SHIFTWM-BENCH: PLANNING WITH FROZEN LATENT WORLD MODELS UNDER HIDDEN WITHIN-EPISODE DYNAMICS SHIFTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained latent world models are typically evaluated under fixed within-episode dynamics, although changes during execution can invalidate the action–outcome relationships used for planning. We introduce ShiftWM-Bench, a benchmark of four task-specific, state-triggered dynamics changes in manipulation and navigation, with event timing and response parameters hidden from the planner. We study a planning stack with two complementary components and no test-time parameter updates. Proposal Shift combines latent retrieval, cross-entropy method (CEM) initialization, and reliability-weighted action-prior scoring to strengthen candidate search. Latent Belief Recovery infers a distribution over predefined response hypotheses from recent prediction–observation residuals, allocates candidate plans according to this distribution, and ranks them under the most probable hypothesis. The belief model is trained on nominal trajectories and synthetic response perturbations, without real shifted training trajectories. Across 300 scenarios per task and three planner seeds per scenario, Proposal Shift raises four-task mean success from 47.7% with native LeWM to 64.8%, and Recovery further raises it to 72.7%. Controlled post-trigger success gains are established on Cube under particular handoff protocols and control budgets; command-delay gains remain trajectory-dependent, and the current contact-friction approximation can degrade planning. These evaluations measure candidate-search improvements and conditional post-shift benefit separately, characterizing the operating boundary of the complete response-conditioned planner.

1 INTRODUCTION

World models encode visual observations and support planning with predicted action-conditioned futures (Ha & Schmidhuber, 2018; Hafner et al., 2019; Maes et al., 2026). Yet a control mapping can drift after contact, actuator strength can change under load, and commands can acquire delay. A frozen planner must then rank plans using a predictor whose action–outcome assumptions may conflict with recent observations (Figure 1).

We ask: *can interaction history improve a frozen world-model planner after an unannounced within-episode dynamics shift?* A useful plan must first be found and then evaluated under an appropriate action–outcome model. In a state-triggered setting, better search also changes whether the planner reaches the event. Full-episode success and recovery after a shared pre-event trajectory therefore answer different questions.

We address these challenges with two components in one planning loop. PROPOSAL SHIFT uses retrieved actions to initialize cross-entropy method (CEM) search and regularize scoring. LATENT BELIEF RECOVERY uses recent prediction–observation residuals to allocate candidates among predefined response hypotheses and select a common scoring rule. Trained on nominal trajectories and synthetic perturbations, its belief model updates response probabilities during interaction, not model parameters.

We introduce SHIFTWM-BENCH with four task-specific event families: actuator scaling in Reacher, contact friction in PushT, action-mapping rotation in Cube, and command delay in TwoRoom.

