How World Models Misjudge Actions: A Case Study of LeWM

Anonymous authors
Paper under double-blind review

Abstract

World models guide planning by comparing predicted action consequences, but the distinctions they make need not match those that matter for execution. We study this mismatch in LeWM across Reacher, TwoRoom, Cube, and PushT through matched candidate executions and interventions on physical states, commands, scoring, and output formation. Beyond an average ranking advantage of realized over predicted representations at matched endpoints in all four tasks, these interventions reveal mismatches in both directions. Cube action encoding nearly ignores differences that change object motion, whereas TwoRoom predictions respond to command differences discarded during execution. In Reacher, single-frame inputs conceal velocities that change future motion. In PushT pools admitting valid orientation-edited references for every candidate, scoring interactions involving the edit and prediction error can reinforce or offset incorrect action preferences. Averaging can carry nearly ignored action differences into execution or break equivalence between individually interchangeable commands, changing success without changing selected members. In two additional TwoRoom weight variants, removing execution-irrelevant command differences improves fixed-pool rankings, while separate closed-loop tests yield no observed gains in success. These findings motivate testing both sensitivity to consequential differences and respect for execution equivalence, together with how scoring and output formation turn model distinctions into decisions.

1 Introduction

Visual world models guide planning by predicting action consequences and comparing them with a goal (Zhou et al., 2025; Terver et al., 2026; Maes et al., 2026). We ask whether they preserve the physical distinctions that determine those consequences, and how scoring and output formation use these distinctions. An action change barely registered by the model may alter object motion; a command change discarded during execution may still alter prediction. Distinguishing outcomes and judging which action to execute are separate requirements.

Predictive objectives can diverge from control performance (Lambert et al., 2020), and latent-planning studies diagnose scoring metrics, execution interfaces, and physical intervention responses (Li et al., 2026; Vesessook & Yang, 2026; Vakalis, 2026). We trace specific physical distinctions through these computations: which are absent from the input, nearly removed by encoding, or retained despite execution equivalence, and how their treatment affects preferences and outputs.

We study fixed LeWM models on Reacher, TwoRoom, Cube, and PushT, using actual execution to compare physical and model responses. Interventions follow three behavioral paths: rescoring and reselecting candidates, replacing commands within fixed elites, and executing a fixed output from different hidden states. The analyses yield three connected findings.

Model distinctions can misalign with physical consequences in both directions. In Cube, action differences nearly ignored by the encoder change object motion. In Reacher, identical single-frame inputs conceal different velocities, and the shared prediction usually lies closer to the zero-velocity outcome. Conversely, TwoRoom predictions respond to command differences discarded during execution.

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.

Scoring can use physical distinctions in ways that misorder actions. Removing execution-irrelevant command differences in TwoRoom improves average prediction and ranking, yet some rankings worsen. In PushT, physically edited references separately test how the realized orientation residual and prediction error align with the remaining goal discrepancy. Removing these directional interactions changes preferences in both directions, identifying contributions that reinforce or offset physical misrankings.

Output formation can change which distinctions matter in execution. In Cube, a common model-insensitive action displacement changes success despite unchanged elite membership and order. In TwoRoom, individually execution-equivalent replacements can change the averaged action and its success. Their gains and losses differ from those obtained through rescoring and reselection.

Across two additional TwoRoom weight variants, removing the same execution-irrelevant command differences improves fixed-pool rankings, while separate closed-loop tests yield no observed success gain. Local judgment improvements therefore need not survive replanning and output formation.

2 Related Work

Representations for planning. Visual planning uses pretrained or jointly learned representations (Terver et al., 2026; Zhou et al., 2025; Maes et al., 2026). Objective mismatch, value-aware learning, and value equivalence relate model quality to downstream use (Lambert et al., 2020; Grimm et al., 2020; Farahmand et al., 2017). We examine physical distinctions from action encoding to execution.

Physical distinctions and action responses. PhyLatent grounds encoded and predicted latents in physical state and separates counterfactual action branches (Zeng et al., 2026); PSG-JEPA examines state changes, including different actions reaching the same configuration (Yan et al., 2026). FlowMo-WM infers motion state from image-action histories (Jiang et al., 2026). Action decoding, inverse dynamics, and physical supervision further examine or shape action-conditioned representations (Zhang et al., 2026; Ivashkov et al., 2026; Boylan & Hokamp, 2026; Zhu et al., 2026). We test consequential changes nearly ignored by encoding and command changes discarded during execution.

Scores and action preferences. TRM studies shared-candidate and closed-loop decisions, SCALE compares readout with metric use, and Decision-Metric Alignment examines ranking across search stages (Li et al., 2026; Hu et al., 2026; Wang et al., 2026). The Objective Is the Bottleneck finds decodable information poorly used by LeWM's goal distance (Singh, 2026). Traj-LeWM evaluates matched endpoints with success–failure AUC (Huang et al., 2026); D-JEPA learns candidate relations from executed outcomes (Liu et al., 2026). WorldGym compares model-based and real policy rankings (Quevedo et al., 2026). Beyond Visual Quality tests whether scores identify successful world-action-model candidates executed from the same state under a common continuation policy (Yuan et al., 2026). We trace LeWM's candidate preferences through native score interactions, elite averaging, and execution.

Planning interfaces and output formation. LEAP scores predicted terminal states and projects optimized actions before rescoring (Pham & Bera, 2026). Hidden Failure Modes examines scoring horizons, executed prefixes, and waypoint interfaces (Vesessook & Yang, 2026); iCEM distinguishes the elite mean from evaluated sequences (Pinneri et al., 2021). We distinguish fixed-pool rescoring from command changes within fixed elites: Cube's encoder-insensitive displacement and TwoRoom's execution-equivalent replacements can alter executed means and success without changing selected members.

Physical intervention diagnostics. The Intervention Gap separates information capture, decodability of real intervention effects, and their propagation through model dynamics (Vakalis, 2026). Physically viable world models argue for preserving distinctions relevant to intervention queries, while decision-centered evaluation emphasizes counterfactual action fidelity and downstream utility (Thorpe et al., 2026; Yu et al., 2026). We test how such distinctions affect native scoring and output formation through cross-term interventions and fixed-member command replacements. Appendix E details protocol, theoretical, and system connections.

3 STUDY DESIGN

We compare model judgments with actual execution and trace controlled interventions through selection, output formation, and behavior (Figure 1).

3.1 WORLD-MODEL PLANNING

We study fixed LeWM checkpoints (Maes et al., 2026) in four tasks. Reacher targets a joint configuration, TwoRoom an agent position, and Cube a three-dimensional block position. PushT targets the agent position, block position, and block orientation. The planner observes images; physical states determine success. Appendix A gives the task criteria and restoration conditions.

On each planning call, the cross-entropy method (CEM) ranks sampled action sequences by predicted squared distance to the goal representation. Model commands can differ from executed commands; TwoRoom clips them before execution.

A query q specifies an initial state, observation o_{q0}, and goal image g_q. We analyze the first call's final pool of $N = 300$ candidates. Each $\mathbf{a}_{qi}$ contains 25 environment actions, processed as five blocks of five. With $W = 25$, image encoder E, and endpoint predictor F_W,

$$\hat{z}_{qi}(W) = F_W(o_{q0}, \mathbf{a}_{qi}), \qquad z_q^g = E(g_q). \tag{1}$$

The native planner averages the $K = 30$ lowest-cost sequences, indexed by $\mathcal{E}_q(c)$:

$$\mathbf{a}_q^{\text{mean}}(c) = \frac{1}{K} \sum_{i \in \mathcal{E}_q(c)} \mathbf{a}_{qi}. \tag{2}$$

This mean can differ from every candidate and requires its own execution to establish its physical outcome.

3.2 MATCHING PREDICTIONS TO PHYSICAL OUTCOMES

The main ranking comparison evaluates every candidate at exactly environment step $W = 25$, the model's prediction endpoint. We execute each fixed candidate from the same initial state for all 25 steps, even if it reaches the goal earlier. With $z_{qi}(W) = E(o_{qi}(W))$, the physical-endpoint and predicted-endpoint costs are

$$c_{qi}^{\text{real}} = \|z_{qi}(W) - z_q^g\|_2^2, \qquad c_{qi}^{\text{pred}} = \|\hat{z}_{qi}(W) - z_q^g\|_2^2. \tag{3}$$

To match the model's step-25 prediction, the main ranking label records whether the candidate's state at step 25 satisfies the success criterion. An earlier success event is not used as an additional label: it counts only if the step-25 state is also successful. This label forms the success–failure pairs in Table 1. Behavioral comparisons separately ask whether an output succeeds at any time within 25 or 50 steps, with native execution stopping at first success. Appendix C and Table 5 report the separate within-window comparison using success-or-timeout endpoints.

3.3 THREE INTERVENTION PATHS

We change one stage at a time to see how a model distinction reaches behavior. Realized outcomes provide the comparison target, and controls test whether an effect comes from the targeted change rather than an arbitrary perturbation (Section 4).

The first path changes candidate scores, selects a new top 30, and averages their original commands. The second keeps the selected candidates and their order fixed but edits their commands before averaging. The third keeps the output command fixed but changes the initial physical state. These paths separate changes in selection, output formation, and execution. Cube's component exchanges and common displacements are separate tests (Sections 4.2 and 5.1).

Paired branches share the same initial state unless the state itself is changed. Appendix C.2 gives output-rule and subsequent-replanning comparisons.

