RESOLUTION COLLAPSE IN LATENT-SPACE CEM: DIAGNOSIS, CAUSAL EVIDENCE, AND A FIX

Anonymous authors
Paper under double-blind review

ABSTRACT

Planning with a learned latent world model usually means using the cross-entropy method (CEM) to search over candidate action sequences and ranking them by how close the predicted final latent state lands to the goal. We show that this ranking step fails in a specific and measurable way. Across four continuous-control environments built on a joint-embedding predictive world model, the correlation between predicted and true candidate quality collapses to chance exactly at the terminal frame CEM optimizes, even though the same predictor ranks candidates well several steps earlier. We call this resolution collapse, and we observe it on every environment we test. A set of horizon-manipulation experiments then shows that the collapse tracks CEM's own optimization target rather than a fixed prediction horizon. Stretching the planning horizon from five steps to eight steps moves the collapse point outward by exactly the same amount, and the correlation at whichever frame is currently terminal stays pinned near chance at every horizon we try. This motivates a direct fix. Ranking candidates on a frame the model can still resolve, instead of the terminal one, raises planner success rate by three to fifteen points across our environments, following an inverted-U pattern in which asking too early actively hurts and asking too late is where the original failure lives. Two independent random-cost controls, matched in magnitude to the real cost, rule out that this gain is a side effect of simply adding another term to the objective. Replacing the resolution-matched cost with noise of identical spread collapses success rate by twenty-three to ninety-four points, and an additive version of the same idea beats a matched-noise version by eleven to thirteen points at $p < 0.001$. Finally, we give a rule that selects the reporting frame directly from the model's own resolution curve, without ever consulting success rate, and show it lands within two points of the oracle frame found by an exhaustive sweep, on every environment. Together, these results turn an invisible bottleneck in latent-space planning into something a practitioner can measure, predict, and correct at no extra test-time cost.

1 INTRODUCTION

A growing family of control systems plans by search inside a learned latent world model. An encoder maps an observation to a latent state, a predictor rolls that state forward under a candidate action sequence, and a search procedure such as the cross-entropy method (CEM) samples many such sequences, scores each by how close its predicted final state lands to a goal, and refits its sampling distribution to the best-scoring elite. This recipe, or a close variant of it, underlies PlaNet (Hafner et al., 2019), Dreamer (Hafner et al., 2023), MuZero (Schrittwieser et al., 2020), TD-MPC and TD-MPC2 (**?**Hansen et al., 2024), and the recent line of decoder-free joint-embedding world models that plan directly on top of a frozen or lightly finetuned visual backbone, including DINO-WM (Zhou et al., 2025) and V-JEPA 2 (Assran et al., 2025). The appeal is obvious. Once a predictor generalizes across a family of tasks, a generic search procedure turns it into a controller for any of them, with no task-specific reward learning and no policy network to train.

The entire recipe rests on one quiet assumption: that the predictor can tell the search procedure which of its candidates is better. We show that this assumption fails in a specific, measurable, and previously undocumented way. We study a decoder-free joint-embedding world model, LEWM

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.

(Maes et al., 2026), planning with CEM across four continuous-control environments that span discrete navigation, a low-dimensional arm, and two contact-rich manipulation tasks. For each frame the predictor emits along a rollout, we re-execute the candidate action sequence in the real environment and compare the model's own predicted cost against the true outcome, restricted to the elite set CEM actually keeps. The correlation is not merely weak at the terminal frame, the one frame every one of these systems reports as its planning objective. It is at chance, on every environment we test, while the very same predictor ranks candidates several steps earlier with correlations of 0.5 to 0.9. We call this *resolution collapse*: the model loses its ability to order outcomes exactly at the point in the rollout the planner has been built to trust the most.

A natural worry is that this is simply prediction error compounding with horizon length, a well known concern in model-based control (**?**Talvitie, 2017). A direct causal manipulation rules this out: if collapse tracked physical time, stretching the CEM horizon should push the collapse point out by exactly the added steps. Instead, extending the horizon moves the collapse point to the new terminal frame every time, and the frame that was previously terminal, now mid-rollout, recovers a strong ranking correlation. This holds at four horizon settings. Resolution collapse is not a property of physical time into the future; it is a property of which frame the search procedure currently treats as the objective.

This diagnosis immediately suggests a fix with no added architecture and no retraining: rank candidates on a frame the model can still resolve, rather than on the terminal frame the task nominally cares about. We show this simple change raises planner success rate by three to fifteen points across our four environments, following a clear inverted-U shape over which frame is queried. Asking too early is not merely unhelpful, it actively hurts, dropping success rate by as much as twenty-two points below the terminal-frame baseline on one environment. The improvement lives in a specific window, and finding that window is the point of the diagnosis above.

Because any added cost term can look like a win for reasons that have nothing to do with the information it carries, we hold our own fix to that standard with two independent random-cost controls, each swapping the real term for a synthetic one of identical spread but no relationship to the goal. Real signal wins decisively in both, and the gain is not a magnitude artifact. We then ask whether the right frame to query can be chosen without ever looking at success rate, since a rule tuned against the metric it predicts would not transfer to a new checkpoint. A rule built entirely from the model's own resolution curve recovers within two points of the oracle frame on every environment.

Our contributions are as follows.

- **We identify and causally validate resolution collapse.** Using a candidate re-execution protocol that isolates ranking ability from every other source of planner error, we show that elite-set ranking correlation falls to chance exactly at the terminal frame of latent-space CEM planning, on four diverse continuous-control environments, and we confirm with horizon manipulation across four planning horizons that the collapse tracks the search procedure's optimization target rather than a fixed physical prediction horizon (Section 4).
- **We introduce resolution-matched planning and verify the gain is real.** Querying the predictor at a frame it can still resolve, instead of the terminal frame, raises planner success rate by three to fifteen points, with a full characterization of the inverted-U trade-off over which frame to ask. Two structurally different spread-matched random-cost controls confirm the gain comes from the information the frame carries, not from adding an extra term to the objective (Section 5).
- **We give a success-rate-blind rule for choosing the query frame.** Built entirely from the model's own resolution curve, the rule never consults planner outcome and still matches an exhaustive oracle sweep on every environment tested, making the fix usable without a success-rate search on a new checkpoint or task (Section 5).

2 Related Work

Latent world models for control. Learning a compact latent dynamics model and planning inside it has been a productive direction since PlaNet showed that a recurrent deterministic-and-stochastic state-space model could support cross-entropy planning directly from pixels (Hafner et al., 2019). Dreamer and DreamerV3 replaced online planning with policy learning inside imagined rollouts

at scale (Hafner et al., 2023). MuZero folded a learned model into Monte Carlo tree search and matched engines given the true rules of the game (Schrittwieser et al., 2020), later formalized by the value equivalence principle (Grimm et al., 2020). TD-MPC and TD-MPC2 pushed decoder-free implicit latent models to over a hundred continuous-control tasks, planning short-horizon trajectories with MPPI and bootstrapping the tail with a learned value function (**?**Hansen et al., 2024). A parallel line replaces pixel reconstruction with joint-embedding prediction, arguing that predicting in representation space is both easier to optimize and more robust to irrelevant visual detail (**?**). I-JEPA and V-JEPA established this objective for images and video (Assran et al., 2023; Bardes et al., 2024), and V-JEPA 2 extended it to action-conditioned planning on a real robot arm (Assran et al., 2025). DINO-WM shows that frozen pretrained visual features are sufficient to train a transition model that plans zero-shot with CEM (Zhou et al., 2025). We build on LEWM (Maes et al., 2026), a joint-embedding predictive architecture trained end to end from pixels with a single regularizer, and ask a question orthogonal to all of this work: once such a model exists, does the search procedure built on top of it actually receive a signal it can use to choose between the candidates it has narrowed down to.