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

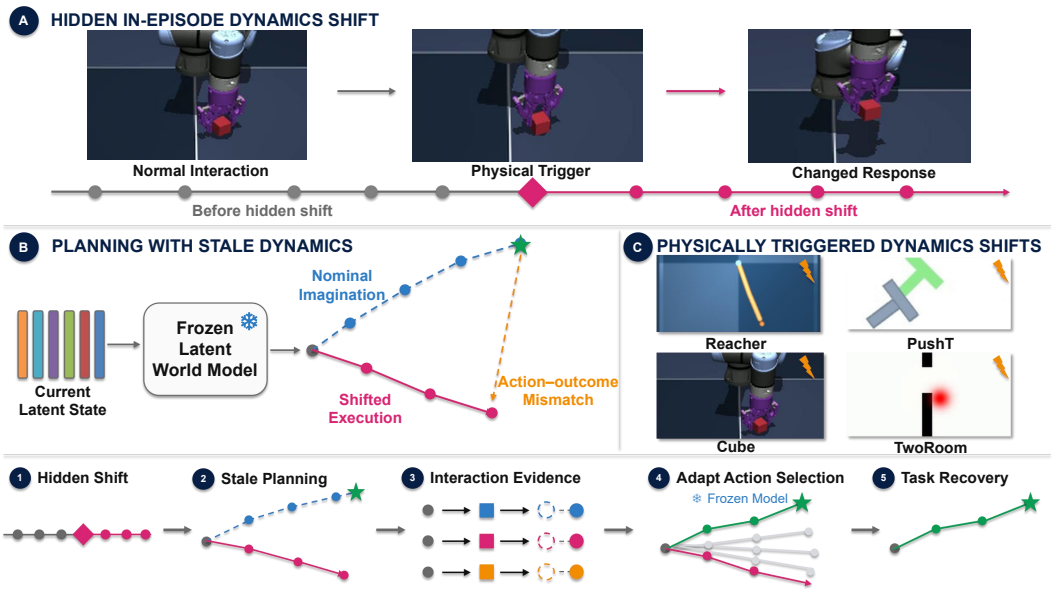

Figure 1: Problem setting of SHIFTWM-BENCH. (A) A hidden event changes dynamics during execution. (B) Nominal predictions can disagree with shifted outcomes. (C) Four task-specific event families. The bottom path illustrates the objective of recovery from interaction with model parameters fixed.

Paired scenarios, matched no-shift runs, and shared-prefix/snapshot controls measure search coverage, closed-loop success, and post-trigger behavior.

Proposal raises four-task success from 47.7% to 64.8%; the complete stack reaches 72.7%. Proposal substantially increases Cube event reachability, while Recovery improves full-episode success on Cube and TwoRoom with both LeWM and Fast LeWorldModel. Giving ordinary Proposal more CEM iterations to cover the same predictor-example allowance leaves a 6.3-point aggregate gap.

Controlled post-trigger success is positive on Cube under particular handoff protocols and budgets; TwoRoom's gain remains trajectory-dependent. Reacher is inconclusive and PushT regresses. These results characterize where the complete response-conditioned interface adds value within the tested task–family pairings.

Our contributions are:

- **Benchmark.** We introduce four task-specific, state-triggered shift families with paired scenarios and matched controls for evaluating event reachability and post-trigger behavior.
- **Planning interface.** We combine retrieval-guided PROPOSAL SHIFT with LATENT BELIEF RECOVERY, using interaction residuals to condition candidate allocation and scoring while keeping model parameters frozen.
- **Empirical findings.** Recovery improves full-episode success on Cube and TwoRoom beyond the tested higher-budget Proposal baseline. Controlled Cube gains depend on handoff protocol and budget; the full-episode pattern also transfers to Fast LeWorldModel.

2 PROBLEM FORMULATION

At physical control step t, the planner receives an image o_t and issues a requested action a_t. A frozen visual encoder e maps the image to a latent feature vector $z_t = e(o_t)$. The simulator has an unobserved state x_t, including pending commands when delays apply. Its transition law $\mathcal{T}_{\xi_t}$ depends on a dynamics parameter ξ_t:

$$x_{t+1} \sim \mathcal{T}_{\xi_t}(x_t, a_t), \qquad \xi_t = \xi_{\text{pre}} \text{ for } t < \tau. \tag{1}$$

Table 1: State-triggered within-episode events in SHIFTWM-BENCH.

Task	Trigger	Hidden change	Physical interpretation
Reacher	accumulated joint displacement	actuator response scale	load or power change
PushT	sustained pusher–object contact	contact-pair friction	surface/contact condition
Cube	stable two-sided grasp	action–mapping rotation	calibration drift
TwoRoom	confirmed door crossing	command delay	control-chain latency

Here $\sim$ denotes a transition drawn from that law, ξ_{pre} is the pre-event parameter, and τ is the first step of a state-triggered event. At τ, the parameter changes abruptly or gradually toward its post-event value ξ_{post}. The response family is predefined, but the planner observes neither these parameters nor the event time or simulator state. A requested command need not equal the action executed after a mapping change or delay.