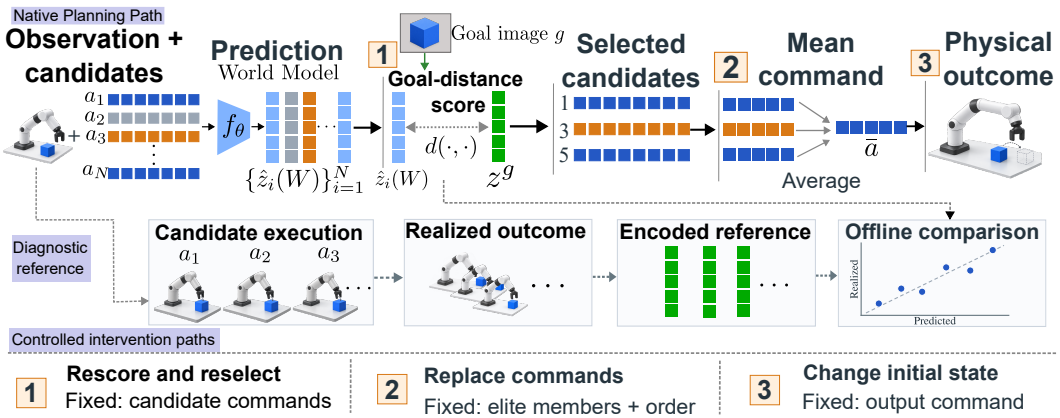


Figure 1: Study design (schematic). The dashed branch executes the same candidates and encodes realized outcomes for offline comparison, outside native planning. Numbers mark intervention sites: (1) rescore and reselect fixed candidate commands; (2) replace commands before averaging with elite membership and order fixed; (3) change initial physical state with output commands fixed. Candidate and mean-output executions are measured separately.

3.4 Samples and Measures

Our main metric asks whether lower costs go to candidates that succeed at step 25 rather than those that fail. For each success–failure pair, the ordering receives 1 if correct, 0.5 if tied, and 0 if reversed. We average first within each query and then equally across queries, separately for the two costs in Equation 3. All-success or all-failure pools contain no such pairs, so their concordance is undefined.

The base comparison uses 150 queries per task. Separate mechanism groups contain 128 additional Cube evaluation episodes, 512 each for Reacher and TwoRoom, and 2,048 for PushT. A further TwoRoom planning study uses 256 episodes; two additional weight variants reuse its native first-call candidate pools for fixed-pool scoring and run separate closed-loop searches. TwoRoom's exact clipping equivalence permits input, aggregation, and replanning tests without retraining the model, while the two additional local weights permit a limited checkpoint comparison. Conditions are paired within episodes, and applicable subsets retain their own denominators. Reacher tests follow two fixed candidates per episode (Appendix D).

Mechanism measures follow their physical question. TwoRoom uses step-25 success and failure. PushT compares whether one endpoint is no worse in agent position, block position, and orientation and better in at least one, even when the binary labels match. Its misranking rate counts strict reversals and keeps ties separate. Behavioral results use execution distance or paired success gains and losses within the stated window.

4 How Action Judgments Are Formed

Lower prediction error need not improve action ranking. We compare predicted and realized scores on fixed candidate pools, then trace which differences the model retains and how prediction errors and physical discrepancies enter the score.

4.1 Action Judgments from Realized and Predicted Outcomes

Before isolating mechanisms, we compare two scores for exactly the same candidates. One uses each candidate's realized representation at step 25; the other uses the model's predicted representation. Candidates, goals, the native squared-distance rule, and step-25 success labels remain fixed (Section 3.4).

Realized-endpoint scores rank step-25 success better in all four tasks, with all four adjusted intervals above zero (Table 1). This diagnoses endpoint scoring on a fixed pool, not policy success; both scores retain the encoder's geometry and neither is a physical success oracle.

Table 1: Concordance with step-25 success for fixed candidates and labels, scored from realized or predicted endpoint representations (Section 3.4). n: mixed-outcome queries among 150 per task. Realized-minus-predicted intervals: 98.75% family-adjusted.

Task	n	Realized endpoint	Predicted endpoint	Realized − predicted	Adjusted interval
Reacher	148	0.9976	0.6596	$+0.3379$	$[0.2906, 0.3889]$
TwoRoom	108	0.7852	0.7197	$+0.0656$	$[0.0086, 0.1240]$
Cube	95	0.9045	0.5424	$+0.3621$	$[0.3044, 0.4117]$
PushT	144	0.9045	0.6451	$+0.2594$	$[0.2040, 0.3183]$

Under the separate success-within-25 definition, scoring observations at success or timeout favors successful actions in Reacher, Cube, and PushT after correction; TwoRoom remains inconclusive (Table 5). Because its mixed-query populations differ, we do not pool it with the step-25 comparison. With success-within-25 labels and representations evaluated at step 25, switching from predicted to realized scores produces more reversed-to-correct than correct-to-reversed pairs in every task. Geometry checks further show that poorer ranking need not mean less representation separation (Appendices C and C.3).

4.2 Which Physical Differences Reach the Model?

World models can treat consequential actions as nearly identical or distinguish commands that execution treats as equivalent. Figure 2 illustrates both directions and hidden state information that changes the future without changing the current observation.

In Cube, each normalized five-action block is a 25-dimensional vector, but the first action layer maps it to only 10 dimensions. It therefore retains a 10-dimensional component and maps the remaining 15-dimensional component to zero. For the rank-10 map W, we write these as the row-space component a_R and null-space component a_N, with $a = a_R + a_N$. For the two lowest-cost candidates, we exchange the ignored component a_N block by block while fixing a_R, then inverse-normalize and execute with native limits. A control exchanges the retained component a_R while fixing a_N. This tests whether action differences invisible to the first model layer still change physical motion.

We repeated the protocol after an initial 32 episodes on 128 nonoverlapping evaluation episodes. Null-space exchanges changed predictions and costs only near numerical precision; retained-component changes produced much larger responses. Maximum block-position separation over shared times and both exchanges averaged 4.47 cm (95% interval: 3.42–5.60 cm), including 12 exact zeros and three machine-precision effects. Separate equal-norm bounded controls also produced early physical changes without activating native limits (Appendix D.1). Encoder near-invariance therefore does not imply physical equivalence.

Reacher exposes information absent from the observation. Because the single frame contains no joint velocities, preserving, reversing, or zeroing them leaves the observation, goal, actions, and prediction identical while real trajectories diverge. In 484 of 512 additional evaluation episodes (94.5%), both fixed candidates' first five-step predictions are closer to the encoded zero-velocity outcome than to the original-velocity outcome. Distinguishing these futures requires visual history or proprioceptive velocity.

TwoRoom encodes unclipped commands, whereas execution clips each component to $[-1, 1]$. Removing excess magnitude preserves execution but changes prediction: episode-averaged squared error decreases in 505 of 512 additional episodes and increases in seven; an equal-magnitude orthogonal control increases it throughout in the 16 initial episodes (Appendix D). Better training-distribution alignment may also contribute to the reduction; the control does not isolate this explanation. The execution equivalence itself is exact.

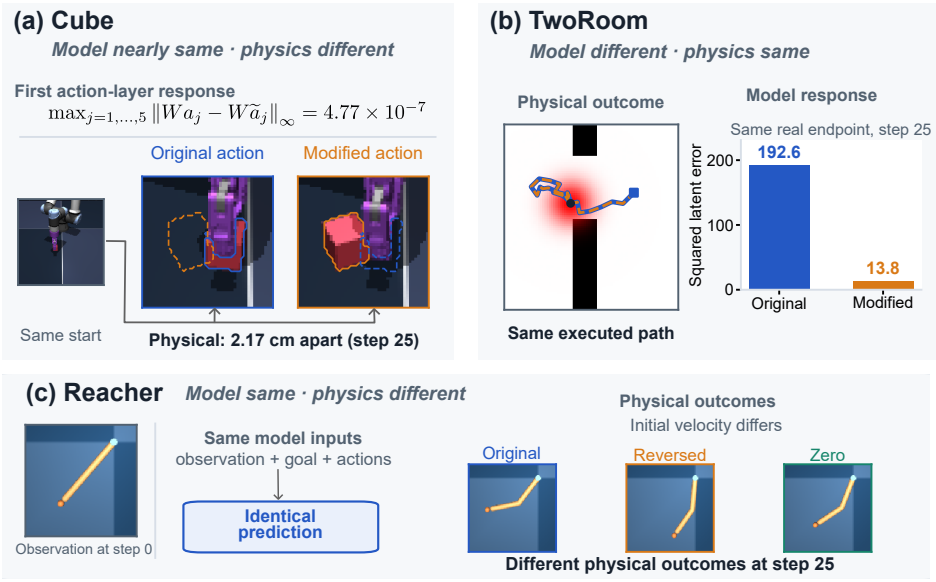


Figure 2: Model responses and physical outcomes. (a) Cube: null-space exchanges preserve retained action components and barely change first-layer responses, but same-start executions leave block positions 2.17 cm apart at step 25; contours mark current and other conditions. (b) TwoRoom: removing command excess preserves the executed path but changes prediction; bars compare squared errors at the same realized endpoint. (c) Reacher: identical observation, goal, and actions give the same prediction but different step-25 outcomes under original, reversed, or zero initial velocity.