Sampling-based planning on learned models. The cross-entropy method iteratively refits a sampling distribution to the elite fraction of scored candidates and remains the default search procedure for learned dynamics models (**?**). PETS paired CEM with an ensemble of probabilistic dynamics models and matched model-free asymptotic performance at a fraction of the sample budget (Chua et al., 2018). Model predictive path integral control derives a softmax-weighted variant from an information-theoretic argument and has been deployed for aggressive real-world driving (Williams et al., 2017). iCEM improves CEM's sample efficiency with temporally correlated action noise and elite memory (Pinneri et al., 2020), and POPLIN folds a policy network into the sampling distribution to warm-start the search (Wang & Ba, 2020). Nearly all of this work optimizes search efficiency given an informative cost, treating elite-refitting itself as a solved, interchangeable back end. We instead audit that assumption: whether the cost computed from the model's rollout can distinguish between the candidates the refit step keeps. Our elite-width ablation (Appendix E) shows the failure we identify is specific to refitting search, since a non-refitting variant of the same planner does not exhibit it, suggesting the diagnosis applies broadly across the CEM, MPPI, and iCEM family rather than to one solver.

3 PRELIMINARIES

World model. We use LEWM (Maes et al., 2026), a joint-embedding predictive architecture trained end to end from pixels with a single next-embedding prediction loss and a Gaussian-norm regularizer that keeps the latent space from collapsing during training. An encoder E maps an observation to a latent state z, and an autoregressive predictor P rolls a latent state forward under a sequence of actions, one action block at a time, to produce a sequence of predicted latents $\hat{z}_1, \ldots, \hat{z}_H$ with no pixel decoder anywhere in the loop. We use the publicly released pretrained checkpoint for every environment and never fine-tune it. Every quantity in this paper is computed from this one fixed model, so all of our findings are about the planner built on top of it, not about how the model was trained.

Planner. We plan with the cross-entropy method (CEM), the default search procedure for this model and the shared ancestor of the sampling-based planners discussed in Section 2. At each control step, CEM samples N candidate action sequences from a diagonal Gaussian, rolls each one forward through P to obtain $\hat{z}_1, \ldots, \hat{z}_H$, scores each candidate with a cost function J evaluated on the predicted latents and a goal embedding g, keeps the k lowest-cost candidates as the elite set, refits the sampling distribution to the elite set's mean and variance, and repeats for a fixed number of iterations before executing the first action block of the best sequence found. We use the paper's default elite fraction ($k = 30$ of $N = 300$), which our own elite-width ablation in Appendix E confirms is where this planner's ranking failure has the most influence on the final decision. The default cost is $J_{\text{last}}(c) = \|\hat{z}_H(c) - g\|^2$, the squared distance between the predicted terminal latent and the goal embedding, which we refer to throughout as the terminal-frame cost.

Environments. We evaluate on four continuous-control environments spanning discrete navigation, low-dimensional articulated control, and contact-rich manipulation, shown in Figure 7:

Table 1: Elite-set ranking fidelity ($n = 50$ episodes, 30 elite candidates each). The model's predicted terminal cost is only weakly informative over the full candidate pool and is at chance inside the elite set on every environment.

Environment	ρ (full pool)	ρ (elite set)	elite true-cost spread
TWOROOM	0.287 ± 0.051	0.055 ± 0.034	60% of full pool
REACHER	0.332 ± 0.050	0.059 ± 0.031	84% of full pool
PUSHT	0.307 ± 0.039	0.046 ± 0.030	44% of full pool
CUBE	0.193 ± 0.039	0.084 ± 0.045	91% of full pool

TWOROOM (top-down point-mass navigation through a gap in a dividing wall), REACHER (a two-link planar arm reaching a target end-effector position), PUSHT (pushing a T-shaped block into a target pose with a circular pusher), and CUBE (a simulated Franka-style manipulator relocating a small cube). Every environment shares the same protocol: an episode is an (init, goal) pair of real frames from a held-out expert dataset, the planner gets a fixed budget of $H = 5$ latent-prediction steps at a 5 environment-step action block (25 environment steps of lookahead, matching the dataset's goal offset), and success is the environment's own binary goal-reaching criterion. Planner comparisons use $n = 200$ held-out pairs per environment; resolution measurements that re-execute every candidate use $n = 50$ pairs, roughly $6\times$ more expensive per episode. Full details are in Appendix A.

The certified action prior. Every arm is compared against, and where relevant combined with, an L2 penalty on the action sequence itself, $J_{\mathrm{L2}}(c) = \|c\|^2$, which requires no model prediction at all. We calibrate its weight λ once per environment so its across-candidate spread equals a fixed fraction $r = 0.1$ of the main cost's own spread, measured on a held-out warm-up set disjoint from the evaluation pairs: $\lambda = r \cdot \mathrm{spread}(J_{\mathrm{main}})/\mathrm{spread}(J_{\mathrm{L2}})$. This keeps the prior a fixed, weak tie-breaker rather than a free hyperparameter that could be tuned to favor any given arm, and lets us compare arms whose main cost has a very different scale on equal footing. We refer to $J_{\mathrm{last}} + \lambda J_{\mathrm{L2}}$ as the *certified L2 baseline*, the reference point every result in this paper is measured against.

Measuring resolution. To ask whether the model's predicted cost can actually rank the candidates a planner will choose between, we take the candidate pool from the final CEM iteration of a real planning episode, re-execute a subset of those candidates in the environment from the same starting state, and compare each candidate's model-predicted cost against its true cost, $J_{\mathrm{true}}(c) = \|z_H(c) - g\|^2$, computed from the latent the model actually encodes after the candidate is executed. We report the Spearman rank correlation (Spearman, 1904) between predicted and true cost, computed separately over the full candidate pool and over only the elite set CEM keeps, which we call the *elite Spearman* ρ_{elite}. This is the quantity CEM's refit step actually depends on: it does not need J_{model} to match J_{true} in scale or even in absolute value, only to agree on which elite candidates are better than which others. Measuring ρ_{elite} at every predicted frame $t \in \{1, \ldots, H\}$, rather than only at the terminal frame H, gives the *resolution curve* that is the central diagnostic tool of Section 4.