Given a goal image g, the planner compares candidate plans $A = (a_t, \ldots, a_{t+H-1})$, each containing H requested actions. A *rollout* is the sequence of future latent features predicted for one such plan. We write $F_\theta(z_t, A)$ for this computation, with frozen world-model parameters θ and the backbone's observation context implicit. Lower predicted goal cost indicates a preferred plan. Recovery changes how candidates are generated and scored using completed interaction history; it does not update the encoder or world model. Success within the physical control budget is the evaluation objective.

3 SHIFTWM-BENCH

3.1 DESIGN PRINCIPLES

SHIFTWM-BENCH requires events to occur within an episode, depend on physical state or contact, and remain unannounced to the planner. Construction uses a source trajectory to confirm event reachability with control steps remaining. Evaluation pairs initial states, goals, and planner seeds across methods. These pairs measure a closed-loop policy difference; shared-prefix controls additionally constrain the pre-event actions. A scenario fixes the start, goal, trigger rule, pre/post dynamics, and transition profile; planner seeds repeat that scenario with different search randomness.

The construction split contains 40 scenarios per task: 20 `official-anchor` scenarios using the task's nominal parameter before the event and 20 `cross-regime` scenarios sampled from the registered feasible pre-event family. The formal test contains 150 scenarios from each stratum per task, for 300 scenarios per task and 1,200 total. Scenarios are stratified by applicable event directions, severity, and abrupt versus gradual transition profile. Planner seeds $\{42, 43, 44\}$ produce 900 paired episodes per task and method. Formal source episodes and starts are disjoint from construction, and evaluated-planner success is never used to select scenarios.

3.2 EVENT FAMILIES

Table 1 pairs each task with one predefined event family. This is not a crossed task-by-family design: differences cannot be attributed to shift type independently of task geometry. Event metadata, activation time, and response parameters are withheld from non-privileged planners.

PushT changes friction only at the pusher–object contact pair. Cube includes both rotation directions; TwoRoom confirms arrival in the new room before activation. Each family includes abrupt and short gradual transitions.

Evaluation arms. We compare **Native LeWM** with ordinary CEM; **PROPOSAL SHIFT** ("Proposal" in tables), adding retrieval, prior initialization, and prior-aware scoring; and **PROPOSAL SHIFT + Recovery**, adding response conditioning. The **matched privileged reference** receives the true post-event parameter and applies a fixed correction to finite candidate search. It diagnoses that correction, not an optimal-policy upper bound.

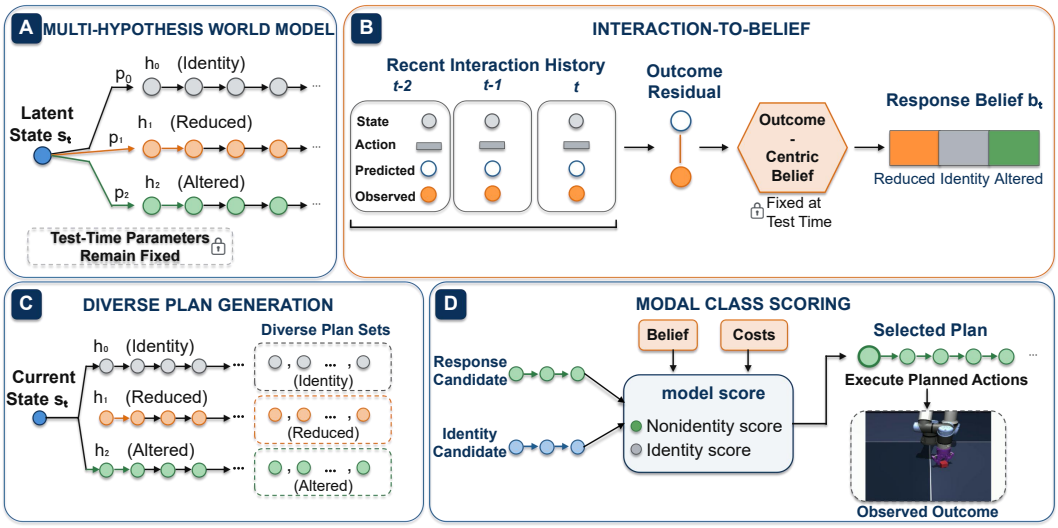

Figure 2: LATENT BELIEF RECOVERY. (A) Operators share one frozen predictor. (B) Beliefs update from five completed action blocks; belief-encoder parameters stay frozen. (C) Beliefs allocate CEM candidates. (D) A common modal score compares the retained sample and all branch means, including zero-allocation branches. Schematic s_t, h_i, and p_i denote latent state z_t, class, and class probability; TwoRoom has four classes. Identity retains the mixture solver.

