generation, including discarded internal proposals.	Resource unit u
$C_{r,\mathrm{score}}^{(u)}$, $C_{r,\mathrm{rule}}^{(u)}$	Candidate comparison and the subsequent search decision.	Resource unit u

Table 4: Candidates, available evidence, and additional computation in each setting. Planning, feedback, and exploration instantiate different stop-or-acquire decisions; the external content-quality panels retain their own protocols.

Task	Candidate	Evidence for comparison	Additional computation
HWM	Full/split-search action plans	Observed context, goal, predicted latent futures, and computation conditions	Another world-model action search
CAST	Direct-policy or searched action trajectories	Visual history, goal, and action-conditioned predictions	Further search and prediction
Target-Bench	Videos in the combined generator pool	Initial frame, route instruction, and generated clips	Construct the six-generator pool, then score and select
SAT-v2	Answers with their reasoning	Original images, question, choices, and released task type	Additional spatial reasoning and answer comparison
A-EQA	Direct or exploration-supported answers	Question, initial views, and subsequently observed views	Navigation, new observations, and answer generation
LIBERO	Ten- or sixteen-action execution chunks	Visual/action context and fitted outcome differences	Earlier visual feedback and another policy call
ROOM	Direct or rescored navigation actions	Encoded visual history, goal, and action summaries	Action comparison with shared cached perception
WBench	Generated multi-turn interaction videos	Instructions, interaction history, and generated sequences	Video generation; quality evaluation accounted separately

A.4 Inference and Accounting

Algorithm 1 in the main paper implements generation, comparison, and stopping. A search round changes the available predictions without executing competing actions; only the selected prefix enters the environment. Equation 8 includes discarded candidates and the executed continuation in the corresponding resource ledger.

A.5 Supplementary Quality Comparisons

Figure 5 retains the WBench, Target-Bench, and external SAT comparisons. They assess generated content, selection from an existing candidate pool, and heterogeneous reasoning systems; the corresponding sections state their acquisition, model, and evaluator conditions.

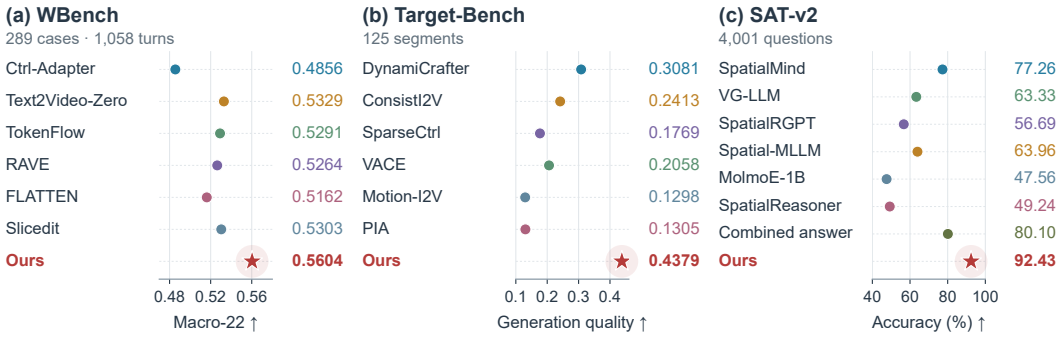

Figure 5: **Supplementary quality comparisons.** Red stars mark StopTheDream. Panels show (a) WBench Macro-22, (b) Target-Bench candidate-pool quality, and (c) SAT accuracy, with independent scales. Video editing/generation baselines and external reasoning models differ in their native interfaces and backbones. Table 11 charges the complete Target-Bench selection pool; SAT selective reasoning uses more requests than the same-backbone combined answer. These panels do not establish matched end-to-end efficiency across external models. Protocols and exact values: Appendices F, D, and I.

B SELECTIVE ADDITIONAL PLANNING ON HWM

HWM evaluates the planning procedure on 120 Diverse Maze start-to-goal tasks (Zhang et al., 2026). The following sections specify candidate allocation, outcome-supervised comparison, execution, and evaluation.

B.1 PAIRED CONTROLLER: DETAILED FORMULATION

This subsection specializes the shared procedure in Section 4 to the HWM controller. Its scoring set contains either the two split-search candidates or the incumbent and one addition. A singleton full-search route executes directly. The paired scorer uses terminal success as its task-specific target.

PLANNING WITH A WORLD MODEL

Using the history, actions, and predictions defined in Section 4, the HWM search returns one candidate for comparison. Internal MPPI samples are distinct from this returned candidate.

A search call returns one candidate, its predictions, and computation features c_b:

$$\xi_b = \mathrm{Plan}(h_t, g; B) = (\mathbf{a}_b, \widehat{\mathbf{z}}^{(b)}, c_b). \tag{9}$$

The budget B sets internal sample counts. The planner uses its predictive objective to generate the candidate; the separately fitted scorer predicts execution outcomes.

Each requested candidate rollout completes before comparison. The selected prefix $\mathbf{a}_{b,0:L-1}$ is executed before real feedback and continuation, as detailed in Appendix B.4.

ALLOCATING SEARCH

Allocating the initial samples. Let B_{f} and B_{s} denote full and split-search sample budgets, with two split searches using approximately one full budget in total. The allocator A_ω reads z_t, g, and computation conditions c_{init} available before candidate generation. Its scores predict the full-search outcome and whether either split candidate succeeds:

$$(q_{\mathrm{full}}, q_{\mathrm{split}}) = A_\omega(z_t, g, c_{\mathrm{init}}), \qquad \text{split if } q_{\mathrm{split}} - q_{\mathrm{full}} \geq \delta. \tag{10}$$

Here ω denotes the allocator parameters. The tolerance δ is selected on separate calibration states. The split target describes the opportunity to choose a successful alternative; the scorer must still identify it. The initial decision precedes candidate generation and uses only observed context and development computation statistics. Appendix B gives the level-specific budgets, and Appendix B.4 defines the pre-generation conditions.

The resulting candidates are obtained by

$$\begin{aligned} \text{full:} \quad & \xi_{\mathrm{f}} = \mathrm{Plan}(h_t, g; B_{\mathrm{f}}), \\ \text{split:} \quad & \xi_i, \xi_j = \text{two calls to } \mathrm{Plan}(h_t, g; B_{\mathrm{s}}). \end{aligned} \tag{11}$$

The full route selects its candidate directly; the split route compares both candidates with the paired scorer.

SCORING AND SELECTING CANDIDATES

Scoring the generated candidates. The scorer estimates terminal success after executing a candidate's prefix and finishing with the prescribed continuation. Its shared encoder E_ϕ summarizes the actions, finite imagined trajectory, observed context, goal, and computation features. A joint comparison module F_ϕ then reads both candidate representations:

$$\begin{aligned} u_b &= E_\phi(z_t, g, \mathbf{a}_{b,0:L-1}, \widehat{\mathbf{z}}^{(b)}, c_b), \\ (v_i, v_j) &= F_\phi(u_i, u_j, g), \qquad (p_i, p_j) = (\sigma(v_i), \sigma(v_j)). \end{aligned} \tag{12}$$

u_b is an encoded candidate, v_b a logit, σ the sigmoid, and ϕ the combined parameters of E_ϕ and F_ϕ. Together these modules instantiate scorer S_ϑ from Section 4.2. Sharing the encoder makes the two representations comparable; joint comparison lets their differences inform each score. The vector c_b describes how the candidate was generated and is not a cost penalty. The outputs are pair-conditioned success scores, so changing the competing candidate requires scoring both again. Appendix B.4 gives the encoder and comparison architecture.

Choosing or searching again. For the split candidates (i, j), let $b^* = \arg\max_{b\in\{i,j\}} p_b$ be the current choice. Ranking alone would select it even if both candidates looked unsuccessful. The controller therefore generates one new full-budget candidate k when

$$\max(p_i, p_j) < \eta, \qquad \eta = 0.5. \tag{13}$$

The fixed threshold η makes it a practical rule: low scores suggest poor available options, but do not estimate how much a new search will help. Candidate k and retained choice b^* enter *Re-score*, which recomputes both scores using Equation 12 before selection. The configuration permits at most one full-budget addition after a split route. This allowance is reserved in advance: the controller tests the score threshold without a separate runtime affordability predicate. Extra search therefore remains bounded even if the new scores remain low. The single-full-search route bypasses this comparison and addition.