4 RESOLUTION COLLAPSE: DIAGNOSIS AND CAUSAL EVIDENCE

4.1 THE MODEL ORDERS SITUATIONS, NOT THE PLANS INSIDE THEM

We first ask, at the coarsest level, whether the predicted terminal cost agrees with the true terminal cost at all. Over the full candidate pool of a real planning episode, the two agree with a modest but consistent positive correlation, $\rho \in [0.19, 0.33]$ across our four environments (Table 1, full statistics in Appendix C). Restricted to the 30 candidates CEM actually keeps as its elite set, the same correlation falls to $\rho \in [0.05, 0.08]$, indistinguishable from chance, on every environment. This is not because the elite candidates all lead to the same outcome. Their true-cost spread is 44 to 91% of the full population's, and the model's own final choice among them is 0.9 to 2.0 true-cost standard deviations worse than the best candidate already sitting in its own elite set. The model can tell that some starting situations are harder than others. It cannot tell, among the plans it has already shortlisted for a given situation, which one is actually better, which is exactly the distinction the search procedure's refit step is built to exploit.

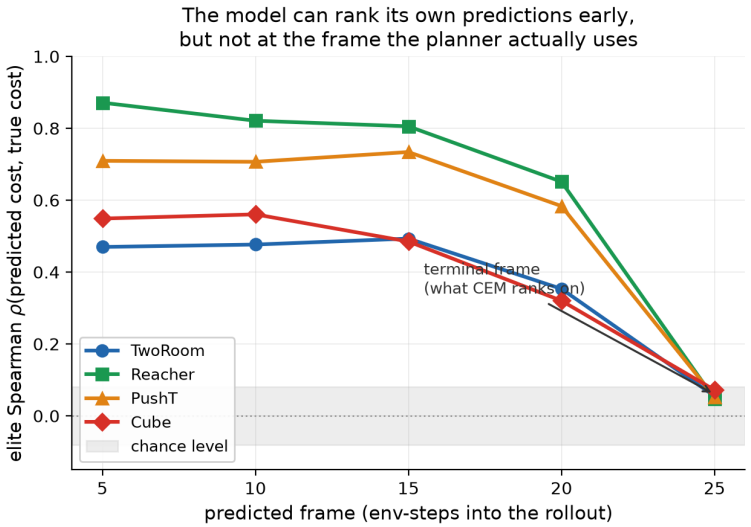


Figure 1: Elite-set Spearman correlation between predicted and true cost, as a function of which predicted frame is queried, on all four environments ($n = 50$). Every environment starts well above chance and ends at chance, exactly at the terminal frame CEM optimizes.

4.2 Resolution collapses toward the terminal frame on every environment

The elite-set failure in Section 4.1 is measured at the terminal frame only. We now ask how ρ_{elite} behaves across every predicted frame in the rollout, using the resolution curve defined in Section 3. Figure 1 shows the result on all four environments. Early in the rollout, the model orders its own elite candidates well, with ρ_{elite} between 0.47 and 0.87 at the first predicted frame. This ability decays steadily and lands at chance, $\rho_{\text{elite}} \in [0.05, 0.07]$, exactly at the terminal frame, on every single environment we test (full per-frame table in Appendix C). The curve is not flat and it is not noise. It is a specific, monotone decay that terminates precisely where the planner's own objective is defined.

4.3 The collapse tracks the search target, not physical time

A resolution curve that decays with rollout length is consistent with two different mechanisms. Under a *physical-time* account, ranking ability degrades with the number of environment steps predicted into the future, because one-step errors compound regardless of what the search procedure does with the prediction. Under a *search-target* account, degradation is instead a property of which frame the search procedure currently refits its elite set against, and frames short of that target keep whatever resolution the model has. These two accounts make different predictions under an intervention that changes the planning horizon H without touching the model or the checkpoint.

We stretch H from 5 to 8 latent-prediction steps and repeat the resolution-curve measurement of Section 4.2 at every new horizon, on TwoRoom and Reacher. The physical-time account predicts the collapse point stays fixed at roughly 25 environment steps regardless of H. The search-target account predicts it moves out to the new terminal frame, roughly 40 environment steps at $H = 8$. Figure 2a shows the result. At 25 environment steps, ρ_{elite} is 0.055 on TwoRoom and 0.047 on Reacher under $H = 5$, where that frame is terminal, matching Section 4.2. Under $H = 8$, the very same 25-step frame is no longer terminal, and its ρ_{elite} recovers to 0.440 on TwoRoom and 0.716 on Reacher, close to the values the model shows at short horizons elsewhere. The crossing point moves out to roughly the new terminal frame at every horizon.

A full dose-response test at $H \in \{5, 6, 7, 8\}$ confirms this sharply (Figure 2b): terminal-frame ρ_{elite} stays inside $[-0.01, 0.09]$ at every horizon on both environments, even as the underlying environment-step count for that frame moves from 25 to 40. This flat dose-response curve, together with the crossing result above, is the strongest evidence in this paper that resolution collapse

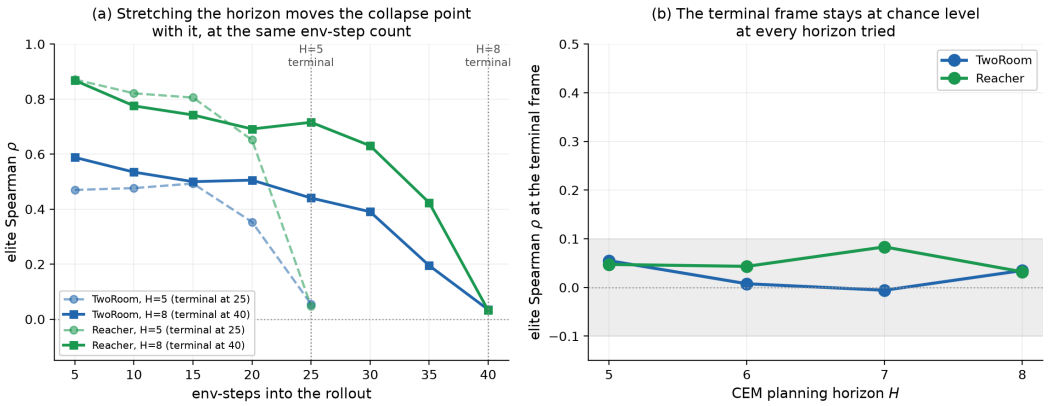


Figure 2: Causal evidence that resolution collapse tracks CEM's optimization target. **(a)** Stretching the horizon from $H{=}5$ to $H{=}8$ moves the collapse point outward by the same amount, at matched environment-step counts. **(b)** The terminal-frame correlation stays pinned near chance at every horizon tried, even though the environment-step count it corresponds to moves from 25 to 40.

is not a property of predicting a fixed number of steps ahead, but of whichever frame the search procedure currently treats as the finish line.

This also explains why a plain L2 penalty on the action sequence, with no model prediction at all, matches or beats every model-derived cost we tried on this codebase (Appendix E): a predictor with no usable elite-set resolution cannot express a preference between candidates whose model cost differs only slightly, and degenerates into noise no matter how it is constructed. The only way to make a model-derived cost carry information inside the elite set is to stop asking it about the one frame where its resolution is guaranteed to be gone.