4 PROPOSAL SHIFT AND LATENT BELIEF RECOVERY

The design separates *which plans to consider* from *how to score them*. PROPOSAL SHIFT supplies retrieval-guided search; Recovery conditions its branches and scores on recent prediction errors (Figure 2). The components operate within one planning loop: at each replanning call, Recovery transforms Proposal's retrieved action reference into response-specific priors, allocates the shared candidate budget, and selects the common score for refinement. History can therefore change both where search concentrates and which plan it prefers. Response operators are planner-side approximations applied to a frozen predictor, not learned replacements for the simulator's transition law.

4.1 PROPOSAL SHIFT: RETRIEVAL-GUIDED SEARCH AND SCORING

PROPOSAL SHIFT builds on retrieval- and prior-guided planning (Gao et al., 2026; Wang et al., 2026), with three operations sharing a retrieved action reference.

Latent retrieval. We concatenate encoded current and goal observations and match this query to a bank of nominal start/end features and 25-action sequences. Nearest-neighbor matching uses standardized features and excludes the query's source episode. The retrieved sequence is an *action prior* for search, not a trajectory to replay; no additional encoder or scoring network is trained.

CEM initialization. The retrieved sequence initializes the search mean when no retained plan is provided. CEM (De Boer et al., 2005) samples plans and refits its Gaussian distribution to the lowest-cost candidates, called *elites*. Including the current mean evaluates the retrieved sequence while allowing search to depart from it.

Reliability-weighted scoring. To retain guidance during refinement, we add mean-squared distance from the retrieved actions, in normalized action coordinates, to the nominal goal cost. Retrieval reliability downweights less similar matches; population normalization expresses the penalty on the task-cost scale (Appendix A.3). Thus the reference guides both initialization and scoring.

Recovery transforms these priors, allocates candidates, and changes response scores using recent interaction. Proposal alone uses nominal predictions without a response belief; *Full* denotes Proposal plus Recovery. Native → Proposal → Recovery compares complete planners.

4.2 OUTCOME-CENTRIC BELIEF

Let $\mathcal{C}$ be a task's fixed set of response classes and $c \in \mathcal{C}$ one class. Reacher and PushT use reduced/identity/amplified latent response; Cube uses negative/identity/positive action rotation; TwoRoom uses four delay bins. Each class specifies a transformation, or *response operator*, for evaluating a changed action–outcome relationship with the frozen predictor. "Identity" leaves that predictor unmodified; it need not describe the actual pre-event dynamics. These hypotheses supply a task-family prior, not simulator labels observed during execution.

To estimate which hypothesis fits recent interaction, we compare the observed endpoint of each completed five-action block with predictions under a fine grid of response values. This grid extracts evidence; coarse classes group responses that share a planning correction. The errors form a log-transformed, normalized vector ϵ_j, where j indexes the latest completed block at step t. The input history h_t concatenates $K = 5$ such vectors and their validity indicators:

$$h_t = [\epsilon_{j-K+1}, \ldots, \epsilon_j,\ m_{j-K+1}, \ldots, m_j], \qquad b_t = q_\phi(h_t). \tag{2}$$

The mask m_j is one for an observed block and zero for a missing, zero-padded block. The classifier q_ϕ, with frozen parameters ϕ, outputs a probability vector b_t; its entry $b_t(c)$ is the probability assigned to class c, and the entries sum to one. This vector is our *response belief*. An ensemble of three multilayer perceptrons averages these probabilities; before any block completes, all mass is on identity. The short history supplies evidence about persistent response differences rather than one prediction error alone.