4.3 How Prediction Errors Change Preferences

Of 512 additional TwoRoom evaluation episodes, 354 contain both successful and failed candidates at step 25. Removing excess magnitude lowers their mean misranking rate by 3.72 percentage points: 224 improve, 127 worsen, and three are unchanged. The other 158 lack comparable pairs.

Goal-distance scoring depends on error direction as well as magnitude. For one candidate, let z be its realized representation, z^g the goal representation, and $e = \hat{z} - z$ its prediction error. Then

$$\underbrace{\|\hat{z} - z^g\|_2^2}_{\text{predicted cost}} = \underbrace{\|z - z^g\|_2^2}_{\text{realized cost}} + \underbrace{2(z - z^g)^\top e}_{\text{directional error term}} + \underbrace{\|e\|_2^2}_{\text{squared error}} . \tag{4}$$

When the prediction error points along the realized goal residual, the directional term raises cost; when it points toward the goal, the term lowers cost. For two candidates, their predicted cost difference adds the differences in these directional and squared-error terms to their realized-cost difference, which can reverse their ordering.

In an initial-study pair, both candidates' squared prediction errors decrease, yet the failed candidate becomes preferred: its squared error decreases more, and the directional term also shifts against the successful candidate. Other reversals expose incorrect realized-cost preferences previously offset by prediction error (Appendix D). Accuracy alone therefore does not determine preferences.

Canonicalizing commands produces a similar average ranking gain in two additional checkpoints with different weights. Query-equal concordance rises from 0.7385 to 0.8438 and from 0.7472 to 0.8421 across 164 mixed queries per checkpoint, although 26 and 37 queries worsen. This nonuniform ranking response therefore recurs under different weights; independent-seed and architecture replication remain unverified (Appendix D.7).

4.4 How Physical Factors and Prediction Errors Enter the Score

PushT success depends on agent and block positions and block orientation. We render each realized endpoint with only the block orientation set to its goal value, and encode this static reference as y. Then $z - y$ is the edit-associated representation difference, and $y - z^g$ is the goal difference left after

editing. The difference $z - y$ is not a direct angle measurement. References are contact-free and in bounds; the edit does not test reachability.

With prediction error $e = \hat{z} - z$, the predicted cost becomes

$$c^{\text{pred}} = \|\hat{z} - y\|_2^2 + \|y - z^g\|_2^2 + A + B, \tag{5}$$

where

$$A = 2(z - y)^\top (y - z^g),$$
$$B = 2e^\top (y - z^g).$$

This rewrites the predicted cost from Equation 4 using the edited reference y. A measures how the edit-associated representation difference aligns with the goal difference remaining after the edit; B measures how prediction error aligns with that same difference. Each term contributes positively to predicted cost when the two vectors point in similar directions, negatively when they point in opposing directions, and zero when they are orthogonal. Candidate-specific changes can reverse rankings. The term $\|\hat{z} - y\|_2^2$ retains both differences. We remove A, B, or both, or double their sum; squared terms, actions, goal, and labels stay fixed. These oracle interventions need realized endpoints; Figure 3 shows the calculation and Table 2 reports effects and counterexamples.

Table 2: Evidence for the mechanisms in Figure 3. n counts episodes. Brackets give nominal 95% episode-bootstrap intervals. Negative changes in prediction error or misranking rate indicate improvement; "pp" denotes percentage points. Controls and full statistics: Appendix D.

What is tested	n	Observed effect	Measurement / variation
Cube: null-space exchanges			
Encoder sensitivity	128	Max. change 9.54×10^{-7}	First action layer
Prediction sensitivity	128	Max. change 2.38×10^{-6}	Predicted coordinates
Physical consequence	128	Mean separation 4.47 cm $[3.42, 5.60]$	12 exactly zero; 3 nonzero $\leq 10^{-12}$ m
TwoRoom: remove excess command magnitude, step 25			
Prediction error	512	Lower / higher / equal: 505 / 7 / 0	Episode-averaged squared error
Misranking rate	354	Change: -3.72 pp $[-6.03, -1.46]$	Improved / worsened / equal: 224 / 127 / 3; 158 lack success–failure pairs
Reacher: zero versus original velocity; squared error at environment step 5			
Squared error	512	Zero − original: -11.50 $[-12.68, -10.33]$	Two fixed candidates per episode
Closer to zero velocity	512	Both candidates: 484 / 512	Compared with original velocity
PushT: misranking change, 396 complete pools out of 2,048			
Remove A	396	Misranking change: -2.62 pp $[-4.59, -0.66]$	Improved / worsened / equal: 232 / 161 / 3
Remove B	396	Misranking change: -14.60 pp $[-16.16, -13.00]$	Improved / worsened / equal: 323 / 73 / 0
Remove $A + B$	396	Misranking change: -19.15 pp $[-20.49, -17.75]$	Improved / worsened / equal: 363 / 32 / 1

Cube responses are maximum absolute coordinate changes; each episode's physical separation is maximized over both exchanges and shared times. Retained-component controls: Appendix D.1. PushT uses legal references for every candidate; other coverage groups: Appendix D.5.

Of 2,048 additional evaluation episodes, 396 have valid references for all 300 candidates and at least one nonzero orientation edit. At step 25, one candidate physically dominates another if it has no greater agent-position, block-position, or orientation error and is better in at least one. Preferring the dominated candidate is a misranking, even when their binary labels match; exact ties remain separate.

Removing B reduces mean misranking more than removing A; removing both gives the largest reduction (Table 2). Doubling their sum reverses the average effect (Table 8). All four interventions retain counterexamples: either term can reinforce or offset wrong preferences, rather than being uniformly harmful. The realized-score comparison also improves on average after removing A, with reversals across episodes (Appendix D.5).

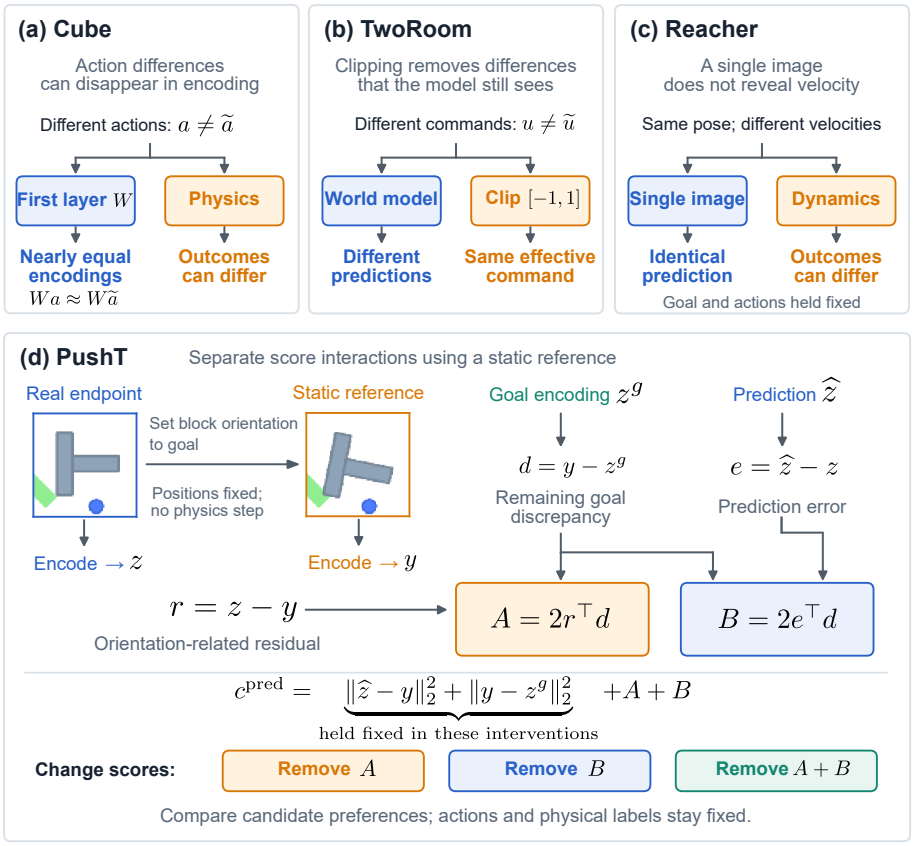


Figure 3: Physical distinctions and score interactions. (a) Cube nearly ignores motion-changing actions; (b) TwoRoom distinguishes execution-equivalent commands; (c) Reacher's single image hides velocity. (d) PushT encodes a static orientation edit as y; with realized z, predicted $\hat{z}$, and goal z^g, $r = z - y$, $e = \hat{z} - z$, and $d = y - z^g$ give $A = 2r^\top d$ and $B = 2e^\top d$. Removing A, B, or both alters only scores; squared terms remain fixed (Table 2).

In 69 zero-edit pools ($A = 0$), removing B lowers mean misranking by 15.17 percentage points. Another 1,393 pools have partial references and 190 none. Complete pools have 332/396 native successes versus 974/1,393 partial (Table 9), so effects remain conditional on reference eligibility. Prediction-error magnitude alone does not determine ranking; the model's distinctions and scoring also matter. Section 5 traces these effects into behavior.

5 When These Differences Affect Actions

Three paths test behavior: rescore and reselect in TwoRoom and PushT, edit fixed elites in Cube and TwoRoom, and execute a fixed output from changed Reacher states. In fixed final pools, TwoRoom input canonicalization changes top-30 membership and mean in all 512 episodes; oracle removal of B does both in all 396 complete-reference PushT pools. Execution tests whether those output changes affect success.