The selected prefix $\mathbf{a}_{b^*,0:L-1}$ enters the environment, followed by real feedback and the prescribed continuation. Every generated candidate contributes to cost, including discarded ones. Algorithm 3 specifies this single initial decision followed by the protocol's fixed continuation.

Outcome Supervision and Pairwise Ranking

Finite imagined trajectories do not reveal whether an episode will ultimately succeed. We obtain supervision by executing candidate prefixes from restored development states and finishing each episode with a common continuation policy π_{cont}:

$$y_b = \text{Success}(\text{Execute}(s_t, \mathbf{a}_{b,0:L-1}; \pi_{\text{cont}})) \in \{0, 1\}. \tag{14}$$

Here Execute restores the physical state s_t and its associated observed history, then includes the continuation to termination; Success applies the task's terminal success criterion. Holding the start and continuation fixed makes the candidates' outcomes comparable, connecting finite predictions to consequences beyond the prediction horizon.

Algorithm 2 specifies the three within-state pairs and separate allocator targets. Map-grouped fitting and calibration counts appear in Appendix B.

Outcome prediction and contrastive ranking. For one fully valid pair, before aggregation across the weighted sampler, outcome supervision teaches whether either candidate succeeds under the specified continuation:

$$\mathcal{L}_{\text{out}} = -\frac{1}{2} \sum_{b\in\{i,j\}} \left[y_b \log p_b + (1 - y_b) \log(1 - p_b) \right]. \tag{15}$$

The pairwise logistic term $\mathcal{L}_{\text{pair}}$ emphasizes which execution has the better outcome. These signals serve different decisions: relative ordering selects a candidate, while outcome scores indicate whether the available choices warrant further search. Here *contrastive* denotes this outcome comparison. The complete objective and all auxiliary terms are defined once in Appendix B.5.

Training boundaries. The scorer updates its encoder and relation network; the separately fitted allocator uses y_{f} and $y_i \vee y_j$ with actions and predictions masked. Perception, planner, and world model remain frozen; evaluation fixes all learned weights and thresholds.

B.2 Training and Planning Procedures

Algorithm 2 in the main paper fits the HWM scorer and allocator using the Plan interface in Equation 9. Physical restoration is used only for matched offline executions.

The root set $\mathcal{T}_0$ contains restorable development states with their associated visible information; $\mathcal{T}_{\text{pair}}$ and $\mathcal{T}_{\text{alloc}}$ are the resulting scorer and allocator datasets. They are distinct from the online search history $\mathcal{D}_r$. Each candidate is generated from the same root, and each training execution restores that root before acting. The terminal label is one if the environment's success criterion is reached during the completed execution, and zero otherwise. Algorithm 2 is entirely offline; evaluation does not execute competing candidates to inspect their outcomes.

Algorithm 3 Outcome-Supervised Selection with One Optional Additional Search

Require: Observed history h_t, context $z_t = f_{\text{enc}}(h_t)$, goal g; fixed planner
Require: Fitted allocator/scorer; budgets $B_{\text{f}}, B_{\text{s}}$; prefix L; thresholds δ, η
Require: Fixed conditions c_{init}; allowance for one extra full search on the split route
Ensure: Selected prefix, its real feedback, and prescribed continuation

$(q_{\text{full}}, q_{\text{split}}) \leftarrow A_\omega(z_t, g, c_{\text{init}})$
if $q_{\text{split}} - q_{\text{full}} \geq \delta$ **then** ▷ Two smaller searches
  $\xi_i, \xi_j \leftarrow$ two separate calls to $\text{Plan}(h_t, g; B_{\text{s}})$
  $(p_i, p_j) \leftarrow \text{Score}(i, j)$; $b^* \leftarrow \arg\max_{b \in \{i,j\}} p_b$
  if $\max(p_i, p_j) < \eta$ **then** ▷ One permitted addition
    $\xi_k \leftarrow \text{Plan}(h_t, g; B_{\text{f}})$
    $(p_{b^*}, p_k) \leftarrow \text{Score}(b^*, k)$ ▷ Recompute both
    $b^* \leftarrow \arg\max_{b \in \{b^*, k\}} p_b$
  end if
else ▷ One full search; no comparison
  $\xi_{\text{f}} \leftarrow \text{Plan}(h_t, g; B_{\text{f}})$; $b^* \leftarrow \text{f}$
end if
Execute $\mathbf{a}_{b^*, 0:L-1}$ until completion or termination; record actual steps $m \leq L$
Form updated history h_{t+m} from executed actions and real observations
if the task has not terminated **then**
  Hand off to π_{cont}
end if

Appendix B.5 defines targets, masks, and reductions for both fits. Appendix B.4 gives the allocator's fixed decision-time computation conditions.

Score ties in Algorithm 3 retain the first candidate, which is the incumbent during recomparison.

B.3 Complete-Policy Results

All rows share the predictive model, coordinate probes, action bounds, success predicate, and 500-step evaluation limit. Table 5 gives the exact totals for the main-paper planning comparison.

Table 5: HWM planning results. All rows use the same 120-task denominator. Transitions count prediction in every generated proposal and continuation; time additionally includes allocator and scorer work.

Method	Successes	Latent transitions	Time (s)
MPOPIS (Asmar et al., 2023)	93	7,413,541,960	245,294.2
SVG-MPPI (Honda et al., 2024)	53	3,081,743,520	119,816.8
CoVO-MPC (Yi et al., 2024)	113	747,475,280	25,036.9
DIAL-MPC (Xue et al., 2025)	82	1,620,717,720	58,012
CSC-MPPI (Park et al., 2025)	105	1,088,987,773	37,941
Feedback-MPPI (Belvedere et al., 2026)	111	741,661,960	28,041.5
Smooth MPPI (Kim et al., 2022)	109	741,661,960	17,358.6
HWM (Zhang et al., 2026)	105	741,661,960	31,765.1
StopTheDream	**117**	**384,386,362**	**18,934**

B.4 Planner and Scorer Configuration

Allocation and execution. The allocator compares the predicted full-search outcome with the predicted success union of two split searches. The allocation tolerance is -0.0043622, selected on 40 calibration tasks from ten maps. A full-budget proposal uses 2,001 high-level and 501 low-level samples on easy and medium tasks, or 4,001 and 1,001 on hard tasks. Each smaller proposal uses 1,000 and 250 samples on easy and medium tasks, or 2,000 and 500 on hard tasks. For each paired comparison, both searches start from the same observed state and share the horizon specified for that call and difficulty.

The two allocations have approximately equal initial sample counts. The split target is the success union of its alternatives; deployed success also depends on the scorer selecting the successful one.

In Algorithm 3, the additional-search threshold is $\eta = 0.5$, fixed on the development panel. The selected candidate supplies four physical actions, followed by one standard half-budget hierarchical continuation and the released 15 final transition steps.

The planners use a shared HWM interface with method-specific optimization mechanisms. MPOPIS retains cross-entropy-method (CEM) proposal fitting, optimized path-integral updates, and shrinkage covariance estimation, elite selection, and joint covariance but omits the author's early stop. SVG-MPPI maps Stein transport from a one-dimensional controller to the two-dimensional HWM hierarchy and rescales its temperature. CoVO-MPC replaces the full curvature matrix with a low-rank finite-difference approximation. DIAL-MPC maps spline controls and annealed updates to HWM; CSC-MPPI uses the HWM signed-distance field, normalized clustering, and a cap-3 projection. Feedback-MPPI retains rollout differentiation and feedback correction at Level 1. All methods are evaluated through this common observation, action, and dynamics interface.

DIAL-MPC. Each action sequence is parameterized by five uniformly spaced knots (the authors' $H_{\text{node}} = 4$ convention) and expanded to the active horizon with a quadratic spline. Each update evaluates the shared HWM sample count K plus the current nominal trajectory. The first controller call uses ten updates and subsequent calls use two. For knot $j \in \{0, \dots, 4\}$ at update r, perturbations scale by $0.9^{4-j}0.5^{r}$; the first knot is held fixed. The path-integral temperature is $0.06\ \mathrm{std}(J_r)$, and each update is projected to the common HWM action bounds.