5 Resolution-Matched Planning

Section 4 shows that the terminal frame is exactly the wrong place to ask the model for a ranking. This suggests a direct correction: query the predictor at a frame it can still resolve, rather than at the frame the task nominally cares about, while keeping the certified L2 baseline as a fixed, weak prior over the whole search. We define $J_{\text{frame}(t)}(c) = \|\hat{z}_t(c) - g\|^2$ for a chosen frame $t \leq H$, paired with the same certified action prior as every other arm in this paper, and ask three questions in turn: does this help, is the gain real, and can the right t be chosen without a success-rate sweep.

5.1 Success rate follows an inverted U over which frame is asked

We sweep every frame index $t \in \{1, \ldots, 5\}$ as the sole main cost against $t{=}5$, the terminal frame, as the baseline (Figure 3, full table in Appendix D). Two patterns hold across all four environments: success rate is an inverted U over the frame index, peaking mid-rollout rather than at either end, and asking too early is actively worse than the baseline, costing 16.5 points on TwoRoom and 22.5 points on PushT at frame 1. The gain lives in a specific window, not simply in querying the model sooner.

Ranking on the window this analysis identifies raises success rate by 4.5 points on TwoRoom (63.5% → 68.0%), 5.0 points on Reacher (91.0% → 96.0%), 3.0 points on PushT (69.5% → 72.5%), and 4.0 points on Cube (64.5% → 68.5%). An additive variant, *MultiHorizon*, which combines a mid-horizon term with the terminal-frame term instead of replacing it, gives a comparable or larger gain on the two environments where we find a real mid-horizon effect (Section 5.2), and is the variant we use for the additive random-cost control below.

Figure 4 shows this gain inside a single real episode on TwoRoom, same model, same start, same goal, same solver. The terminal-frame baseline circles near the dividing wall and times out without finding the gap; the resolution-matched fix, changing only which predicted frame the cost reads, threads the gap and reaches the goal five steps early.

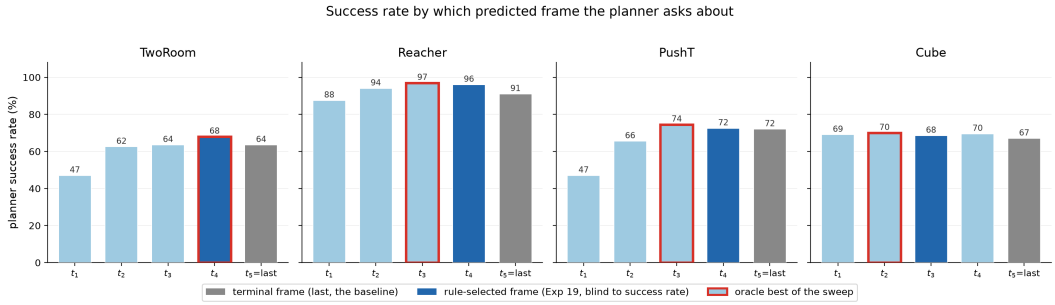


Figure 3: Planner success rate as a function of which predicted frame is used as the main cost, on all four environments ($n = 200$). Grey bars are the terminal-frame baseline, blue bars are the frame our success-rate-blind rule (Section 5.3) selects, and the red outline marks the oracle best of the sweep. The blind rule lands on or within one point of the oracle everywhere.

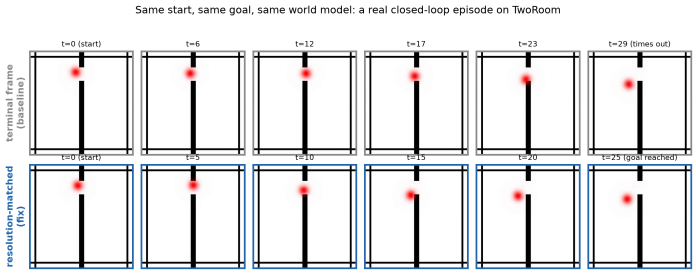


Figure 4: A real closed-loop episode on TWOROOM, same start and goal, same world model and solver, differing only in which predicted frame the planning cost reads. **Top**: the terminal-frame baseline circles near the wall and times out. **Bottom**: the resolution-matched fix finds the gap and reaches the goal.

5.2 THE GAIN IS NOT A MAGNITUDE ARTIFACT

Any additional cost term handed to a refitting search procedure can appear to help for a reason that has nothing to do with the information it carries: a term with the right magnitude can break ties inside the elite set more decisively than the terminal-frame cost alone, regardless of its content. We rule this out with two structurally different random-cost controls, each replacing the real cost with a synthetic cost of identical across-candidate spread but no relationship to the goal.

The *additive* control tests *MultiHorizon* ($J_{\text{last}} + J_{\text{mid}}$) against $J_{\text{last}} + J_{\text{rand}}$, where J_{rand} is seeded noise matched in spread to J_{mid}. On the two environments with a real mid-horizon effect, the real term beats matched noise by 12.5 points on TWOROOM and 11.5 points on REACHER, both at $p < 0.001$, while matched noise alone does not beat the terminal-frame baseline on either environment (Figure 5a, full table in Appendix D). The *replacement* control is sharper, swapping the entire main cost for matched noise outright, using the rule-selected frame from Section 5.3. Every environment collapses when the real signal is removed, by 55.0 points on TWOROOM, 93.5 points on REACHER, 72.0 points on PUSHT, and 23.0 points on CUBE, all at $p < 10^{-11}$ (Figure 5b), including PUSHT and CUBE, where the resolution-matched frame was not itself a significant gain over the terminal baseline. That contrast is informative: adding real signal where the terminal frame is already usable buys little, but removing all distance signal is catastrophic everywhere, confirming the control discriminates rather than uniformly rejecting every synthetic cost.

5.3 A RULE THAT NEVER LOOKS AT SUCCESS RATE

The frame sweep in Section 5.1 chooses its query frame by trying every option and reading off the best success rate, which is exactly the kind of selection a reviewer should distrust: it offers no evidence the same choice would transfer to a new environment or checkpoint that was not swept. We instead select the query frame from the resolution curve itself, a signal that never touches planning

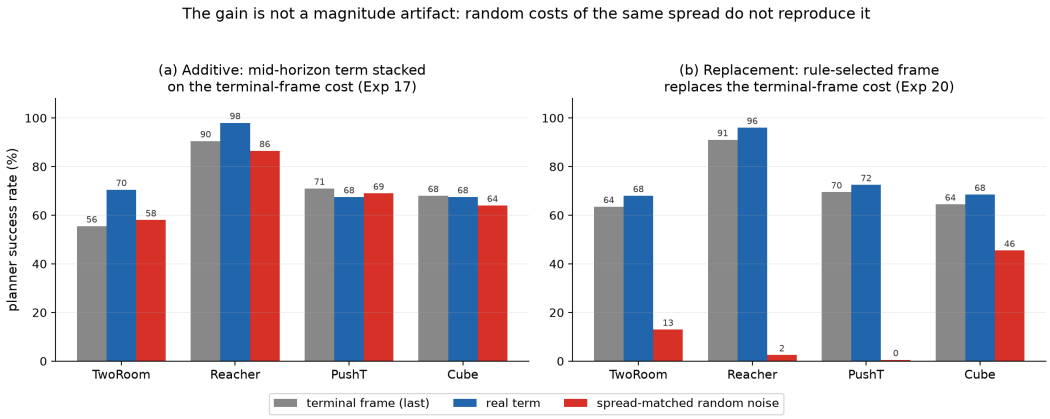


Figure 5: Two independent random-cost controls. **(a)** Adding a real mid-horizon term to the terminal-frame cost beats a spread-matched random term by 11 to 13 points at $p < 0.001$, on the two environments where the term has a real effect. **(b)** Replacing the terminal-frame cost with the rule-selected frame and then swapping in spread-matched noise collapses success rate by 23 to 94 points on every environment, at $p < 10^{-11}$.