5.1 What Averaging Preserves and Changes

Averaging can pass nearly ignored action differences into execution or break candidate-level execution equivalence. In Cube, adding the same null-space displacement to every elite keeps membership and order fixed and changes predictions and costs only near numerical precision, yet shifts the mean and can change trajectories. This differs from the pairwise exchanges in Section 4.2.

In TwoRoom, we remove command excess from fixed top-30 elites before averaging. Each candidate executes identically after clipping, but the means can differ: scalar pairs (2, 0) and (1, 0) execute as

1 and 0.5. This yields three gains and five losses in 512 paired episodes, with an adjusted interval crossing zero (Table 3).

5.2 From Physical Changes to Task Success

Behavior depends on the changed interface: rescoring changes selected means throughout the tested TwoRoom and complete-reference PushT pools but seldom flips binary success; fixed-member edits act without reselection, and changed initial states act under fixed outputs (Table 3).

For TwoRoom, all 512 rescores change top-30 membership and mean, but only 24 flip success (20 gains, four losses; adjusted interval crosses zero). PushT oracle removal of B changes both in all 396 complete-reference pools, with 23 gains and one loss and a positive interval. Unchanged success does not establish unchanged trajectories.

In partial-reference PushT pools, only eligible scores change; 48 gains and 35 losses give an interval crossing zero. Both PushT interventions require realized endpoints and static references.

With membership fixed, the TwoRoom edit in Section 5.1 and opposite Cube shifts (7/10 and 5/6 gains/losses) have adjusted intervals crossing zero. With output fixed, zeroing Reacher's initial velocity gives 216 gains and 21 losses with a positive interval, while reversing it gives 96 gains and 128 losses with an interval crossing zero. Reacher changes physical state, not planner output.

Separate 256-episode TwoRoom replanning tests show no 25- or 50-step adjusted effect or interaction excluding zero for input/output canonicalization. For checkpoint A, fixed-pool concordance rises from 0.7385 to 0.8438, yet canonical-input search changes 50-step success from 217/256 to 208/256 (adjusted interval $[-11.35, 4.56]$ percentage points; Table 13). Output-rule and CEM variants show no uniform gain (Table 12); Table 7 changes first-call scores only. Local ranking gain does not establish closed-loop benefit.

6 Discussion and Conclusion

Section 5 shows that local ranking gains and changes to averaged actions need not improve closed-loop success. The relevant unit for design is the full path from representation to execution.

Reacher could infer missing velocity from image-action history, as in FlowMo-WM (Jiang et al., 2026), or receive measured velocity to distinguish visually identical states with different futures. Cube calls for an action encoder that retains directions capable of moving the block, which its first map currently suppresses. In TwoRoom, training on executed controls or searching directly within legal bounds could align prediction with execution; tested inference-time clipping showed no stable closed-loop gain. PushT could add a goal-relative position-and-angle score to native cost using readouts from predicted latents, as in LEAP (Pham & Bera, 2026), and the goal image.

These cases locate misjudgments in state input, action encoding, command conditioning, and scoring. Future designs should consider how these interfaces work together, beyond local changes. They offer starting points for LeWM and other latent world models.

Table 3: Paired success within 25 steps (native early termination). Family-adjusted paired-binary intervals; Cube uses episode bootstrap. PushT complete/partial: all/eligible scores.

Task / intervention	n	Gains / losses	Δ [adjusted interval] (pp)
Rescore and reselect; average original commands			
TwoRoom: remove excess	512	20 / 4	$+3.12\ [-0.61, 6.75]$
PushT complete: remove A	396	6 / 9	$-0.76\ [-4.82, 3.34]$
PushT complete: remove B	396	23 / 1	$+5.56\ [1.08, 9.78]$
PushT complete: remove $A + B$	396	23 / 2	$+5.30\ [0.61, 9.76]$
PushT partial: remove B	1,393	48 / 35	$+0.93\ [-1.64, 3.49]$
Replace commands with elite membership and order fixed			
TwoRoom: remove excess	512	3 / 5	$-0.39\ [-2.81, 2.06]$
Cube: positive null-space shift	128	7 / 10	$-2.34\ [-9.38, 4.69]$
Cube: negative null-space shift	128	5 / 6	$-0.78\ [-6.25, 4.69]$
Change initial state with output fixed			
Reacher: reverse velocity	512	96 / 128	$-6.25\ [-16.29, 3.93]$
Reacher: zero velocity	512	216 / 21	$+38.09\ [29.12, 46.22]$

AI USE STATEMENT

Generative AI tools were used to improve and review analysis and evaluation code initially written by the authors; provide feedback on experimental methodology and result interpretation; and support literature search and summarization, manuscript organization and drafting, language editing, figure and table preparation, and LaTeX formatting. They were not used to generate experimental data or reported results. The authors defined the research questions and experimental protocols, developed the original code, verified reported results against recorded experimental artifacts, and made the final decisions on evidence interpretation and claims. The authors reviewed all work involving generative AI and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

Appendix A.1 specifies the fixed models, input processing, and planning configuration. Appendix A describes task predicates, sampling, state restoration, and replay checks; Appendices B and D give the statistical procedures and task-specific interventions.

REFERENCES

Hidehisa Arai, Keishi Ishihara, Tsubasa Takahashi, and Yu Yamaguchi. ACT-Bench: Towards Action Controllable World Models for Autonomous Driving, 2024. URL https://arxiv.org/abs/2412.05337.

Jack Boylan and Chris Hokamp. No Gaussian Required: Contrastive Inverse Dynamics for JEPA World Models, 2026. URL https://arxiv.org/abs/2608.17542v1.

Peterson Co, Sicheng Hu, Chunxuan Jiao, Hongyang Cheng, Yulin Luo, Yijie Xu, Sixiang Chen, Zhongxia Zhao, Zihao Wang, DaFeng Chi, Peidong Liu, YuTong Chen, Henghua Liu, Zhihao Yuan, Huizhu Jia, Yuzheng Zhuang, Tianle Zhang, Liang Lin, Huajie Tan, and Shanghang Zhang. WorldSimProbe: Diagnosing Simulator Faithfulness in Action-Conditioned World Models for Embodied Manipulation, 2026. URL https://arxiv.org/abs/2608.09298v1.

Amir-Massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-Aware Loss Function for Model-based Reinforcement Learning. In *Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1486–1494. PMLR, 2017. URL https://proceedings.mlr.press/v54/farahmand17a.html.

Zhenfeng Gan, ZiTong Zeng, Jiajun Cheng, Yeke Song, Yongyi Tang, and Xueqian Wang. ActSWM: Action-Sensitive World Models for Long-Horizon Planning in Open-World Games, 2026. URL https://arxiv.org/abs/2607.26712v1.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2170–2179. PMLR, 2019. URL https://proceedings.mlr.press/v97/gelada19a.html.

Christopher Grimm, Andre Barreto, Satinder Singh, and David Silver. The Value Equivalence Principle for Model-Based Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5541–5552, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3bb585ea00014b0e3ebe4c6dd165a358-Abstract.html.

Jiaming Hu, Yan Zheng, and Tian Wang. SCALE: State-Calibrated Latent Embeddings for JEPA Planning in the Right Geometry, 2026. URL https://arxiv.org/abs/2608.16287v1.

Xiaodi Huang, Ziyi Ding, Jingtian Wan, Yuchen Liu, Yuan Zhang, Xiao-Ping Zhang, Jiayu Chen, Zhang Zhang, and Tao Huang. Traj-LeWM: Path-Aware World-Model Planning via Latent Trajectory Cost, 2026. URL https://arxiv.org/abs/2608.14125v1.

Petr Ivashkov, Randall Balestriero, and Bernhard Schölkopf. Sensorimotor World Models: Perception for Action via Inverse Dynamics, 2026. URL https://arxiv.org/abs/2606.20104v1.

Yitao Jiang, Luyang Zhao, Muhao Chen, and Devin Balkcom. FlowMo-WM: A World Model with Object Momentum and Hidden Ambient Drift, 2026. URL https://arxiv.org/abs/2606.13817v1.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 761–770. PMLR, 2020. URL https://proceedings.mlr.press/v120/lambert20a.html.

Liangyu Li, Shengzhi Wang, Libin Qiu, Mingliang Xiong, and Qingwen Liu. World model control by trajectory reachability metrics, 2026. URL https://arxiv.org/abs/2605.22164.

Shuaijun Liu, Chengyu Wu, Qifu Wen, Feiyang You, Chenglong Zhang, Shuyang Hao, Xi Lin, and Ningxin Su. D-JEPA: A Decision-Aligned Latent World Model. Author manuscript, 2026. URL https://nebulis-lab.com/D-JEPA/static/paper/D-JEPA.pdf.

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

Phu Pham and Aniket Bera. Latent Energy Action Planning with World Models, 2026. URL https://arxiv.org/abs/2609.03294v1.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient Cross-Entropy Method for Real-time Planning. In *Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1049–1065. PMLR, 2021. URL https://proceedings.mlr.press/v155/pinneri21a.html.

Julian Quevedo, Ansh Kumar Sharma, Yixiang Sun, Varad Suryavanshi, Percy Liang, and Sherry Yang. WorldGym: World Model as An Environment for Policy Evaluation. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/7f5e909ac0324db03506b380c695ffaf-Paper-Conference.pdf.

Gautam Singh, Skand Peri, Junghyun Kim, Hyunseok Kim, and Sungjin Ahn. Structured world belief for reinforcement learning in POMDP. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9744–9755. PMLR, 2021. URL https://proceedings.mlr.press/v139/singh21a.html.