CSC-MPPI. CSC-MPPI draws the shared K HWM trajectories at each planning call. A differentiable maze signed-distance field, rasterized at 24 pixels per cell and queried bilinearly, constrains Level-1 physical-action trajectories with margin 0.12. Primal-dual projection uses chunks of 128, at most three iterations, primal and dual step sizes 0.6 and 0.4, and per-action gradient scales 0.1 and 1.0; it stops when every trajectory in a chunk is feasible. Both hierarchy levels use density-based spatial clustering of applications with noise (DBSCAN) with $\epsilon = 1.35$ and minimum samples 2 on median/median-absolute-deviation (MAD)-standardized mean-action and rollout-cost features. Each cluster contributes an MPPI-weighted representative, and the representative with the lowest re-evaluated HWM cost is executed.

HWM model and evaluation. The predictive backbone is the released hierarchical latent world model of Zhang et al. (2026). Its low-level model predicts the effect of physical actions, while its high-level model predicts coarser latent changes used by hierarchical planning. Both provide the action-conditioned interface in Equation 1; MPPI is the action-search optimizer, not another predictive model. All rows load the released hierarchical world-model checkpoint and receive 98×98 red–green–blue (RGB) observations. The released evaluator declares success at route distance at most 1.0 and permits at most 500 environment steps. The low-level predictor accepts two-dimensional actions. The planner bounds the low-level step norm to $[0, 1]$ and the high-level latent step to $[-2.5, 2.5]$, then uses the released normalizer for environment execution. MPPI uses temperature $\lambda = 0.0025$ and Gaussian noise scales 5 and 10 at Levels 1 and 2. Level 1 plans 50 latent steps; Level 2 uses horizons 25, 35, and 47 for easy, medium, and hard tasks. The complete method executes 265, 365, and 485 steps, respectively, including the 15 final transition steps. The default comparison budget uses 500/500/1,000 Level-1 samples and 2,000/2,000/4,000 Level-2 samples. CoVO-MPC estimates a low-rank curvature update from eight antithetic probe pairs at scale 0.1 and clips curvature eigenvalues to $[10^{-3}, 10^{3}]$.

Scorer architecture and supervision. Training contains 100 restorable decision states from 25 development maps. Each state provides one full-budget and two half-budget candidates. Candidate labels are terminal outcomes after executing their four actions and an ordinary continuation. The allocator compares the full-budget outcome with the success union of the two half-budget candidates. The paired scorer learns from all 300 pairs formed within these states. Training, allocator calibration, and official testing use 25, ten, and 20 disjoint map layouts, respectively; test outcomes are not scorer inputs or training labels.

Both networks use width 192, eight attention heads, two spatial-transformer layers, three temporal-transformer layers, and two task-cross-attention layers. Spatial pooling produces 16 tokens with 16

channels. The scorer receives current and goal features, low- and high-level latent trajectories, four candidate actions, and the computation-condition vector defined below. A candidate-relation transformer compares the two summaries before the success head. Its pair-conditioned sigmoid scores rank candidates and trigger additional search.

Candidate computation features. Each generated candidate receives a computation-condition vector $c_b \in \mathbb{R}^5$ with the following entries. These are computation conditions attached to an already generated candidate; the visual and predicted-trajectory tokens are separate inputs.

Entry	Meaning	Normalization
$c_{b,1}$	Remaining fraction of the 500-step episode; t is the root's physical step index	$\max(0, 500 - t)/500$
$c_{b,2}$	Search type	Full: 0; smaller: 1
$c_{b,3}$	Level-1 predicted transitions used to produce candidate b	Count / 10,000
$c_{b,4}$	Level-2 predicted transitions used to produce candidate b	Count / 100,000
$c_{b,5}$	Measured generation time for candidate b	Seconds / 10

The physical step and search type are known before generation; the last three entries are measured from the generated proposal. The controller makes its learned decision at the episode root, so its first entry is one.

Conditions before generation. The allocator uses fixed development conditions c_{init}, with one row for each initial allocation. The allocator query sets both rows to 10,020 Level-1 and 188,094 Level-2 transitions. It supplies development time medians of 5.157683650846593 seconds for the full route and 6.862892043311149 seconds for the split route. After the same normalization, the two rows are

$$\begin{aligned} c_{\text{init,f}} &= (1, 0, 1.002, 1.88094, 0.5157683650846593), \\ c_{\text{init,s}} &= (1, 1, 1.002, 1.88094, 0.6862892043311149). \end{aligned} \tag{16}$$

The subscripts f and s identify full and split allocation. The allocator masks candidate actions and imagined trajectories while retaining these conditions. Thus its query uses preset development statistics, not the realized cost of a future candidate. Training retains the computation conditions of its development records; inference replaces them with the fixed rows above.

Hierarchical prediction steps and physical actions have different time scales: level-specific latent horizons are given above, while the executed prefix contains four physical actions.

Scorer inputs use padding masks and temporal subsampling to bound sequence length. These operations affect the encoded candidate representation; the planner still generates its complete trajectory. Proposal runtime is a measured scorer input.

Training uses AdamW with learning rate 3×10^{-4}, weight decay 0.01, batch size 32, dropout 0.1, gradient clipping at 1.0, and bfloat16 autocasting. The allocator trains for 153 epochs and retains epoch 112; the scorer trains for 400 epochs and retains epoch 384, using zero-based epoch indices. The exact six-term objective, target normalizations, and two supervision groups are defined in Appendix B.5. Inference uses success scores and the fixed acquisition rule; auxiliary quantile heads do not select the action.

Feedback after the selected prefix. The controller restores the physical root and planner snapshot before each proposal and again before executing the final selection. After the four physical actions, it captures the returned observation, the environment's current state, the number of steps already used, and the remaining step allowance. The state snapshot restores the evaluator; the predictive model receives the associated observation through its usual perception interface. The selected candidate's imagined endpoint is not substituted for that real observation.

The downstream evaluator resumes the ordinary hierarchical planner with the smaller sample budget and the released 15 final transition steps. The learned allocator and addition test run only at the initial decision. Prefix and continuation rewards form one episode record under the 500-step cap; all proposals, comparisons, and downstream planning enter the cost ledger.

B.5 SCORER OBJECTIVE

The scorer combines execution-outcome supervision with pairwise ranking and auxiliary prediction targets:

$$\begin{aligned} \mathcal{L}_\phi &= \mathcal{L}_{\text{decision}} + \mathcal{L}_{\text{aux}}, \\ \mathcal{L}_{\text{decision}} &= \mathcal{L}_{\text{out}} + 2\mathcal{L}_{\text{pair}}, \\ \mathcal{L}_{\text{aux}} &= \mathcal{L}_{\text{gain}} + 0.2\mathcal{L}_{\text{first}} + 0.2\mathcal{L}_{\text{end}} + 0.1\mathcal{L}_{\text{cal}}. \end{aligned} \tag{17}$$

Outcome scores select candidates and trigger search; the four auxiliary predictions contribute only during fitting.

Table 6: **Scorer supervision terms.** Each term uses the sampler's importance weights and its own valid-element denominator. Empty masks contribute zero.

Term	Weight	Target, mask, and reduction
$\mathcal{L}_{\text{out}}$	1	Binary cross-entropy on success logits over valid candidates
$\mathcal{L}_{\text{pair}}$	2	Logistic ranking of success logits; both candidates valid and labels unequal
$\mathcal{L}_{\text{gain}}$	1	Pinball loss at quantiles $0.1, 0.5, 0.9$ for ordered failure-to-success label
$\mathcal{L}_{\text{first}}$	0.2	Smooth-L1 on first-success step /500; successful candidates with finite targets only
$\mathcal{L}_{\text{end}}$	0.2	Smooth-L1 on endpoint distance /10; valid candidates with finite targets
$\mathcal{L}_{\text{cal}}$	0.1	Mean squared probability error of the success and lower-success heads against the same label

Outcome loss supplies the decision scale; ranking distinguishes candidates with unequal outcomes. Table 6 defines the auxiliary targets and their normalizations.