Table 2: The success-rate-blind rule (Section 5.3) against the terminal-frame baseline and the exhaustive-sweep oracle ($n = 200$, McNemar exact test).

Env	t^*	baseline	rule SR	oracle SR	rule vs. baseline	rule vs. oracle
TwoRoom	4	63.5%	68.0%	68.0%	+4.5pp, p=0.22	+0.0pp, p=1.00
Reacher	4	91.0%	96.0%	97.0%	+5.0pp, $\mathbf{p=0.013}$	−1.0pp, p=0.75
PushT	4	69.5%	72.5%	74.5%	+3.0pp, p=0.36	−2.0pp, p=0.60
Cube	3	64.5%	68.5%	70.0%	+4.0pp, p=0.12	−1.5pp, p=0.65

outcome. Given the per-frame elite correlation $\rho(t)$ measured as in Section 4.2, we choose

$$t^* = \max\{t : \rho(t) \geq \text{frac} \cdot \rho(1)\},$$

the latest frame that still retains at least a fixed fraction of the model's shortest-horizon resolution. A single value, $\text{frac} = 0.7$, chosen from the middle of a stable plateau ($\text{frac} \in [0.5, 0.7]$ gives the same frame choice on three of four environments, full sensitivity table in Appendix D), selects frame 4 on TwoRoom, Reacher, and PushT, and frame 3 on Cube, using the same threshold everywhere.

Table 2 compares this success-rate-blind choice against the terminal-frame baseline and against the oracle frame identified by the exhaustive sweep. Two results hold on every environment at once. The blind rule beats the terminal-frame baseline by 3.0 to 5.0 points everywhere, reaching significance on Reacher and never landing on the wrong side of zero, including on PushT and Cube, where the sweep's own oracle gain was not itself significant. And the blind rule is statistically indistinguishable from the oracle everywhere ($p \geq 0.6$ in every comparison), landing exactly on the oracle on TwoRoom and within 2.0 points of it elsewhere. A rule that never consults success rate recovers the sweep's gain anyway, using only a measurement of the model's own resolution.

Taken together, resolution-matched planning follows the shape the diagnosis in Section 4 predicts, survives two independent controls against being a magnitude artifact, and needs no success-rate sweep to find, since the model's own resolution curve already contains the information needed to choose where to look.

6 Limitations

Our experiments use one pretrained world model and one search procedure, CEM, across four simulated environments; the elite-width ablation in Appendix E gives a structural reason to expect the same failure in MPPI and iCEM, though we have not run the diagnosis directly on those solvers. The

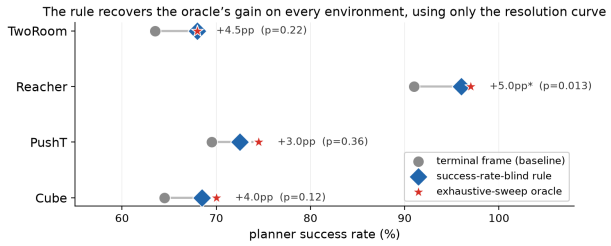


Figure 6: Table 2 at a glance. The success-rate-blind rule (diamond) sits close to the exhaustive-sweep oracle (star) and away from the terminal-frame baseline (circle) on every environment, using only the resolution curve to choose where to look.

gain is clear and significant on TWOROOM and REACHER, and smaller on PUSHT and CUBE, and the rule's threshold, $\mathrm{frac} = 0.7$, is chosen from a stable plateau rather than derived parameter-free.

7 CONCLUSION

We identified resolution collapse, a specific and previously undocumented failure in which a latent world model loses its ability to rank the candidates a planner has already narrowed down to, exactly at the frame the planner treats as its objective, and showed with horizon manipulation that it tracks the search procedure's optimization target rather than physical prediction error. This diagnosis leads directly to a fix that requires no retraining and no architecture change, survives two independent random-cost controls, and can be applied with a rule that never consults success rate. Because the underlying mechanism is shared by the wider family of planners built on latent world models, and because the fix is a one-line change to a single existing cost function, we expect the diagnostic tools and the fix introduced here to be directly useful, at no extra inference cost, beyond the single model and solver studied in this paper.

8 AI USE STATEMENT

An LLM was used only to polish the writing (grammar and clarity) of this paper and to help run some experiments. It did not contribute to the research ideas, methodology, or analysis.

9 ETHICS STATEMENT

This work studies a representational property of self-supervised world models trained on simulated robotics benchmarks (TWOROOM, REACHER, PUSHT, and CUBE). No human subjects data is used. We do not foresee direct harms from this research; the environments and benchmarks used are standard in the model-based reinforcement learning literature.

10 REPRODUCIBILITY STATEMENT

Full model, planner, and evaluation-protocol details for every environment are given in Appendix A. The statistical methodology behind every reported test, the elite Spearman correlation and the exact McNemar test, is specified in Appendix B. Complete per-frame, per-environment numeric results underlying every figure and table in the main text are given in Appendices C and D, and a discussion of which environments reproduce bit-exactly across process invocations, together with the resulting tolerance on isolated success-rate numbers, is given in Appendix G. Code implementing the resolution-curve diagnostic, the horizon-manipulation causal test, the spread-matched random-cost controls, and every planning and evaluation script used to produce the numbers in this paper is included in the supplementary material.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.

Mahmoud Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *PMLR*, pp. 2555–2565, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

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

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *Proceedings of the 4th Conference on Robot Learning*, volume 155 of *PMLR*, 2020.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588: 604–609, 2020.

Charles Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15(1):72–101, 1904.

Erik Talvitie. Self-correcting models for model-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. In *International Conference on Learning Representations*, 2020.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic MPC for model-based reinforcement learning. In *IEEE International Conference on Robotics and Automation*, pp. 1714–1721, 2017.