Joyjeet Singh. The Objective Is the Bottleneck: Latent World Models Encode What Their Planners Cannot Use, 2026. URL https://arxiv.org/abs/2608.12959v1.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *Transactions on Machine Learning Research*, 2026. doi: 10.48550/arXiv.2512.24497. URL https://arxiv.org/abs/2512.24497. arXiv:2512.24497.

Adam J. Thorpe, Stepan Tretiakov, Cheng-Hsi Hsiao, Su Ann Low, Xingjian Li, Hassan Iqbal, Neel P. Bhatt, Ufuk Topcu, and Krishna Kumar. Physically Viable World Models: A Case for Query-Conditioned Embodied AI, 2026. URL https://arxiv.org/abs/2605.30542.

Wei-Cheng Tseng, Jinwei Gu, Qinsheng Zhang, Hanzi Mao, Ming-Yu Liu, Florian Shkurti, and Lin Yen-Chen. Scalable Policy Evaluation with Video World Models, 2025. URL https://arxiv.org/abs/2511.11520.

Donna Vakalis. The Intervention Gap in Latent World Models, 2026. URL https://arxiv.org/abs/2608.29998v1.

Kanpat Vesessook and Kevin Yang. Hidden Failure Modes in Latent World-Model Planning from Offline Data. ICML 2026 DEMO Workshop manuscript, 2026. URL https://raw.githubusercontent.com/24GUNV/LeWMRO/main/paper.pdf.

Jiawei Wang, Ke Rui, Yushen Zuo, Yichun Feng, and Minglei Li. Decision-Metric Alignment in Latent World Models: Diagnostics and Action-Conditioned Objectives for MPC Planning, 2026. URL https://arxiv.org/abs/2608.18746v1.

Haodong Yan, Jiaguan Zhu, Mingyuan Jia, Ruiqing Yin, Junjie He, Zhide Zhong, Junfeng Li, Jinxuan Lu, Hengtao Li, Tianran Zhang, Jiayi Chen, Wenxuan Song, Wen Chen, Yuxiang Gao, and Haoang Li. Is forward prediction enough? physical state grounding for JEPA world models, 2026. URL https://arxiv.org/abs/2608.06799.

Yang Yu, Shiyuan Zhang, Yifei Sheng, Haoxiang Ren, and Haoxin Lin. How Should World Models Be Evaluated for Embodied Decision-Making? A Decision-Making-Centric Position, 2026. URL https://arxiv.org/abs/2606.15032.

Jianhao Yuan, Yu Yuan, Benjamin Ramtoula, Lukas Vierling, Paul Newman, Lars Kunze, Philip Torr, and Daniele De Martini. Beyond Visual Quality: A Study of Test-Time Planning with World Action Models, 2026. URL https://arxiv.org/abs/2609.24745v1.

Xi Zeng, Haojie Ren, and Ziying Song. PhyLatent: Learning Dynamics-Relevant Representations for JEPA World Models, 2026. URL https://arxiv.org/abs/2608.05720v1.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning Invariant Representations for Reinforcement Learning without Reconstruction. In *International Conference on Learning Representations*, 2021. doi: 10.48550/arXiv.2006.10742. URL https://arxiv.org/abs/2006.10742.

Zhenghao Zhang, Yuanxiang Wang, Zhenyu Guan, Yujia Yang, Bingkang Shi, Tianyu Zong, Hongzhu Yi, Guoqing Chao, Xingchen Chen, Tiankun Yang, Chenxi Bao, Tao Yu, Jingjing Zhou, and Jungang Xu. Delta-JEPA: Learning action-sensitive world models via latent difference decoding, 2026. URL https://arxiv.org/abs/2606.31232.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

Penghao Zhu, Salvatore Penachio, Kaustav Mukherjee, and Aneesh Jonelagadda. Spectral-Target Physical Latent Structuring for JEPA-Style World Models, 2026. URL https://arxiv.org/abs/2609.04264v1.

A Task Definitions and Evaluation Details

A.1 Models, inputs, and planning configuration

We evaluate one fixed LeWM model per task; Table 4 links to the corresponding public model and source dataset pages. Each model runs in evaluation mode with frozen parameters, float32 computation, and TF32 disabled. The planner scores the final predicted 192-dimensional representation by its squared Euclidean distance to the encoded goal.

Task	Public model	Public dataset
Reacher	LeWM Reacher	Reacher trajectories
TwoRoom	LeWM TwoRoom	TwoRoom trajectories
Cube	LeWM Cube	Cube single-expert trajectories
PushT	LeWM PushT	PushT expert training trajectories

Table 4: Public LeWM models and source datasets. Model and dataset names link to their respective Hugging Face pages.

Each planning call observes one image, scaled to float32, ImageNet-normalized, and resized to 224×224; predicted representations provide context for later rollout steps. CEM draws 300 candidate sequences in each of 30 rounds and retains 30 elites per round; it uses batch size one and variance scale 1.0. Each sequence spans five model steps of five environment commands. The native output is the round-30 elite mean, with warm starts on later planning calls.

Two additional TwoRoom checkpoints with different weights are evaluated on the same 256 episodes as the base planning study. Their fixed-pool comparisons reuse the base model's recorded first-call physical candidates and realized endpoints, which each checkpoint encodes and scores separately. Each checkpoint runs its own closed-loop search.

Actions are standardized componentwise using source-dataset statistics, processed in five-command blocks, inverse-transformed in float32, and passed to the environment. TwoRoom then clips commands to $[-1, 1]$; Cube retains its native target limits.

A.2 Task success and action timing

Reacher succeeds when every joint satisfies $|q_j - q_j^g| < 0.05$ radians, using the original coordinate difference. TwoRoom requires agent–target Euclidean distance below 16 environment units. Cube requires block-0 position to be within 0.04 m of the target in three dimensions, with no orientation criterion. PushT requires the joint four-dimensional vector of agent and block positions to lie within Euclidean distance 20 of its goal, and the block's shortest circular angle error to be below $\pi/9$. Candidate ranking uses the success label at environment step 25; behavioral comparisons use success at any point within the reported window. Missing or invalid observations leave success undefined.

The 25-step window counts environment action steps. Five consecutive environment commands form one model input block, and the endpoint prediction spans five such model steps. Reacher repeats each environment action over two internal steps and can stop on success within that repeat. We match its native and fixed-time trajectories at the internal physical step of first success.

A.3 Query construction and initial states

We sample episodes without replacement, taking one start from each. Each start has the required state fields and is paired with a goal 25 steps later in the same episode; it lies outside that goal's success region. For each task, 150 training, 50 validation, and 150 test episodes form disjoint splits created for this analysis from source trajectories; the base comparison uses the test split.

For Cube, all 350 sampled source starts lie outside their query goals' success regions. The 450 test-split closed-loop branches (three per query) lack recorded step-zero task states, so this initial-success check covers source starts only.

A.4 PushT observations and restoration

For PushT, we restore the recorded agent position and velocity, block position, and block angle. The block's linear and angular velocities come from the seeded reset. Within each query, paired candidates and branches start from the same reconstructed state. The current environment renders the initial and subsequent observations; a separate environment with the same seed renders the goal image from its recorded state. These rendered images are the planner's visual inputs.

A.5 Replay and paired-execution checks

The base comparison contains 600 complete query pairs. Exact cost ties follow the recorded native elite order and candidate identities. Across formal queries, we checked that candidate identities and output commands matched the recorded runs, together with the controller context, action conversion, and post-search random state. Targeted trajectory replays matched native and fixed-time execution through the first native success, including Reacher's internal stopping step. A further check on one continued query per task reproduced closed-loop execution using the cached first output.

B Statistical Procedures

Queries are the units of reported averages, and source episodes are the bootstrap resampling units. We adjust intervals separately for the comparison families specified below.

B.1 Supplementary comparisons

For the supplementary success-within-25 analysis, we classify each success–failure pair as correctly ordered, tied, or reversed under realized scoring and then under predicted scoring. These realized-to-predicted comparisons give nine transition categories. Reversing the scoring direction swaps correct-to-reversed and reversed-to-correct transitions, as in Section 4.1. We compute each category's fraction over all success–failure pairs within a query, then average equally across queries. Latent distances use the same two-stage aggregation. The 25-step input-by-output interaction is the realized-minus-predicted success-rate change for the lowest-cost candidate minus the corresponding change for the top-30 mean (Appendix C.2). Transition, geometry, membership, timing, and interaction analyses were developed after inspecting the initial results and are exploratory.

B.2 Paired bootstrap and comparison families

We compute ranking effects within queries and form percentile intervals from 20,000 source-episode bootstrap resamples, keeping paired conditions together. Exact ties are evaluated at stored precision. Nominal 95% intervals use the 2.5th and 97.5th percentiles. For a Bonferroni family of m comparisons, the adjusted interval uses quantiles $0.05/(2m)$ and $1 - 0.05/(2m)$.

Three bootstrap families each contain one contrast per task ($m = 4$): realized-minus-predicted concordance at fixed step 25 (Table 1), realized success-or-timeout concordance minus 0.5 (Table 5), and the 25-step input-by-output interaction (Appendix C.2). Each family therefore uses 98.75% adjusted intervals. The 50-step paired success comparison in Table 7 uses the binary interval construction below.