Outputs attached to the candidate evidence. Besides the selection logit v_b, the network predicts a raw margin m_b, a nonnegative normalized first-success estimate $\widehat{T}_b$, and a nonnegative normalized endpoint-distance estimate $\widehat{D}_b$. The latter two outputs use $\text{softplus}(u) = \log(1 + \exp u)$, where u is the corresponding raw head output. The auxiliary success score is

$$p_b^- = \sigma(v_b - \text{softplus}(m_b)) \le p_b. \tag{18}$$

The minus superscript denotes the lower auxiliary prediction; it is not a statistical confidence bound. The probability-error contribution for candidate b is $\frac{1}{2}[(p_b - y_b)^2 + (p_b^- - y_b)^2]$.

An ordered pair also has gain predictions $\widehat{G}_{ij}^{(\tau)}$ at quantile levels τ. The network obtains them by adding nonnegative softplus increments to the first raw quantile. With gain target $G_{ij} = \mathbf{1}[y_i = 0, y_j = 1]$ and residual $e = G_{ij} - \widehat{G}_{ij}^{(\tau)}$, its pinball penalty is $\rho_\tau(e) = e(\tau - \mathbf{1}[e < 0])$. These targets teach outcome-related structure in the candidate representation. Only p_b enters the selection and acquisition rules; $\widehat{T}_b$, $\widehat{D}_b$, p_b^-, and the gain quantiles remain training outputs. The first-success and endpoint targets use the scales and validity masks in Table 6.

Ranking and ordered gain. For an ordered pair (i, j) with $y_i \neq y_j$, the ranking penalty is

$$\ell_{\text{pair}}(i, j) = \log(1 + \exp\{-\text{sign}(y_i - y_j)(v_i - v_j)\}). \tag{19}$$

Pairs with equal terminal outcomes do not contribute to this term. The auxiliary gain label is $\mathbf{1}[y_i = 0,\ y_j = 1]$ for the pair order. It is not the expected value of a future, ungenerated search. The quantile head fits this binary target using pinball loss. The probability-error term averages the two success-head squared errors and does not establish calibration under distribution shift.

Training reductions and data construction. For values ℓ_{nb}, importance weights w_n, and validity mask m_{nb}, the implementation returns $\sum_{n,b} w_n m_{nb} \ell_{nb} / \sum_{n,b} w_n m_{nb}$. Each loss uses its own mask; the quantile term also averages over its three quantile positions. The sampler stratifies ordered pairs by their outcome pattern and returns the weights used in this reduction. Unsuccessful first-success targets are excluded, not assigned a successful terminal step.

The scorer receives the three pairs (full, first split), (full, second split), and (first split, second split) from each of 100 development states. The allocator receives 100 full-versus-split-union records. Its split success is $y_i \vee y_j$; first success is the earliest successful split execution, and endpoint distance is the minimum among successful split candidates, or among both when neither succeeds. Training uses the same six-term loss for separate allocator and scorer fits, with the allocator's candidate actions and trajectory tokens masked.

Allocator inputs. The allocator retains measured development computation features during fitting and uses the fixed pre-generation conditions in Equation 16 at inference.

B.6 Candidate Identity and Information Flow

Figure 6 exposes the action–future records used in Equation 12. When candidate j is retained and k is generated, the new comparison uses (j, k) and recomputes both scores. The offline row distinguishes the allocator's supervision from the scorer's supervision; the shared encoder E and relation module F denote E_ϕ and F_ϕ, respectively.

B.7 Parameter Sensitivity

Threshold sensitivity fixes learned weights and varies the additional-search decision on the same 120 tasks.

Search thresholds. The grid is $\eta \in \{0, 0.25, 0.4, 0.5, 0.6, 0.75, 1\}$ for the strict test $\max(p_i, p_j) < \eta$, with $\eta = 0.5$ as the reference. At zero, addition is disabled; at one, a saturated score can still prevent addition. Table 2 reports the additional-search scan on the same 120 tasks. The complete numerical scan appears in Table 2 in the main paper. No model is retrained between thresholds.

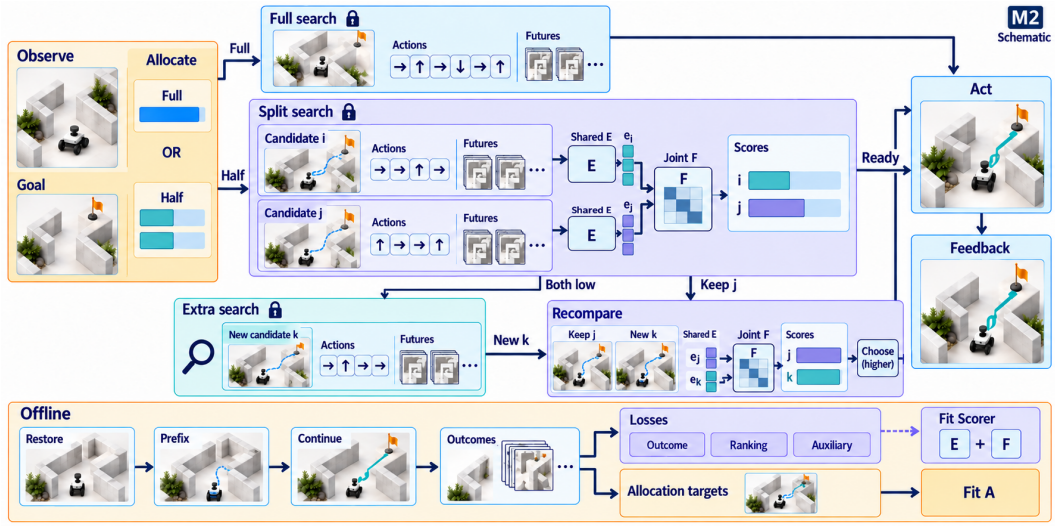

Figure 6: **Candidate records, recomparison, and training destinations.** The supplementary view exposes the action–future pairing inside each record, the shared encoder E, the joint comparison module F, and the separate fits of A and (E, F). Candidate j illustrates the retained incumbent and k the additional proposal; either can win the new comparison. The offline row connects executed outcomes to the fitted modules. Scenes and score bars are schematic.

C Adaptive Planning on CAST

CAST evaluates continuous trajectory quality on 1,000 visual trajectories. The controller follows the allocation, comparison, and optional additional-search structure in Section 4, with a separately trained continuous scorer. It also permits a direct-policy candidate whose predicted trajectory is empty ($H_b = 0$). After additional search, the retained candidate and the new proposal are rescored together. CAST uses its own visual predictive model and independently trained continuous scoring modules.

C.1 Observation, Action, and Prediction Interfaces

The evaluation uses validation trajectories 0–999, with trajectories 1000–1252 reserved for method selection and placeholder examples excluded. Each method receives four proposals from an Octo-Small policy (Octo Model Team et al., 2024) trained on the original, counterfactual, and atomic training mixture. The predictive backbone follows the frame-causal PiJEPA architecture (Chahe & Zhou, 2026) with frozen DINOv2 visual features (Oquab et al., 2023). The policy supplies action proposals and the action-conditioned predictor supplies latent futures.

Actions have four body-frame coordinates. Policy proposals are decoded to physical actions and encoded using the predictor's single-step first- and ninety-ninth-percentile bounds; the same bounds decode the clipped evaluation actions. Absolute trajectory error (ATE) is the primary endpoint, and scale-normalized relative pose error (RPE) is complementary. Both are computed from decoded trajectories by the common evaluator, with smaller values preferred.

C.2 Planner Comparisons

The six sampling-based planners share policy proposals, predictive dynamics, terminal latent goal cost, action bounds, and evaluation observations. Table 7 specifies their optimization mechanisms. The PiJEPA reference uses the same observation and proposal interfaces and is trained locally from the published model specification.

For DIAL-MPC, the baseline action sequence has seven four-dimensional steps. Four annealing rounds use 32 candidates each, with standard deviation $0.30(0.55)^{j}(0.90)^{6-h}$ at round $j \in$

Table 7: CAST planner implementations with shared policy proposals, dynamics, and evaluation. Method names identify the source algorithms; the second column describes the implementation.