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

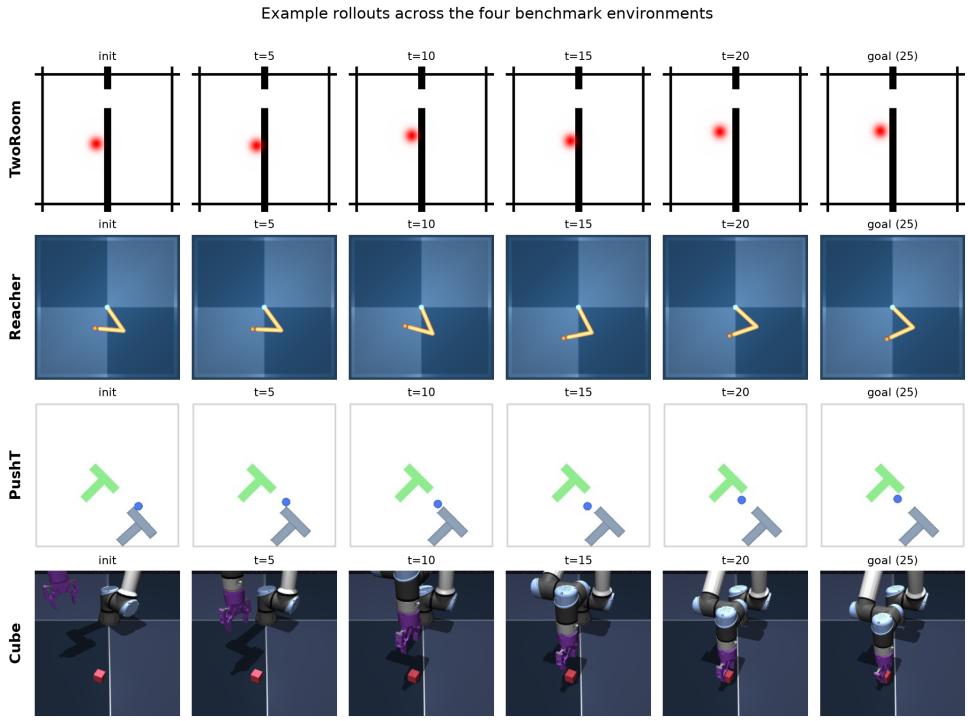


Figure 7: One representative rollout per environment, at every 5-step planning checkpoint from the initial frame to the goal frame 25 environment steps later. Ground-truth simulator frames.

A Environments, Data, and Model Details

Model. We use the publicly released LeWM checkpoint (Maes et al., 2026) for every environment, with no fine-tuning. The encoder and predictor operate on an embedding of dimension 192 from 224×224 pixel observations. The predictor is autoregressive over an arbitrary number of latent-prediction steps, so extending the planning horizon (Section 4.3) requires no architecture change, only a longer rollout at inference time.

Planner. CEM samples $N = 300$ candidate action sequences per control step from a diagonal Gaussian with initial variance scale 1.0, keeps the $k = 30$ lowest-cost candidates as the elite set, refits the sampling distribution to the elite mean and variance, and repeats for 30 iterations before executing the first action block of the lowest-cost sequence found in the final iteration. Each latent-prediction step corresponds to an action block of 5 raw environment steps. The default horizon is $H = 5$ latent-prediction steps, so the terminal frame corresponds to 25 environment steps of lookahead, matching the dataset's goal offset of 25 steps used throughout the paper except in the horizon-manipulation experiments of Section 4.3.

Environments. TwoRoom is a top-down point-mass navigation task through a gap in a wall dividing two rooms, drawn from a real expert dataset of 10,000 episodes. Reacher is a two-link planar arm reaching a target end-effector position, with deterministic dynamics that make it the one environment in our suite that reproduces bit-exactly across repeated process invocations of the same episode set. PushT requires pushing a T-shaped block to a target pose with a circular pusher, drawn from an expert dataset of 18,685 episodes, and exhibits a measurement noise floor of roughly 5 points of success rate between otherwise identical repeated runs, because its physics are not bit-reproducible across process invocations. Cube is a simulated Franka-style manipulator relocating a small cube and shows a similar noise floor. Figure 7 shows one representative initial-frame-to-goal-frame rollout per environment, rendered directly from the simulator (these are ground-truth frames, not model reconstructions, since LeWM has no pixel decoder). Per-environment storyboards at every planning checkpoint are given in Figures 8–11.

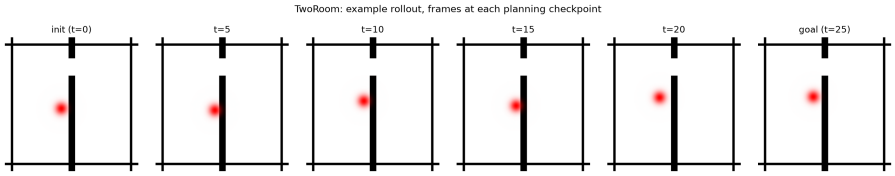


Figure 8: TWOROOM: example rollout, one frame per planning checkpoint.

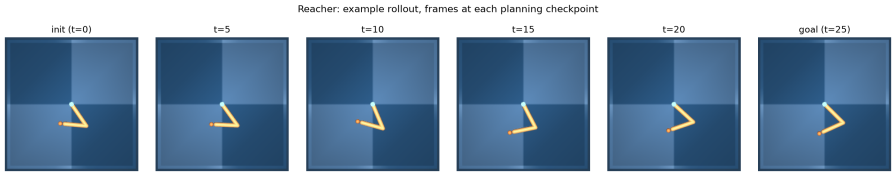


Figure 9: REACHER: example rollout, one frame per planning checkpoint.

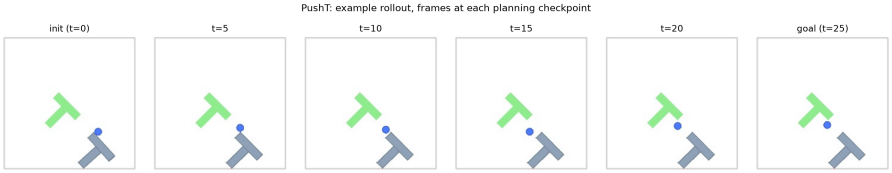


Figure 10: PUSHT: example rollout, one frame per planning checkpoint.

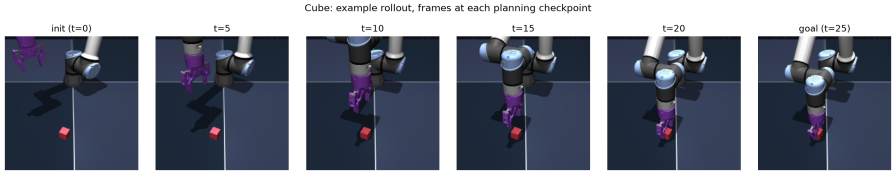


Figure 11: CUBE: example rollout, one frame per planning checkpoint.

Evaluation protocol. Every experiment draws (init, goal) pairs from a held-out split of each environment's expert dataset, using seed 3072 throughout, so the same 200 pairs are reused across every planner comparison in the paper unless otherwise noted, and the same 50 pairs are reused across every resolution measurement. This lets us run every paired comparison (real vs. random-cost control, rule vs. oracle, terminal vs. resolution-matched) within a single process invocation on identical episodes, which is what makes the exact McNemar test in Appendix B valid even though CEM's own candidate sampling is not seeded across separate process invocations (we discuss this reproducibility caveat further in Appendix G). Success is the environment's own binary goal-reaching criterion, unchanged from its standard evaluation harness.

B STATISTICAL METHODOLOGY

Elite Spearman correlation. For a set of candidates scored by both a model-predicted cost and a re-executed true cost, we report the Spearman rank correlation (Spearman, 1904) restricted to the 30 candidates CEM's final iteration keeps as its elite set. This is the only quantity CEM's refit step consumes: it needs the relative order of elite candidates to be correct, not the absolute value or even the sign of the cost gap between them.