B.3 Paired binary success intervals

For two conditions, let n_{ab} count queries with success a under the first and b under the second, where $a, b \in \{0, 1\}$. With $n = n_{00} + n_{01} + n_{10} + n_{11}$ complete pairs, the second-minus-first success-rate difference is

$$\widehat{\Delta} = \frac{n_{01} - n_{10}}{n}. \tag{6}$$

The denominator includes both-success and both-failure queries. A query with a missing outcome in either condition is recorded as an incomplete pair and excluded from this complete-pair estimate.

For a difference interval with target error rate α, we construct a two-sided Clopper–Pearson interval for each discordance probability p_{01} and p_{10}, each using error rate $\alpha/2$ and the full denominator n.

If their bounds are $[L_{01}, U_{01}]$ and $[L_{10}, U_{10}]$, the conservative difference interval is

$$[L_{01} - U_{10},\ U_{01} - L_{10}]. \tag{7}$$

This interval retains sampling uncertainty even when all paired outcomes agree. An interval spanning zero is compatible with gains, losses, and zero net change. Nominal intervals use $\alpha = 0.05$; Bonferroni-adjusted intervals use $\alpha = 0.05/m$ for the following families.

The four-task 50-step comparison in Table 7 has $m = 4$. For the mechanism interventions in Table 10, Reacher's two velocity contrasts retain the original four-comparison adjustment. The four TwoRoom contrasts, four PushT complete-pool contrasts, and four PushT partial-pool contrasts each form a separate $m = 4$ family. Cube's positive and negative shifts form an $m = 2$ family and use 20,000 paired episode-bootstrap resamples clustered by source episode.

The TwoRoom planning study (Table 11) groups four simple input/output effects at two horizons, 25 and 50 steps, into an $m = 8$ family. Its two horizon-specific interactions form a separate $m = 2$ family evaluated with paired episode bootstrap. These interactions and Cube's signed-shift contrasts use exploratory bootstrap intervals; the other success contrasts use the conservative paired-binary construction.

For the output-rule comparisons in Table 12, argmin and mean-as-candidate span two model-input settings and two horizons ($m = 8$); weighted elites span the same inputs and horizons ($m = 4$). The six frozen-CEM configurations at two horizons form a separate $m = 12$ family.

The checkpoint comparison uses another $m = 12$ family: two checkpoints, three changes from each checkpoint's raw-input/native-output baseline (canonical input, canonical output, or both), and two horizons. Table 13 shows the four canonical-input/native-output contrasts from this full family, retaining the 12-comparison adjustment.

C Supplementary Results and Analyses

Table 5 reports concordance with success within 25 steps, using realized representations at first success or timeout. The transition and geometry analyses below use the same within-window success definition with representations evaluated at fixed step 25. Moving from realized to predicted scoring, correct-to-reversed transitions exceed reversed-to-correct transitions in every task: 27.04% versus 4.54% in Reacher, 19.25% versus 12.58% in TwoRoom, 37.54% versus 7.56% in Cube, and 29.74% versus 5.42% in PushT. These fractions average equally across mixed-outcome queries, with ties recorded as separate transition categories (Appendix B.1).

Table 5: Concordance for the supplementary success-within-25 analysis. Realized scoring uses the observation at first success or timeout. n counts equally weighted mixed-outcome queries out of 150; intervals are adjusted across four tasks (98.75%).

Task	n	Concordance ↑	Adjusted interval for concordance -0.5
Reacher	148	0.9977	$[0.4972, 0.4982]$
TwoRoom	89	0.5603	$[-0.0225, 0.1439]$
Cube	91	0.7918	$[0.2092, 0.3663]$
PushT	145	0.8906	$[0.3627, 0.4144]$

C.1 Candidate Availability and Scoring-Based Selection

Within the native 25-step execution window, at least one of 300 candidates succeeds in 149/150 Reacher, 120/150 TwoRoom, 103/150 Cube, and 145/150 PushT queries. Among these pools with successful candidates, the lowest predicted-cost candidate fails in 93/149 Reacher and 35/103 Cube queries; realized scoring reduces these counts to 0/149 and 7/103. Replacing predicted with realized costs increases successful top-30 membership in 131 Reacher and 77 PushT queries, while decreasing it in nine and ten, respectively. These results locate selection errors within pools that contain successful candidates and show that rescoring can increase or decrease successful elite membership.

C.2 Scoring Inputs and Output Rules

On 150 paired queries per task, we cross predicted or realized endpoint scoring with the lowest-cost candidate or top-30 mean, using the same final pool and success within 25 execution steps (Table 6). Realized scoring uses encoded fixed-25 execution outcomes as a diagnostic reference. Its success gain is larger for the lowest-cost candidate than for the mean in all four tasks.

The exploratory paired-query interaction measures this difference in gains (Appendix B.1). Estimates and 98.75% intervals adjusted across the four tasks are, in percentage points: Reacher $+6.67$ $[-1.33, 15.33]$, TwoRoom $+5.33$ $[0.67, 10.67]$, Cube $+8.67$ $[1.33, 16.67]$, and PushT $+6.67$ $[-0.67, 14.00]$. The intervals are above zero for TwoRoom and Cube and span zero for Reacher and PushT. Thus, TwoRoom and Cube provide adjusted support for a smaller scoring-input benefit after elite averaging in this 150-query comparison.

Table 6: Success within the first 25 environment steps for four output conditions on the same final pool from the first planning call. Predicted and real inputs use the native distance to the goal at step 25. The real inputs are encoded fixed-duration replay observations. Argmin selects the lowest-cost candidate, whereas mean averages the top 30 action sequences and is executed separately. Each cell reports successes among 150 complete queries per task. Appendix C.2 reports the paired interaction estimates and adjusted intervals.

	Predicted input		Real input	
Task	Argmin ↑	Mean ↑	Argmin ↑	Mean ↑
Reacher	56	56	149	139
TwoRoom	101	105	116	112
Cube	68	69	96	84
PushT	99	104	142	137

Continuation under native replanning. Table 7 follows the first-call scoring intervention through a total budget of 50 environment steps. The diagnostic branch scores the first final candidate pool using realized fixed-25 endpoint representations and executes the top-30 mean. Both branches then continue with native control after the first 25-step segment when needed. This comparison uses future execution outcomes for the initial scoring intervention and measures success after subsequent native replanning.

Table 7: Paired success within a total budget of 50 environment steps. The real-input branch substitutes realized fixed-25 endpoint representations for predicted endpoints when scoring the final candidate pool of the first planning call. Both branches use the native distance and top-30 mean, followed by native control after the first 25-step segment if needed. The real-input branch uses future execution outcomes as a diagnostic intervention. All 150 query pairs per task are complete. Δ is real-input minus native success rate in percentage points (pp). Paired binary intervals are nominal 95% and Bonferroni-adjusted 98.75% for the four-task comparison family.

Task	Native	Real-input	Δ (pp)	95% CI	Adjusted CI
Reacher	127/150	150/150	$+15.33$	$[6.47, 23.11]$	$[4.53, 24.91]$
TwoRoom	121/150	122/150	$+0.67$	$[-4.99, 6.24]$	$[-6.21, 7.46]$
Cube	72/150	85/150	$+8.67$	$[-0.07, 16.74]$	$[-1.93, 18.44]$
PushT	127/150	141/150	$+9.33$	$[1.87, 16.05]$	$[0.26, 17.66]$

C.3 Outcome Separation and Local Ranking Errors

For each successful–failed candidate pair, we compute the squared L2 distance between its two realized representations and between its two predicted representations at step 25. Success is defined within the native 25-step window. We average within queries and then equally across queries supporting the relevant group. In Reacher, the mean squared distance increases from 79.65 under realized representations to 83.03 under predicted representations while concordance decreases. In

Cube, this distance contracts for 95.04% of pairs that change from correctly ordered to reversed and 94.72% of pairs that remain correctly ordered. The two Cube groups thus have similar contraction frequencies despite different ranking outcomes.

C.4 Local ranking examples

For Reacher and Cube, we show the first eligible correct-to-reversed pair in query and candidate identity order. Physical errors below come from native success-or-timeout events; scores use representations from independent fixed-25 replays.

In Reacher, one candidate succeeds at step 24 with maximum joint error 0.0277 radians, while the other fails at step 25 with error 0.2044 radians. Their realized costs are 16.99 and 56.94, respectively, favoring the successful candidate; predicted costs are 63.73 and 28.85, favoring the failed candidate.

In Cube, the successful and failed candidates terminate at steps 17 and 25 with block errors 3.93 and 5.07 cm. Their realized costs are 130.85 and 269.96, while predicted costs are 113.16 and 82.91. The successful-minus-failed cost margin therefore changes from -139.12 to $+30.25$. Both the Reacher and Cube pairs lie outside the selected top 30, and both queries have successful outputs.

D Mechanism Evidence

D.1 Action-component exchanges and execution equivalence

Each Cube block contains five normalized five-dimensional commands. The first action layer is a rank-10 10×25 matrix with nullity 15; float64 SVD defines orthogonal row- and null-space projectors. We exchange null-space components between the two lowest-cost candidates, inverse-transform in float32, and execute with native target limits and no added clipping. Complementary exchanges change retained components instead.

Across 128 confirmation episodes, null-space exchanges change the first-layer response by at most 9.54×10^{-7}, a predicted coordinate by 2.38×10^{-6}, and cost by 4.58×10^{-5}; retained-component responses reach 2.54. Maximum block-position separation averages 4.47 cm (95% bootstrap interval 3.42–5.60 cm), retaining 12 exact zeros and three machine-precision effects. The original 32 and additional 128 episodes do not overlap.