Method	Optimization mechanism
MPOPIS (Asmar et al., 2023)	elite fitting with a joint covariance over horizon and action
Smooth MPPI (Kim et al., 2022)	input lifting and temporally smooth control perturbations
SVG-MPPI (Honda et al., 2024)	Stein guide transport, guide covariance, and weighted sampling
CoVO-MPC (Yi et al., 2024)	objective curvature controls the sampling covariance
Biased-MPPI (Trevisan & Alonso-Mora, 2024)	policy-prior mixture sampling with the nominal MPPI update
DIAL-MPC (Xue et al., 2025)	direct seven-step actions, horizon and iteration annealing, and normalized softmax updates

$\{0, 1, 2, 3\}$ and action index $h \in \{0, \ldots, 6\}$. Each population includes the four policy proposals, the current mean, and 27 perturbed sequences. Costs are centered and divided by their population standard deviation before a softmax update with temperature 0.25. The final choice minimizes goal cost between the best sampled candidate and the final weighted mean.

C.3 Trajectory Quality and Prediction Cost

Table 8: **CAST quality and prediction cost on 1,000 trajectories.** ATE is primary and RPE is complementary; calls and latent outputs are means per trajectory.

Method	ATE	RPE	Calls/traj.	Outputs/traj.
SVG-MPPI	1.5733	0.5100	35	588
MPOPIS	1.9446	0.7433	42	938
DIAL-MPC	1.5083	0.5412	42	938
CoVO-MPC	1.7149	0.6164	49	1,337
Biased-MPPI	1.5518	0.5370	42	938
Smooth MPPI	2.0944	0.6602	42	938
PiJEPA (local)	1.4224	0.4484	28	896
StopTheDream	**1.3629**	**0.4080**	**12.85**	**397**

All 1,000 trajectories contribute to the aggregate metrics. Relative to PiJEPA, StopTheDream reduces RPE by 9.01% and latent outputs by 55.69%; Section 5 reports ATE and predictor-call comparisons. Predictor calls count invocations, while latent outputs count the predicted states produced by those calls. These counters retain their separate units under batched generation.

D Controlled Imagination Depth

The depth studies intervene on prediction extent under matched conditions. They motivate measuring the return from computation; they do not determine the adaptive controller's stopping rule or isolate its efficiency.

D.1 Planning Success versus Prediction Horizon

The primary curve evaluates sixteen horizons from 3 to 48 predicted transitions, in increments of three, on 90 difficult closed-loop continuation tasks. Three frozen task sources contribute 30 tasks each. Every depth shares the map, start, goal, world-model checkpoint, downstream planner, 1,000 candidates per call, and the same prefix of the same 50-step noise sequence. Using a shared noise prefix avoids replacing the sampling randomness together with the horizon; the intended intervention is the number of imagined steps. The main-paper depth panel plots all sixteen successful-task counts. A shape-preserving cubic interpolant connects them only as a visual guide, without adding observations or estimating uncertainty. Table 9 also reports the corresponding rates.

Table 9: Primary HWM depth curve on 90 paired tasks.

Imagination depth	Successes	Success rate
3	51/90	56.67%
6	53/90	58.89%
9	54/90	60.00%
12	57/90	63.33%
15	61/90	67.78%
18	68/90	75.56%
21	**74/90**	**82.22%**
24	67/90	74.44%
27	62/90	68.89%
30	61/90	67.78%
33	62/90	68.89%
36	63/90	70.00%
39	64/90	71.11%
42	65/90	72.22%
45	63/90	70.00%
48	66/90	73.33%

Table 9 peaks at 74/90 at depth 21; counts at depths 24–48 fluctuate between 61 and 67. These aggregates establish non-monotonic performance, without identifying a universal horizon, prediction-error mechanism, or paired uncertainty.

Comparison across settings. The HWM curve is a 90-task closed-loop success sweep. The CAST and Target-Bench panels in Figure 2 plot aggregate horizon responses under their respective metrics. They characterize the observed curve shapes; they do not estimate paired uncertainty or a common optimal horizon.

Separate timing replication. An independent matched comparison at depths 20 and 26 measures success rates of 74.44% and 66.67%, while mean high-level planning time increases from 160.80 to 208.72 milliseconds, or 29.80%. This two-horizon comparison is separate from the sixteen-horizon success sweep.

D.2 Horizon Responses Across Settings

Table 10 gives every CAST and Target-Bench coordinate used in Figure 2; the HWM coordinates appear in Table 9. The settings use different performance metrics and horizon scales, so each panel keeps its native axis. CAST absolute trajectory error improves from 1.73 at horizon 1 to 1.54 at horizon 4, then worsens to 1.93 at horizon 7. The reported Target-Bench score rises from 0.0392 at horizon 3 to 0.0428 at horizon 12, then falls to 0.0395 at horizon 21. The curves connect observed aggregates with shape-preserving cubic interpolation only as a visual guide.

Table 10: CAST and Target-Bench aggregate coordinates plotted in Figure 2. Lower CAST ATE and higher Target-Bench score are better.

CAST horizon	ATE ↓	Target horizon	Score ↑
1	1.73	3	0.0392
2	1.65	6	0.0397
3	1.60	9	0.0413
4	**1.54**	12	**0.0428**
5	1.62	15	0.0417
6	1.78	18	0.0403
7	1.93	21	0.0395

D.3 Video Generators for Semantic Path Planning

Table 11 compares StopTheDream and six video generators on 125 semantic path-planning segments from Target-Bench (Wang et al., 2025); its candidate-pool calibration uses cross-fitting within the evaluated pool. The generators are DynamiCrafter (Xing et al., 2023), ConsistI2V (Ren et al., 2024), VACE (Jiang et al., 2025), SparseCtrl (Guo et al., 2023), PIA (Zhang et al., 2023), and Motion-I2V (Shi et al., 2024).

Table 11: Full-125 Target-Bench candidate-pool comparison of StopTheDream and six video generators. Quality uses cross-fitting within the candidate pool. †: the proposed row includes the summed generation cost of all six generators used to build its pool; selection and scoring add further cost.

Method	Venue	Quality ↑	Generation GPU s
DynamiCrafter	ECCV 2024	0.3081	10,220.00
ConsistI2V	TMLR 2024	0.2413	2,756.74
VACE	ICCV 2025	0.2058	87,520.58
SparseCtrl	ECCV 2024	0.1769	6,982.00
PIA	CVPR 2024	0.1305	1,021.76
Motion-I2V	SIGGRAPH 2024	0.1298	5,054.00
StopTheDream	–	**0.4379**	$113{,}555.08^{\dagger}$

StopTheDream selects from the combined outputs of all six generators on all 125 segments; it is a candidate-pool selection result, not a new generator evaluated at one baseline's generation cost. Quality is 0.4379 versus DynamiCrafter's 0.3081, an absolute difference of 0.1298 (42.13% relative). Cross-fitting and evaluation both operate within this fixed six-generator pool.

Pool construction and selection cost. The complete pool is required before selection. Summing the six reported generation totals gives $C_{\text{pool}} = 113{,}555.08$ GPU seconds. This is a derived construction cost, not a separately measured end-to-end runtime. In the same GPU-second accounting, $C_{\text{ours}} = C_{\text{pool}} + C_{\text{select}} \geq 113{,}555.08$, where $C_{\text{select}} \geq 0$ includes encoding, scoring, and selection. The construction term exceeds each individual generator's listed cost: the quality gain is obtained after paying for the complete candidate pool. Offline fitting and cross-fitting costs are separate from this evaluation-time lower bound.

E Task-Conditioned Feedback on LIBERO

LIBERO evaluates instructed manipulation under visual feedback (Liu et al., 2023). Both studies use the frozen FasterWAM release from World-in-World (Zhang et al., 2025b), receiving the instruction and two 224×224 camera views and producing sixteen seven-dimensional actions. Choosing ten or sixteen actions changes the execution prefix, not the prediction extent. The Long study below learns this decision from paired outcomes.

E.1 Four-Suite Results and Evaluation Scope

The supplementary four-suite schedule selects sixteen actions for LIBERO-10 and Object, and ten for Goal and Spatial, using complete executions on the official initial states. Evaluation uses those same states: this is a calibration-pool comparison. Every method receives 2,000 episodes (ten tasks and fifty initial states per suite), the same reset wait, and native success checks. FasterWAM-based rows share one checkpoint, observations, action normalization, and continuation interface; other policies retain their released backbones.

Released controls are VLA-Adapter Pro (Wang et al., 2026c), VLA-Cache on OpenVLA-OFT (Xu et al., 2025; Kim et al., 2025), CogVLA (Li et al., 2025), and $\pi_{0.5}$ (Black et al., 2025). Calls count feedback requests, whose internal computation differs across backbones. Process time includes work outside the execution loop; it is distinct from the policy and loop times below.