Paired significance testing. All success-rate comparisons in this paper are on paired outcomes, since every arm being compared is evaluated on the identical set of (init, goal) pairs within the same process invocation. We use the exact McNemar test (McNemar, 1947), computed as a two-sided exact binomial test on the discordant pairs (episodes where the two arms disagree), implemented via `scipy.stats.binomtest` rather than an asymptotic chi-squared approximation, which matters when discordant counts are small.

Matched random-cost controls. A spread-matched random cost J_{rand} is seeded noise, independent of the goal and of the candidate's content, whose per-call standard deviation across the candidate population is set equal to the measured across-candidate spread of the real term it replaces or is added alongside, on the same warm-up pairs used to calibrate the certified L2 weight. This ensures J_{rand} can reorder the elite set exactly as forcefully as the real term can, while carrying no information about the goal, so any success-rate difference between the real term and J_{rand} cannot be explained by a difference in magnitude.

C Full Resolution Collapse Results

Elite-set ranking fidelity (Section 4.1). Table 3 extends Table 1 with the pairwise ranking accuracy inside the elite set (fraction of elite candidate pairs correctly ordered, against a 50% chance baseline) and the noise ceiling on the achievable correlation, obtained by re-executing a subset of candidates twice and measuring how much the resulting latent deviates. TwoRoom is exact (deviation 0.0 on a cost scale of 138); Reacher and Cube deviate by 0.003% and 0.02%; PushT deviates by 8.5%. Converting these into the largest correlation any predictor could attain gives a ceiling of 0.99 to 1.00 on every environment, ruling out measurement noise as an explanation for an elite correlation of 0.05.

Table 3: Elite-set ranking fidelity, extended ($n = 50$ episodes, 60 re-executed candidates per episode: all 30 elite plus 30 spanning the rest of the population).

Env	elite pairwise accuracy	most confident bin	noise ceiling	regret (true-cost sd)
TwoRoom	51.3%	53.6%	1.00	2.02
PushT	50.8%	56.3%	0.99	0.97
Reacher	50.7%	54.3%	1.00	0.90
Cube	50.4%	55.1%	1.00	1.37

Hierarchical signal decomposition. Planning is a two-level problem: candidates are nested inside situations. Splitting the model-versus-truth agreement into these levels (Table 4) shows the same collapse from a different angle. The model agrees with the truth about which *situations* are hard, agrees more weakly about which *plans* are better, and does not agree at all about which of the *good* plans is better, the only distinction a planner actually needs once it has already ruled out the bad ones.

Table 4: Correlation between model and true terminal cost at three levels of granularity.

Env	which situation is hard	which plan is better	which good plan is better
TwoRoom	0.670	0.406	0.086
PushT	0.830	0.467	-0.007
Reacher	0.175	0.159	-0.014
Cube	0.662	0.108	0.093

Resolution curve, full table (Section 4.2). Table 5 gives the per-frame elite Spearman correlation underlying Figure 1.

Table 5: Elite Spearman correlation by predicted frame ($n = 50$). t_1–t_4 are intermediate frames at 5-step intervals; t_5 is the terminal frame at 25 environment steps.

Env	t_1 (5 steps)	t_2 (10)	t_3 (15)	t_4 (20)	t_5 (25, terminal)
TWOROOM	0.470	0.476	0.493	0.353	0.055
REACHER	0.871	0.821	0.805	0.651	0.047
PUSHT	0.709	0.707	0.734	0.583	0.052
CUBE	0.549	0.560	0.485	0.320	0.072

Causal horizon manipulation, full tables (Section 4.3). Table 6 gives the matched environment-step comparison between $H = 5$ and $H = 8$ underlying Figure 2a. Table 7 gives the full dose-response sweep over $H \in \{5, 6, 7, 8\}$ underlying Figure 2b.

Table 6: Elite Spearman correlation at matched environment-step counts, $H = 5$ vs. $H = 8$ ($n = 50$).

env-steps	TWOROOM, H=5	TWOROOM, H=8	REACHER, H=5	REACHER, H=8
5	0.470	0.588	0.871	0.868
10	0.476	0.535	0.821	0.775
15	0.493	0.500	0.805	0.742
20	0.353	0.505	0.651	0.691
25	**0.055**	0.440	**0.047**	0.716
30	–	0.391	–	0.630
35	–	0.195	–	0.424
40	–	**0.035**	–	**0.032**

Table 7: Terminal-frame elite Spearman correlation as a function of the planning horizon H ($n = 50$). The correlation stays inside $[-0.01, 0.09]$ at every horizon on both environments, even as the environment-step count of the terminal frame moves from 25 to 40.

Env	H=5	H=6	H=7	H=8
TWOROOM	0.055	0.008	−0.006	0.035
REACHER	0.047	0.043	0.083	0.032

D FULL RESOLUTION-MATCHED PLANNING RESULTS

Frame sweep, full table (Section 5.1). Table 8 gives the full frame sweep underlying Figure 3, including the additive *MultiHorizon* arm. Asterisks mark McNemar $p < 0.05$ against the terminal-frame (t_5) baseline.

Table 8: Planner success rate by main cost, all arms paired with the certified L2 prior at $r = 0.1$ ($n = 200$).

Env	t_1	t_2	t_3	t_4	t_5 (baseline)	MultiHorizon
TWOROOM	47.0%***	62.5%	63.5%	**68.0%**	63.5%	70.5%
REACHER	87.5%	94.0%	**97.0%***	96.0%*	91.0%	98.0%*
PUSHT	47.0%***	65.5%	**74.5%**	72.5%	69.5%	71.5%
CUBE	69.0%	**70.0%**	68.5%	69.5%	64.5%	67.0%

Additive random-cost control, full table (Section 5.2). Table 9 gives the three-arm comparison underlying Figure 5a. All three arms share one certified-L2 calibration per environment, so `multi_h_l2` versus `randmid_l2` is a controlled swap of only the added term's content.

Table 9: Additive control: *MultiHorizon* vs. spread-matched random noise added to the terminal-frame cost ($n = 200$, exact McNemar).

Env	terminal (last)	MultiHorizon	random (matched)	MH vs. last	MH vs. random
TWOROOM	55.5%	70.5%	58.0%	+15.0pp, p<0.001	+12.5pp, p<0.001
REACHER	90.5%	98.0%	86.5%	+7.5pp, p=0.001	+11.5pp, p<0.001
PUSHT	71.0%	67.5%	69.0%	−3.5pp, p=0.35	−1.5pp, p=0.75
CUBE	68.0%	67.5%	64.0%	−0.5pp, p=1.00	+3.5pp, p=0.17

Frame-selection rule, sensitivity and comparison (Section 5.3). The rule selects $t^* = \max\{t : \rho(t) \geq \text{frac} \cdot \rho(1)\}$. Table 10 shows the frame chosen at seven values of frac swept from 0.5 to 0.9: the choice is stable at $t^* = 4$ on three of four environments across the entire $[0.5, 0.7]$ range, and we report $\text{frac} = 0.7$, the middle of this plateau, as the primary rule in the main text.