A separate control compares equal-norm null-space and retained-component displacements around feasible action centers, with commands inside the action bounds over the tested window. All 14 null-space directions produce physical differences, including 10 before any recorded native-limit activation. Two directions change success within the first five steps, one gain and one loss, with neither paired branch activating native limits during execution. The eight directions tested over the full 25-step window preserve step-25 success labels.

D.2 Fixed-member aggregation interventions

Figure 4 illustrates a fixed-member TwoRoom trajectory change and the three behavioral paths.

For Cube, a native elite supplies a common projected displacement added to or subtracted from every elite; a rescaled row-space displacement is the control. Native selection and float32 averaging are unchanged. All 256 signed null-space conditions preserve elite identities and order. Of these comparisons, 232 have nonzero trajectory differences, including machine-scale differences. Paired success counts appear in Table 10.

For TwoRoom, fixed-member conditions retain the original top-30 identities and order while removing all or half of each command's excess magnitude before float32 averaging. Individual executed commands remain unchanged after inverse transformation and clipping, but the mean can change. The expanded 512-episode test is separate from rescoring and uses paired initial states with native early termination.

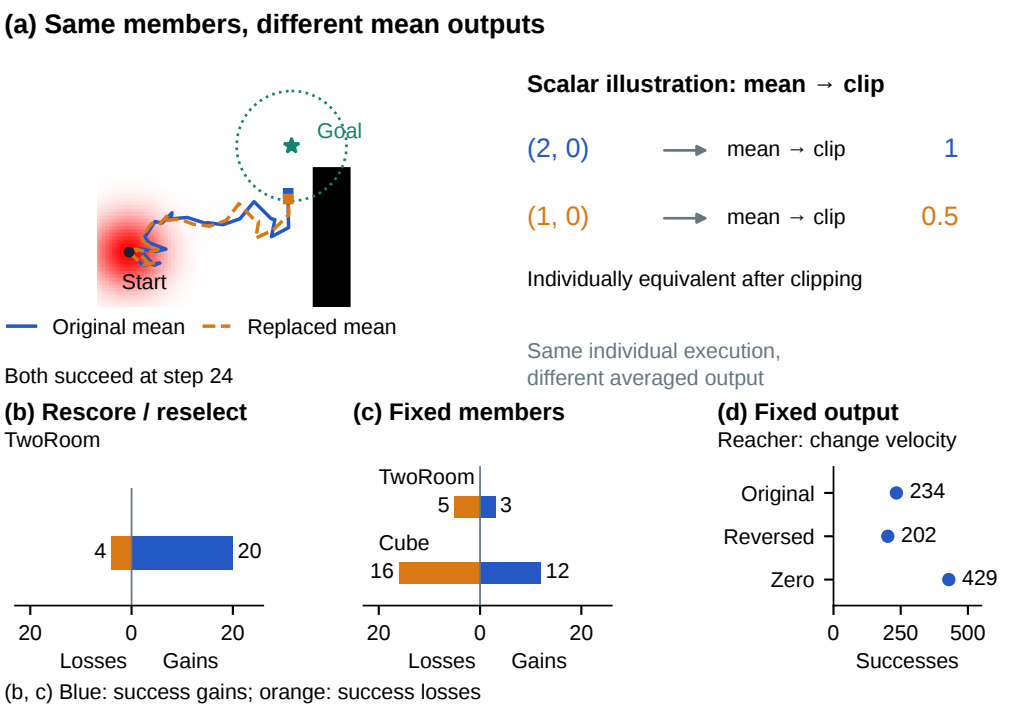


Figure 4: Three paths to behavior. (a) Fixed-member TwoRoom mean outputs follow different paths but both succeed at step 24; this initial-study case is separate from the expanded samples. The schematic scalar commands illustrate clipping and averaging. (b,c) Paired success gains/losses: TwoRoom, 512 episodes per intervention; Cube, 256 signed comparisons from 128 episodes. (d) Reacher: success counts in 512 episodes under fixed outputs and altered initial velocities. Full paired counts appear in Table 10.

D.3 Samples and scoring mechanisms

Mechanism groups use additional episodes from the same task datasets and checkpoints as the base 150-query comparisons. Episode lists for Cube, Reacher, TwoRoom, and PushT were frozen before outcomes were collected and exclude previously exposed source episodes. Reacher uses two fixed candidates per episode, TwoRoom averages squared prediction error across all 300 candidates, and Cube keeps its paired exchanges within each bootstrap unit.

The equal-magnitude TwoRoom control rotates the replacement-induced predicted endpoint change by 90 degrees in adjacent coordinate pairs. Command replacement lowers mean squared prediction error in all 16 initial episodes, whereas the rotated control increases it in all 16.

D.4 TwoRoom cost-margin examples

In the expanded TwoRoom group, 354 episodes contain both successful and failed candidates at step 25. Their equally weighted misranking change is -3.72 percentage points (95% bootstrap interval $[-6.03, -1.46]$); the other 158 episodes lack such pairs. We select the first worsening and first improving examples in query and candidate identity order. In the worsening example, the successful-minus-failed cost margin changes from -23.06 to 13.23 even though both candidates' squared prediction errors decrease. In the improving example, the margin changes from 20.78 to -237.99. The worsening example shows that lower prediction error for both candidates can accompany an incorrect preference.

D.5 Expanded PushT coverage and scoring effects

We retain static orientation references that pass contact and bounds checks.

Table 8: PushT dominance misranking changes under predicted scoring, averaged over 396 complete pools from 2,048 new episodes. Negative changes indicate improvement. Intervals are nominal 95% episode-bootstrap intervals; counts report episodes, not candidate pairs.

Intervention	Change (pp)	95% interval	Better / worse / unchanged
Remove A	-2.62	$[-4.59, -0.66]$	232 / 161 / 3
Remove B	-14.60	$[-16.16, -13.00]$	323 / 73 / 0
Remove $A + B$	-19.15	$[-20.49, -17.75]$	363 / 32 / 1
Double $A + B$	$+40.05$	$[+38.40, +41.71]$	9 / 387 / 0

Table 9: PushT legal-reference coverage over 2,048 episodes. Candidate counts are within each 300-candidate pool; native success means success at any time within 25 steps under native early termination.

Group	Episodes	Eligible median	Nonzero-edit median	Native success
Complete	396	300	300	332 / 396
Partial	1,393	89	72	974 / 1,393
All-legal zero-angle	69	300	0	54 / 69
Unavailable	190	0	0	106 / 190

The 2,048 frozen episodes contain 396 complete, 1,393 partial, 69 all-legal-zero-angle, and 190 unavailable pools (Table 9). Complete-pool scoring effects appear in Table 8. Removing A from realized costs changes mean misranking by -1.94 percentage points (95% interval $[-2.39, -1.52]$). In all-zero-angle pools, $A = 0$ and removing B changes mean misranking by -15.17 points ($[-17.37, -12.82]$; 64 improve, five worsen). In partial pools, removing B changes it by -2.11 points ($[-2.75, -1.45]$; 707 improve, 667 worsen, 19 unchanged). This intervention modifies eligible candidates and retains native scores for the remaining candidates. Results are reported separately for each reference-coverage group. Intervals use 10,000 paired episode-bootstrap resamples and are nominal descriptive intervals.

D.6 Behavior propagation

In TwoRoom, rescoring with canonicalized model inputs or the rotated-delta control changes elite membership and mean output in all 512 episodes. Fixed-member command replacements instead preserve membership and order by construction. In PushT complete pools, removing B, removing $A + B$, or doubling $A + B$ changes both in every pool; removing A changes membership in 386 and the mean in 387. Table 10 reports paired success within 25 steps under native early termination. Reacher changes velocity under a fixed output; TwoRoom separates rescoring from fixed-member replacement; PushT remains an oracle diagnostic using realized endpoints and edited references.

In PushT, removing B yields nine gains and no losses in all-zero-angle pools, and 48 gains and 35 losses in partial pools with an interval crossing zero. Unavailable pools retain native outputs.

Pairwise candidate-component exchanges in Cube yield 15 gains and 16 losses across 320 null-space comparisons from the combined 160 episodes. Table 10 reports the separate fixed-member mean-output shifts on 128 episodes. All denominators retain cases with zero response or native-limit activation.

D.7 Planning interfaces, output rules, and weight variants

The TwoRoom planning study crosses raw or canonical model inputs with native or canonicalized output commands over 256 paired episodes (Table 11). Its two input-by-output interactions are 0.00 percentage points with adjusted interval $[-1.17, 1.17]$ at step 25 and $+1.17$ points with $[-1.17, 3.91]$ at step 50; all adjusted simple-effect and interaction intervals cross zero.

The output-rule study compares three alternatives. Argmin executes the minimum-cost sequence from the final 300-candidate pool. Mean-as-candidate scores the native top-K mean as one additional candidate and retains the mean on an exact tie.

Table 10: Paired task success under the expanded mechanism interventions. Gains and losses count changes in success, not trajectory. Intervals are family-adjusted conservative paired-binary intervals, except Cube's adjusted episode-bootstrap intervals across two signs. PushT complete rows intervene on every candidate; partial rows retain native scores for ineligible candidates.