The proposed schedule solves 495, 493, 498, and 495 episodes on LIBERO-10, Goal, Object, and Spatial, with 7,819, 5,769, 4,364, and 5,493 calls. Relative to ten-step FasterWAM, the totals give

Table 12: Suite-calibrated four-suite LIBERO comparison. Every row uses 2,000 episodes. VLA-Adapter and OpenVLA-OFT calls are reconstructed from complete episode replays; VLA-Cache and $\pi_{0.5}$ calls are counted during execution. The first three published policies execute eight actions per request; $\pi_{0.5}$ executes five. Calls measure feedback frequency; computation within a call differs across backbones.

Controller	Correct / total	Success (%) ↑	Policy calls ↓	Process (s)
FasterWAM, 10-step feedback	1,975 / 2,000	98.75	31,438	47,120
Fixed 16-step reuse	1,967 / 2,000	98.35	20,022	49,234
VLA-Adapter Pro	1,955 / 2,000	97.75	39,099	44,784
VLA-Cache on OFT	1,945 / 2,000	97.25	39,665	48,185
OpenVLA-OFT	1,944 / 2,000	97.20	39,563	45,380
CogVLA	1,928 / 2,000	96.40	39,688	45,897
$\pi_{0.5}$	1,934 / 2,000	96.70	64,489	51,036
StopTheDream	**1,981 / 2,000**	**99.05**	**23,445**	**41,612**

six additional successes, 25.42% fewer calls, and 11.69% less complete-process time. VLA-Adapter and OpenVLA-OFT calls are reconstructed as $\lceil T/8 \rceil$ from replayed control frames; VLA-Cache and $\pi_{0.5}$ count requests during execution.

Table 13: Measured outcomes and episode costs for released LIBERO policies. Each suite contains 500 official episodes; the aggregate contains 2,000. Policy time is client latency for $\pi_{0.5}$ and synchronized inference for CogVLA. Loop time includes observation preparation, policy, simulator, and controller work; it excludes loading and video export. Calls and times are summed over the reported episodes.

Method	Suite	Correct / total	Calls	Controls	Policy (s)	Loop (s)
$\pi_{0.5}$	Spatial	486 / 500	10,989	54,019	1,025.36	3,268.40
$\pi_{0.5}$	Object	494 / 500	14,326	70,676	1,326.35	3,239.44
$\pi_{0.5}$	Goal	487 / 500	11,711	57,625	1,075.40	3,000.42
$\pi_{0.5}$	Long	467 / 500	27,463	136,445	2,580.49	6,008.14
$\pi_{0.5}$	All four	1,934 / 2,000	64,489	318,765	6,007.61	15,516.41
CogVLA	Spatial	487 / 500	6,768	52,384	2,329.59	5,131.48
CogVLA	Object	494 / 500	8,779	68,459	2,777.84	5,275.14
CogVLA	Goal	480 / 500	7,359	57,132	2,340.88	4,823.61
CogVLA	Long	467 / 500	16,782	132,532	6,351.83	10,921.56
CogVLA	All four	1,928 / 2,000	39,688	310,507	13,800.14	26,151.80

E.2 Learned Execution Horizons on LIBERO-Long

The study controls how long a selected action block executes before feedback. Fixed short execution applied 10 actions, and fixed long execution applied all 16 actions. Both paths used the same backbone checkpoint, language task, simulator initialization, normalized action block, continuation policy, 700-step limit, and native success predicate.

The study contains 500 official initial states and two action sources per state, yielding 1,000 paired closed-loop paths. The two action sources from one state remain together throughout model fitting, calibration, policy selection, and final testing. Reported learned-policy values are complete aggregates over three evaluation runs.

Table 14 compares success and policy calls. The learned rule improves success over every non-oracle control, while fixed sixteen-step execution and PlanCP remain lower-cost points. Section 5 discusses the feedback trade-off.

The task pattern agrees across both action sources. Tasks 0, 1, 8, and 9 favor 16-step execution in both replications, while the other six favor earlier feedback. Replacing the gain distribution with a point estimate lowers success by 1.17 points.

Table 14: Task-conditioned execution horizons on LIBERO-Long. The evaluation unit is a set of 1,000 paired paths, with both action sources grouped by initial state. Learned-policy results aggregate three complete evaluation runs. Calls are reported per path; 20.853 corresponds to 20,853 calls per 1,000 paths.

Controller	Success (%) ↑	Calls/path ↓
Fixed 10-step feedback	91.60	27.619
Fixed 16-step feedback	88.10	**18.670**
Training-fold task prior	92.40	22.252
Causal-attention controller (Wang et al., 2026b)	89.83	23.641
Imagine-then-Plan-style horizon (Liu et al., 2026)	90.63	22.404
PlanCP-style absolute interval (Sun et al., 2023)	90.17	19.961
EV-WM-style event verification (Wang et al., 2026a)	88.10	**18.670**
StopTheDream	**92.83**	20.853
Retrospective outcome oracle	97.40	18.971

F WBench Full-289 Comparison

WBench evaluates instructed multi-turn interactions generated by a video world model (Ying et al., 2026). The comparison covers all 289 released cases and 1,058 ordered turns. Table 15 compares all methods with the same Macro-22 evaluator.

The baselines are Ctrl-Adapter (Lin et al., 2024), Text2Video-Zero (Khachatryan et al., 2023), TokenFlow (Geyer et al., 2023), RAVE (Kara et al., 2023), FLATTEN (Cong et al., 2023), and Slicedit (Cohen et al., 2024b).

Scoring protocol. Macro-22 is the equal-weight mean of the 22 metric outputs defined by the local evaluator. All seven methods are scored on the same 289 cases and 1,058 turns. The proposed score is 0.560351, 0.027460 above the strongest listed baseline, Text2Video-Zero (5.15% relative).

Table 15: WBench Full-289 under the Macro-22 evaluator. Generation time covers video generation, separately from metric evaluation.

Method	Macro-22 ↑	Generation (s)
Ctrl-Adapter	0.485613	8,736.674
Text2Video-Zero	0.532891	19,197.720
TokenFlow	0.529144	48,249.262
RAVE	0.526407	42,496.426
FLATTEN	0.516225	140,261.202
Slicedit	0.530326	187,101.788
StopTheDream	**0.560351**	18,402.539

These video generation and editing baselines compare content quality under a common evaluator while retaining their native conditioning and editing interfaces. Generation takes 18,402.539 seconds for StopTheDream, 4.14% less than Text2Video-Zero's 19,197.720 seconds, alongside the 5.15% relative Macro-22 improvement.

G Supplementary ROOM Results

ROOM is the Craftium navigation task used by Horizon Imagination (Cohen et al., 2026). Strict success requires termination inside the goal radius. The proposed student distills the frozen teacher, caches each newly observed frame once, and selectively compares navigation actions. Both the teacher and student use zero online future-generation calls. The comparison therefore measures observation reuse and action computation.

The student uses 103,517 replay states split by episode into 81,475 training, 11,716 calibration, and 10,326 validation states. Its action comparison uses 112 terminal executions from 12 development

roots; leave-one-root-out calibration fixes the decision criterion. All 100 evaluation roots are disjoint from these roots and share the initial states, goals, action limits, and success checks across methods. Additional controls are PPO (Schulman et al., 2017), REM (Cohen et al., 2024a), DreamerV3 (Hafner et al., 2023), IRIS (Micheli et al., 2022), and DIAMOND (Alonso et al., 2024).

Table 16: ROOM strict Full-100 outcomes and evaluation costs on the same frozen initial states.

Method	Strict success ↑	Env. steps	Policy time (s)	Frames encoded
Horizon Imagination	100/100	574	40.854	2,918
PPO-CNN	31/100	69,068	85.483	–
REM (ICML 2024)	31/100	69,068	138.908	–
DreamerV3	100/100	488	36.015	–
IRIS	100/100	1,645	288.564	–
DIAMOND	99/100	6,149	258.153	–
StopTheDream	**100/100**	**573**	**22.075**	**573**

Policy time sums warmed decisions on the same accelerator type, excluding environment-server waiting, loading, and reset; shared GPU allocations had concurrent workloads. PPO and REM receive 100,000 training interactions, and their failures often reach the 1,000-step cap, so episode totals do not isolate per-step speed. The proposed and Horizon Imagination policies both solve every root; their lengths match on 73 roots, with 12 shorter and 15 longer episodes for the student. Its one-step aggregate reduction explains little of the latency difference: mean policy time per step falls from 0.071174 to 0.038525 seconds. DreamerV3 uses fewer environment steps at the same success count.