Table 10: Frame chosen by the rule across a sweep of frac.

Env	0.50	0.60	0.65	0.70	0.75	0.80	0.90
TWOROOM	4	4	4	4	4	3	3
REACHER	4	4	4	4	3	3	3
PUSHT	4	4	4	4	4	4	3
CUBE	4	3	3	3	3	3	2

We also record a rule we tried and rejected, for transparency. An earlier version selected $t^* = \max\{t : \rho(t) \geq \tau\}$ for a fixed absolute threshold $\tau = 0.5$, falling back to t=1 if no frame cleared the bar. This failed on TWOROOM, whose resolution curve never exceeds 0.493: the rule fell through to its t=1 fallback, which Section 5.1 already shows is the single worst arm on that environment (−16.5pp versus the terminal baseline). An absolute threshold treats a correlation coefficient as comparable across environments with different baseline predictability, and treats the shortest horizon as a safe fallback. Both assumptions are wrong, which is why the relative-threshold rule reported in Section 5.3 normalizes by each environment's own $\rho(1)$ and never degenerates to that fallback in practice.

Replacement random-cost control, full table (Section 5.2). Table 11 gives the comparison underlying Figure 5b, reusing the exact spread and certified-L2 calibration from the rule-selected frame in Section 5.3 with no recalibration, so this is a drop-in content swap rather than a re-tuned arm.

Table 11: Replacement control: rule-selected frame vs. spread-matched random noise as the entire main cost ($n = 200$, exact McNemar).

Env	t^*	rule-selected frame	random (matched spread)	difference
TWOROOM	4	68.0%	13.0%	+55.0pp, p=2e−33
REACHER	4	96.0%	2.5%	+93.5pp, p=1e−56
PUSHT	4	72.5%	0.5%	+72.0pp, p=9e−44
CUBE	3	68.5%	45.5%	+23.0pp, p=2e−12

E MOTIVATING EVIDENCE: WHY THE SEARCH PROCEDURE NEEDS THIS FIX

This appendix collects three earlier analyses on the same codebase that originally motivated the diagnosis in Section 4 and support two claims made in the main text: that near-tied elite candidates are the norm rather than the exception, and that the failure is specific to a search procedure that refits its sampling distribution to a scored elite set.

A plain action-magnitude prior beats every model-derived cost we tried. Before diagnosing why, we first noticed that a term which never queries the model, an L2 penalty on the action sequence, was the strongest single arm on every environment in an earlier sweep that also included two other model-derived candidate costs (Table 12). A colored-noise action-sampling prior in the style of iCEM (Pinneri et al., 2020) was, by contrast, the worst arm tested, which we trace to reduced candidate diversity inside the elite set rather than to any property of the temporal correlation structure itself, an implementation detail we discuss further alongside the numbers below. REACHER is the cleanest environment for this comparison, since it is the one that reproduces bit-exactly across process invocations and the one where the spread-matched random control shows no effect at all (-1.0pp), meaning the confound this paper studies is absent there and any remaining gain is attributable to real information content.

Table 12: Success-rate gain over a same-process baseline, best setting per arm, 200 episodes per arm (*: McNemar $p < 0.05$).

Arm	TWOROOM	PUSHT	REACHER	CUBE
action L2 penalty (model-free)	$+14.5^*$	$+3.5$	$+16.5^*$	$+10.5^*$
action smoothness (model-free)	$+8.0$	$+0.0$	$+9.5^*$	$+3.5$
colored-noise proposal (model-free)	-5.5	-14.5^*	-22.5^*	-5.5
spread-matched random (control)	$+8.5^*$	$+5.5$	-1.0	$+5.0$
a model-derived risk signal	$+7.5^*$	-1.0	$+0.0$	$+5.5$
novelty-based model-derived signal	$+10.0^*$	$+5.0$	-3.5	$+10.5^*$

The confound is specific to a refitting search procedure. Sweeping the CEM elite size $k \in \{5, 15, 30, 60, 150\}$ at a fixed population of $N = 300$ produces an inverted U rather than a monotone trend (Table 13): a five-candidate elite already over-commits the proposal distribution before an added cost has room to act, and a 150-candidate elite approaches a non-refitting variant of the same planner, where an added cost no longer has a measurable effect on REACHER and CUBE. The confound lives in the window $k \in \{15, 30\}$, which contains the paper's default and every result in this paper.

Table 13: Success-rate gain from an added cost over the terminal-frame baseline, by elite width k at fixed $N = 300$ (*: $p < 0.05$).

Env	k=5	k=15	k=30	k=60	k=150
TWOROOM	$+0.5$	$+6.0$	$+4.5$	$+2.5$	$+1.5$
PUSHT	-6.5	-2.0	$+4.5$	-1.5	$+2.0$
REACHER	$+4.5$	$+9.5$	$+1.5$	$+0.5$	-1.5
CUBE	$+1.5$	$+10.0^*$	$+6.5^*$	$+5.0$	-2.0

F A COMPANION NEGATIVE CONTROL

To calibrate how discriminating our random-cost control methodology actually is, it is useful to report a case, studied in a companion investigation on the same codebase, where a different model-derived planning cost did *not* survive it. A regularizer used during training to keep the latent space from collapsing can also be evaluated, at no extra training cost, on the latents the model predicts during planning, giving a candidate risk signal. On its own, this signal reaches an AUROC of up to 0.77 for predicting episode failure and, read before the agent executes a single action, flags roughly a fifth of CUBE episodes that go on to fail. Once rebuilt with the same spread-matched random-cost control used in Section 5.2 and given the same weight sweep as the real term, this signal becomes indistinguishable from noise as a planning cost on all four environments, reversing an earlier conclusion drawn before the control was corrected. This is the outcome our own methodology predicts should be possible: the control is not a formality that every candidate cost passes automatically, and the fact that it rejects one plausible model-derived signal while the resolution-matched cost in Section 5.2 passes the same test is itself evidence that the test is doing real work.

G REPRODUCIBILITY

TWOROOM and REACHER reproduce bit-exactly across separate process invocations of the identical episode set and configuration. PUSHT and CUBE do not: repeating the identical baseline protocol on the identical episode set changes the outcome of 17 to 22.5% of episodes and moves success rate by 4.5 to 5.0 points between two otherwise identical runs, a floor we attribute to CEM's own candidate sampling not being seeded consistently across process boundaries in the underlying planning library. This does not threaten any comparison in this paper, since every paired comparison we report (real vs. random cost, rule vs. oracle, resolution-matched vs. terminal) is computed within a single process invocation on identical episodes and identical random draws for every arm, which is what the exact McNemar test in Appendix B requires. It does mean that an isolated success-rate number for a single arm, quoted without its same-process baseline, should be read with a few points of tolerance on PUSHT and CUBE. All experiments were run on a single machine with one NVIDIA A100 80GB GPU. A full resolution-curve measurement ($n = 50$, four planning frames re-executed) takes under ten minutes per environment; a full 200-episode planner comparison across three arms takes fifteen to forty minutes per environment depending on the environment's own simulation cost.