Task / intervention	n	Success: old $\rightarrow$ new	G / L	Δ [adj. interval] (pp)
Reacher: reverse velocity	512	$234 \rightarrow 202$	96 / 128	-6.25 $[-16.29, 3.93]$
Reacher: zero velocity	512	$234 \rightarrow 429$	216 / 21	$+38.09$ $[29.12, 46.22]$
TwoRoom: rescore excess	512	$356 \rightarrow 372$	20 / 4	$+3.12$ $[-0.61, 6.75]$
TwoRoom: rescore control	512	$356 \rightarrow 355$	6 / 7	-0.20 $[-3.18, 2.80]$
TwoRoom: fixed remove	512	$356 \rightarrow 354$	3 / 5	-0.39 $[-2.81, 2.06]$
TwoRoom: fixed halve	512	$356 \rightarrow 357$	1 / 0	$+0.20$ $[-1.12, 1.54]$
PushT complete: remove A	396	$332 \rightarrow 329$	6 / 9	-0.76 $[-4.82, 3.34]$
PushT complete: remove B	396	$332 \rightarrow 354$	23 / 1	$+5.56$ $[1.08, 9.78]$
PushT complete: remove $A + B$	396	$332 \rightarrow 353$	23 / 2	$+5.30$ $[0.61, 9.76]$
PushT complete: double $A + B$	396	$332 \rightarrow 248$	5 / 89	-21.21 $[-28.45, -13.32]$
PushT partial: remove A	1,393	$974 \rightarrow 979$	27 / 22	$+0.36$ $[-1.65, 2.37]$
PushT partial: remove B	1,393	$974 \rightarrow 987$	48 / 35	$+0.93$ $[-1.64, 3.49]$
PushT partial: remove $A + B$	1,393	$974 \rightarrow 992$	49 / 31	$+1.29$ $[-1.23, 3.80]$
PushT partial: double $A + B$	1,393	$974 \rightarrow 705$	20 / 289	-19.31 $[-23.15, -15.31]$
Cube: positive null shift	128	$67 \rightarrow 64$	7 / 10	-2.34 $[-9.38, 4.69]$
Cube: negative null shift	128	$67 \rightarrow 66$	5 / 6	-0.78 $[-6.25, 4.69]$

Table 11: TwoRoom input/output interventions throughout closed-loop replanning on 256 paired episodes. Raw/canonical denotes model input and native/canonical the output interface. Simple effects use family-adjusted paired-binary intervals (family size 8); interactions use paired episode-bootstrap intervals (family size 2).

Input / output arm	Success by 25	Success by 50
Raw / native	187 / 256	216 / 256
Canonical / native	190 / 256	198 / 256
Raw / canonical	187 / 256	214 / 256
Canonical / canonical	190 / 256	199 / 256
Simple effect	**Δ_{25} [adjusted interval]**	**Δ_{50} [adjusted interval]**
Input at native output	$+1.17$ $[-9.49, 11.77]$	-7.03 $[-15.48, 1.84]$
Input at canonical output	$+1.17$ $[-9.49, 11.77]$	-5.86 $[-14.64, 3.27]$
Output at raw input	0.00 $[-3.36, 3.36]$	-0.78 $[-5.41, 3.92]$
Output at canonical input	0.00 $[-2.49, 2.49]$	$+0.39$ $[-3.34, 4.11]$
Input $\times$ output interaction	0.00 $[-1.17, 1.17]$	$+1.17$ $[-1.17, 3.91]$

Weighted elites average the final top-K sequences with weights proportional to $\exp[-(c - \min c)/\max(\mathrm{sd}(c), \epsilon)]$, normalized to sum to one. The minimum and standard deviation are computed over these elite costs, and ϵ is float64 machine epsilon. Frozen-CEM rows vary N, K, or the number of rounds from the native $N = 300$, $K = 30$, 30-round setting; rows that change both N and K are joint configurations. All conditions are paired against the raw-input native-mean baseline with 187/256 successes by step 25 and 216/256 by step 50 (Table 12).

The $N = 300, K = 60$ configuration worsens success at both horizons after adjustment; no tested alternative has an adjusted positive effect at both horizons. The checkpoint audit reuses the base TwoRoom model's native/native first-call final 300-candidate pools from the same 256 episodes. The identical physical command pools and realized endpoints are encoded and scored separately by the two additional checkpoints, so latent quantities are compared only within checkpoint. Both improve fixed-pool concordance under canonical input, while their separately generated canonical-input closed-loop differences are nonpositive and all adjusted intervals cross zero (Table 13).

Table 12: TwoRoom output-rule and frozen-CEM extensions on 256 episodes, paired against native/native (187 successes by step 25; 216 by step 50). Cells give success (gains/losses) and percentage-point change with family-adjusted paired-binary intervals; N/K rows are joint configurations.

Condition	S_{25} (G/L)	Δ_{25} [interval]	S_{50} (G/L)	Δ_{50} [interval]
Output rules with raw model input				
Argmin	186 (4/5)	-0.39 $[-5.45, 4.70]$	212 (4/8)	-1.56 $[-7.16, 4.15]$
Mean as candidate	186 (4/5)	-0.39 $[-5.45, 4.70]$	213 (4/7)	-1.17 $[-6.61, 4.35]$
Weighted elites	187 (0/0)	0.00 $[-2.23, 2.23]$	213 (1/4)	-1.17 $[-5.08, 2.84]$
Output rules with canonical model input				
Argmin	192 (23/18)	$+1.95$ $[-7.75, 11.56]$	199 (6/23)	-6.64 $[-14.40, 1.48]$
Mean as candidate	192 (23/18)	$+1.95$ $[-7.75, 11.56]$	198 (6/24)	-7.03 $[-14.88, 1.19]$
Weighted elites	191 (23/19)	$+1.56$ $[-8.24, 11.29]$	201 (6/21)	-5.86 $[-13.45, 2.05]$
Frozen CEM configurations with raw model input				
$N150/K15$	175 (15/27)	-4.69 $[-14.24, 5.09]$	210 (8/14)	-2.34 $[-9.61, 5.06]$
$N300/K15$	192 (24/19)	$+1.95$ $[-7.95, 11.76]$	227 (16/5)	$+4.30$ $[-2.83, 11.17]$
$N300/K60$	152 (9/44)	-13.67 $[-23.35, -3.35]$	186 (4/34)	-11.72 $[-19.81, -3.02]$
$N600/K60$	204 (28/11)	$+6.64$ $[-2.74, 15.69]$	223 (12/5)	$+2.73$ $[-3.85, 9.14]$
15 rounds	198 (13/2)	$+4.30$ $[-1.70, 10.01]$	219 (6/3)	$+1.17$ $[-3.91, 6.16]$
45 rounds	189 (5/3)	$+0.78$ $[-4.08, 5.57]$	215 (5/6)	-0.39 $[-5.89, 5.14]$

Table 13: TwoRoom checkpoint extension for two additional checkpoints, A and B. The fixed-pool comparison uses 164 mixed queries per checkpoint. Closed-loop rows compare canonical model input with each checkpoint's native/native baseline over 256 episodes; paired-binary intervals use a 12-contrast family. The checkpoints have different weights.

Checkpoint	Raw $\rightarrow$ canonical concordance	Improved / worsened / unchanged
A	$0.7385 \rightarrow 0.8438$	134 / 26 / 4
B	$0.7472 \rightarrow 0.8421$	122 / 37 / 5

Checkpoint / horizon	Native $\rightarrow$ canonical-input success	Δ [adjusted interval] (pp)
A / 25	$204 \rightarrow 200$ (10/14)	-1.56 $[-10.10, 7.08]$
A / 50	$217 \rightarrow 208$ (6/15)	-3.52 $[-11.35, 4.56]$
B / 25	$180 \rightarrow 171$ (11/20)	-3.52 $[-12.92, 6.11]$
B / 50	$200 \rightarrow 183$ (4/21)	-6.64 $[-14.63, 1.79]$

E Additional Related Work

Task-relevant equivalence and hidden state. DeepMDP learns representations through reward and next-latent-state prediction (Gelada et al., 2019), while bisimulation-based learning aligns latent distances with behavioral similarity (Zhang et al., 2021). These objectives connect representation similarity to task dynamics. Our Cube interventions test physically consequential action directions nearly ignored by the first action layer; TwoRoom tests prediction responses to command differences removed by execution-time clipping. Partial observability adds a different issue: Structured World Belief maintains multiple object-state hypotheses when observations are incomplete (Singh et al., 2021). In Reacher, our velocity interventions expose differences hidden from a single image, motivating history or additional state input.

Action fidelity and execution equivalence. ActSWM trains latent rollouts to distinguish alternative-action futures (Gan et al., 2026). WorldSimProbe checks whether supplied actions induce corresponding robot motion and whether predicted environment interactions follow that motion (Co et al., 2026). Our comparisons separately test similarity of model responses, equivalence of candidate execution, and equivalence after elite averaging. Cube shows physical motion differences along directions nearly ignored by action encoding. TwoRoom shows the converse: commands with identical clipped execution produce different predictions, and averaging these commands can break their execution equivalence.

Evaluation units. ACT-Bench measures how faithfully generated driving videos follow supplied action trajectories (Arai et al., 2024). Tseng et al. (2025) evaluate video world models through policy rankings and agreement between predicted and actual policy values. We evaluate preferences between candidates executed from the same initial state, using task success or physical dominance, then follow the selected elite mean into execution. Fixed-pool rescoring reevaluates candidate costs and permits reselection, fixed-member edits change output formation, and fixed-output tests change initial physical state. This separates candidate-ranking evidence from the behavior of the command the planner ultimately executes.