Table 17: Policy-computation profile for the matched Horizon Imagination and StopTheDream policies.

Measure	Horizon Imagination	StopTheDream
Supported operations (trillions)	33.255	6.837
Peak allocated memory (MiB)	575.946	475.726
Parameters (millions)	107.829	113.215
Frames encoded	2,918	573

Supported-operation totals extrapolate two matched root profiles over the observed episode lengths and 151 relation evaluations; they are not direct accelerator-wide measurements. Cached history reduces encoded frames from 2,918 to 573. The following interventions retain the cached policy's task interface and each solve all 100 roots. No-cache recomputes retained observations, no-sharing re-encodes the relation inputs, and dense querying evaluates relations at every decision. Latency comparisons use matched action sequences: 99 roots for no-cache and no-sharing, and 100 for dense querying.

Table 18: Component costs relative to the complete cached policy. All conditions solve 100 of 100 tasks.

Policy computation	Operations	Policy time
Complete cached policy	reference	reference
Recompute retained history	+301.38%	+61.45%
Re-encode relation input	+0.0009%	+1.56%
Evaluate relations at every step	+0.30%	+23.96%

Caching accounts for the largest computation reduction. Dense relation querying adds little arithmetic but increases latency by 23.96%; these controls isolate executed computation rather than the optimality of the invocation criterion.

H Active Embodied Question Answering

The LLaVA-based study uses LLaVA-OneVision-Qwen2-7B (Li et al., 2024) to decide whether an answer from initial views warrants opening the Pred-EQA exploration branch. Tables 19–21 give its

utility and resources; the Gemma-based quality comparison uses a separate common-judge protocol below.

The selector estimates exploration gain from initial evidence, ranks the evaluation pool, and closes a development-calibrated fraction. Every opened branch supplies its full exploration answer. Acquisition changes the physical state and is charged separately from pre-execution imagined search.

The evaluation uses 184 public OpenEQA questions from 57 HM3D scenes (Majumdar et al., 2024; Ramakrishnan et al., 2021). Every matched policy begins at the released pose, collects the same center view and six auxiliary views, and uses the same LLaVA-OneVision-Qwen2-7B backbone. Pred-EQA supplies the complete exploration branch with its released prediction, correction, mapping, memory, and answer interfaces (Yuan et al., 2026a). We score all three policies with the same model-based judge. It gives each answer a score $s \in \{1, \ldots, 5\}$, which is mapped to $25(s-1)$ on a scale from 0 to 100. The authors' separately reported Pred-EQA result is 53.3 LLM-Match and 48.5 LLM-SPL; all differences below use the shared model-based judge.

The development set contains 41 questions from five scenes. The selected controller closes 7 of 41 development branches, a 17.07% fraction, and reaches 54.268 compared with 53.049 for full exploration. During evaluation, 16 matching development questions are forced open. The same closure fraction closes the 28 lowest-valued questions among 168 questions from 52 unseen scenes. Every opened branch retains the complete Pred-EQA answer.

Ranking by predicted exploration value allocates the fixed closure fraction to questions for which exploration is estimated to add least. The controller makes a pool-level allocation before exploration; reported utility measures aggregate answer quality over that question pool.

Table 19: LLaVA-based A-EQA task utility and exploration on all 184 questions.

Policy	Explorations	LLM-Match	LLM-SPL
Direct answer	0	42.935	42.935
Full Pred-EQA	184	48.913	48.025
StopTheDream	156	48.913	48.025

Table 20: Complete LLaVA-based A-EQA resource use, including direct answering and selective acquisition.

Resource	Direct answer	Full Pred-EQA	StopTheDream
Process time (s)	918.75	31,626.23	26,120
Tokens	581,736	57,461,016	53,043,718
Model requests	184	8,332	7,702
Input images	1,288	33,500	31,847
Travel (m)	0.00	724.14	663.55

Table 21: LLaVA-based A-EQA quality and token use on 168 questions from 52 unseen scenes.

Evaluation	LLM-Match	LLM-SPL	Tokens
Full Pred-EQA	48.512	47.639	53,752,080
StopTheDream	48.512	47.639	49,284,137

The full evaluation reduces process time by 17.41%, while the unseen-scene subset retains the same aggregate answer quality and uses fewer tokens.

The selector closes 28 branches before any navigation, mapping, exploration request, exploration image, or long-context answer call occurs. All 184 direct answers remain charged. The online controller adds 92.20 seconds of selector execution and reaches a peak allocated accelerator memory of 2.61 GB. The controller selects the 28 questions before reading their exploration observations. The resulting cost reductions therefore measure work removed from the executed system.

H.1 Selector Architecture and Training

The exploration selector distinguishes what can be predicted before opening a branch from what could be assessed after observing it. Separate summaries of initial views and exploration views keep these inputs distinguishable. The unopened branch uses a missing-evidence representation, so an empty exploration input is not treated as an observed empty scene.

The selective acquisition controller used in Table 19 uses frozen 1,024-dimensional BGE-large-en-v1.5 text embeddings and 512-dimensional CLIP ViT-B/32 image embeddings (Xiao et al., 2023; Radford et al., 2021). Its five nodes represent the question, seven-view initial state, direct answer, potential exploration answer, and observations collected by that exploration. An unopened exploration has no realized observation input. A two-layer temporal transformer summarizes a state-query token plus the seven initial views. A second two-layer transformer summarizes an evidence-query token, one branch token, and up to 40 temporally sampled exploration views. Four relation-attention layers then produce five ordered exploration-gain quantiles at probabilities $\{0.10, 0.25, 0.50, 0.75, 0.90\}$, one positive-gain probability, and three log-cost predictions for process time, tokens, and distance. All learned blocks use width 192, eight attention heads, feed-forward width 768, and dropout 0.10. The controller has 3,867,465 trainable parameters. Training uses batch size 16, learning rate 2×10^{-4}, weight decay 10^{-3}, 180 epochs, gradient-norm cap 1.0, and 15% view dropout. This acquisition rule reads the prior branch value, closes a calibrated fraction, and adopts every opened Pred-EQA branch; its posterior blocks remain part of the shared training architecture but do not reject an exploration answer.

Calibration and fitting. The exploration selector's quantiles are ordered by construction. Losses and normalization statistics are fit only on training groups. The selective A-EQA controller obtains one prediction for each development question from a model trained on the other four scenes. Calibration selects the largest closure count that reaches the stronger endpoint globally and within every scene while retaining at least 90% of beneficial explorations. The primary result closes 7 of 41 development branches. Its 17.07% closure fraction maps to 28 closures among the 168 unseen-scene evaluation questions. The chosen model is refit on all 41 development questions for 180 epochs before evaluation.

H.2 Same-Evaluator Full-184 Comparison

This panel uses Google Gemma 4 31B IT (Google DeepMind, 2026) for answer generation and to judge all six baselines and our policy.

Table 22: A-EQA comparison on all 184 questions using the same Gemma judge for every method.

Method	Gemma score (%)	Non-null answers
3D-Mem	0.815217	172/184
Explore until Confident	0.679348	143/184
GraphEQA	0.951087	173/184
EmbodiedQA	9.103261	154/184
Map-EQA	11.005435	175/184
Mind Palace	15.489130	182/184
StopTheDream	**45.380435**	167/184

All seven rows retain the same 184 question IDs; null answers score zero and no scorer errors occurred. The proposed row scores 45.380435 versus Mind Palace's 15.489130. A common judge does not equate the methods' information, exploration, or training pipelines. This panel is separate from the matched LLaVA exploration comparison; neither score is directly comparable to the original Pred-EQA report of 53.3 LLM-Match under its Qwen3-VL protocol.

I SELECTIVE SPATIAL REASONING ON SAT

SAT evaluates spatial reasoning from chronological images and a two-choice question (Ray et al., 2024). The measured policies use original images and textual reasoning. Table 23 reports the full-set outcomes.

I.1 EVALUATION AND DECISION RULE

The full-scale protocol evaluates all 4,001 released SAT-v2 questions. It contains 1,336 action-consequence, 647 action-sequence, 592 goal-aim, 647 object-movement, and 779 perspective questions. The input to every case is its chronological image sequence, question, two answer choices, and released question-type label; this public metadata is visible at inference and determines whether the policy requests an additional perspective analysis. The offline evaluator compares the parsed choice with the released target index. Every question remains in the denominator.

A direct branch answers from the observations, while a structured branch first describes the relevant state transition and then selects a choice. Each answer remains attached to its evidence source. When both choices are parseable and disagree, an answer scorer compares whether their described changes agree with the images, follow the stated action, and support the selected choice. This produces the combined answer baseline used below. The scorer retains the chosen answer together with the observations and reasoning that support it. Action-consequence questions retain their native branch whenever the scorer would introduce a switch.

Only the 779 perspective questions receive an additional SpatialMind spatial analysis. The official prompt constructor receives the original selected frames and asks for a text-based top-down object layout, spatial reasoning, and an answer. This full-scale instance does not generate new images or render a new camera view. The other four task families return the combined answer baseline without this additional analysis. When the base and SpatialMind produce different parseable choices, an answer-comparison scorer selects the answer whose spatial explanation better matches the observed objects, geometry, and offered choices. Every other path retains the executable base.

The released question type, rather than a learned gain prediction, triggers the additional perspective analysis. This task-dependent rule does not establish that other families would not benefit.

The resource protocol counts requests, returned tokens, and model time across Direct, Structured, answer comparison, SpatialMind, and final answer selection. All stages use Google Gemma 4 31B IT (Google DeepMind, 2026) with deterministic decoding. Parallel wall time is recorded separately from the sum of per-request model time.

Quality and request cost. The full-set evaluation contains 3,698 correct answers and 9,853 model requests. Relative to the combined baseline's 3,205 correct answers and 8,543 requests, this is 493 additional correct answers at 1,310 additional requests (15.33%). It is a quality–computation trade-off.

The external panel includes Spatial-MLLM (Wu et al., 2025), VG-LLM (Zheng et al., 2025), SpatialRGPT (Cheng et al., 2024), and SpatialReasoner (Ma et al., 2025), alongside SpatialMind and MolmoE.

MolmoE is the sixth external row. It uses frozen MolmoE-1B-0924 weights with chronological released images, the released shuffled choice order, deterministic decoding, and a fixed A/B parser. It supplies a general-purpose vision–language comparison under the same SAT input and choice interface. All 4,001 questions are evaluated; 151 unparsed outputs count as incorrect. The external panel compares complete systems with heterogeneous released backbones, model sizes, training procedures, and native cost units. The same-Gemma comparison to the combined answer base provides the within-backbone quality and request-cost comparison.

The selective aggregate improves answer accuracy over SpatialMind and the combined baseline (Section 5). The selective-policy comparison uses accuracy and total model requests.

Table 23: Full SAT-v2 comparison on all 4,001 questions. The external panel includes six vision–language reasoning methods. Costs retain their reported native units. The selective row reports higher accuracy at higher request cost than the combined baseline.

Method	Correct	Accuracy	Primary executed cost
SpatialMind (Zhang et al., 2025a)	3,091	77.2557%	4,001 model requests
Spatial-MLLM	2,559	63.9590%	518.76 model s
VG-LLM	2,534	63.3342%	1,466.72 model s
SpatialRGPT	2,268	56.6858%	218.40 model s
SpatialReasoner	1,970	49.2377%	6,519.85 model s
MolmoE-1B-0924 (Deitke et al., 2025)	1,903	47.5631%	4,588.09 model s
Combined answer base	3,205	80.1050%	8,543 model requests
StopTheDream selective acquisition	**3,698**	**92.4269%**	9,853 model requests
Supervised native control	3,256	81.3797%	2,278 model requests; 257 training labels
Clarified native control	3,299	82.4544%	2,278 model requests; same fitted scorers

I.2 Supervised Reference Controls

The two native controls in Table 23 use 257 training labels for camera-motion and target-bearing scorers, with the same Gemma backbone handling abstentions and remaining questions. Clarification uses 32 separate development questions and changes only the 779 perspective responses. This additional supervision differs from the selective policy's question-type routing.

Clarification increases the native control from 3,256 to 3,299 correct at 2,278 requests, with all 87 unparsed final answers counted as incorrect. The perspective subgroup changes from 443 to 486 correct, including both improved answer completion and spatial-decision changes. All 207 distance-change targets in that subgroup are "Closer", limiting interpretation of that particular improvement.

The clarified control consumes 1,794,899 tokens and 35,388.169 summed request seconds. These durations include service queueing and are not end-to-end wall time or GPU computation. Its reduced request count therefore does not imply reduced elapsed latency.

J Statistical Summaries from Aggregate Outcomes

This section reports calculations determined by the aggregate outcomes in the task tables. HWM and CAST report single-run evaluations; LIBERO-Long reports three-run aggregates without confidence intervals. Wilson intervals (Wilson, 1927) describe individual success proportions under a binomial model; they do not quantify a paired difference or variation across training seeds. The calculations below use individual tasks or questions as independent units and are unadjusted for multiple comparisons. Shared maps, scenes, or initial states require grouped uncertainty analysis for population-level conclusions.

Table 24: Success proportions and nominal Wilson 95% intervals calculated from reported counts. Rates and interval endpoints are percentages.

Setting	Method	Correct / total	Success (%) [95% CI]
HWM	StopTheDream	117/120	97.50 [92.91, 99.15]
	CoVO-MPC	113/120	94.17 [88.45, 97.15]
	HWM planner	105/120	87.50 [80.40, 92.28]
SAT	Selective acquisition	3,698/4,001	92.43 [91.57, 93.21]
	Combined answer base	3,205/4,001	80.10 [78.84, 81.31]

Paired-test bounds from marginal counts. For two methods evaluated on the same n cases, let x and y be their success counts and a their unknown number of shared successes. Then

$$\max(0, x + y - n) \le a \le \min(x, y), \qquad b = x - a, \quad c = y - a,$$

where b, c count successes unique to the first and second method. For $b + c > 0$, the nominal two-sided exact McNemar test (McNemar, 1947) has value $\min\{1, 2\Pr[\text{Binomial}(b + c, 1/2) \le$

$\min(b, c)]\}$. Enumerating all feasible integer a gives Table 25. These are envelopes of possible test values, not measured paired-test results or confidence intervals; feasible values within an envelope are discrete.

Table 25: Feasible ranges of nominal exact McNemar p values determined by aggregate counts, with rounded endpoints. Δ is the first method's success rate minus the second's, in percentage points.

Comparison	Δ (pp)	Feasible p range
HWM: StopTheDream vs. CoVO-MPC	+3.33	$[0.125000,\ 0.343750]$
HWM: StopTheDream vs. Feedback-MPPI	+5.00	$[0.031250,\ 0.145996]$
HWM: StopTheDream vs. Smooth MPPI	+6.67	$[0.007813,\ 0.057373]$
HWM: StopTheDream vs. HWM planner	+10.00	$[0.000488,\ 0.007538]$
HWM: $\eta = 0.5$ vs. $\eta = 0$	+0.83	1
LIBERO suites: ours vs. ten-step FasterWAM	+0.30	$[0.031250,\ 0.451381]$
SAT: selective vs. combined answer	+12.32	$[7.821\times10^{-149},\ 1.441\times10^{-51}]$

The HWM–CoVO difference does not reject equal marginal success at the nominal 0.05 level for any feasible pairing. This does not establish equivalence or non-inferiority. HWM versus the original HWM planner and SAT versus the combined answer have upper bounds below 0.05 under this task-level test. The four-suite LIBERO calculation describes its calibration-pool comparison, not held-out generalization.

Descriptive quality and computation differences. HWM's 117 versus 113 successes accompany 48.58% fewer latent transitions and 24.38% less planning time. Changing η from zero to 0.5 adds one aggregate success and 1,135,002 latent transitions (0.296%). CAST's mean ATE difference from PiJEPA is $1.3629 - 1.4224 = -0.0595$, with 54.11% fewer predictor calls. These are differences of reported aggregates, not confidence intervals or estimates of seed variability. Component attribution requires the corresponding controlled interventions.