Response operators and planner costs. Reacher and PushT scale the predicted latent difference between a candidate rollout and a zero-action rollout, approximating weaker or stronger action response. Cube rotates candidate commands to predict their effect and inversely rotates prior commands to compensate. TwoRoom prepends commands still pending under a hypothesized delay. Thus response conditioning changes the inputs or outputs of the predictor, not its weights. The scale approximation is not a physical contact model. Appendix A.1 defines each operator, its response grid, and the corresponding prior transformation.

These operators differ in how directly they represent the intervention. Cube's rotation is explicit in command coordinates. Contact friction changes interactions between bodies: latent scaling need not capture candidate-dependent contact modes or object rotation. A likely response class can therefore still give an unsuitable ranking correction. The response family is specified by the interface, not discovered from unrestricted dynamics.

4.3 BELIEF-CONDITIONED PROPOSAL AND RANKING

At a planning call, each class initializes a CEM *branch*, a separate Gaussian sampling distribution centered on its transformed prior. Let $p_c(A \mid z_t, g)$ denote the current distribution for class c, N the total candidate budget per iteration, and N_c the number assigned to that branch. Belief-weighted allocation approximates the mixture

$$N_c \approx N b_t(c), \qquad p_{\text{mix}}(A \mid z_t, g, b_t) = \sum_{c \in \mathcal{C}} b_t(c) p_c(A \mid z_t, g). \tag{3}$$

Here p_{mix} is the combined proposal distribution. Integer counts are rounded to retain exactly $N = 300$ candidates in total, not per class. Allocation uses the whole belief so that alternatives can receive samples even when one response is most likely. This preserves search directions when short histories support several hypotheses. Ranking instead compares candidates under one shared criterion. Distributing samples therefore does not require averaging costs; the soft control changes scoring while retaining allocation.

All candidates compete under the same response hypothesis, rather than each branch judging its plans under a different response. Let $J_c(A)$ be the predicted goal cost of plan A under class c, and $D_c(A)$ its mean-squared distance to that class's prior in normalized action coordinates. The score is

$$\bar{J}_c(A) = \mathcal{N}[J_c](A) + \lambda \rho_t \mathcal{N}[D_c](A). \tag{4}$$

Here $\lambda = 0.35$ weights the prior penalty, and $\rho_t \in [0, 1]$ measures retrieval reliability, reducing that penalty for less similar retrieved examples. The operator $\mathcal{N}$ centers each cost at its population's 10th

percentile and divides by its 90th–10th percentile spread, with a 10^{-6} denominator floor. Task cost and prior distance are normalized separately so their numerical scales do not determine their relative weight.

The modal class c^* is the class with the largest belief probability. It supplies the common score used in CEM refinement and final plan selection:

$$c^* = \arg\max_{c \in \mathcal{C}} b_t(c), \qquad A_t = \arg\min_{A \in \mathcal{A}_t} \bar{J}_{c^*}(A). \tag{5}$$

The operators $\arg\max$ and $\arg\min$ return the maximizing class and minimizing plan; $\mathcal{A}_t$ contains the final branch means and retained best sampled plan, and A_t is the selected plan. Each branch refits to its own elites under this common score. Allocation retains alternative hypotheses, while ranking makes one shared response assumption. This design does not imply that modal scoring is optimal: Section 5.3 compares belief-weighted scoring in the same solver. The selected 25-action buffer executes before replanning; five-step belief updates do not interrupt it.

4.4 Identity scoring without task routing

Recovery remains enabled on every task in the LeWM, PRISM-prior, IMWM-adapter, and Fast evaluations. When identity is the modal class, it uses the unmodified rollout and prior *within the mixture solver*. This is nominal response scoring, not a switch to the original Proposal policy: the mixture allocation and final branch-mean comparison remain in place. The reported all-task Recovery results use no task-level fallback.

4.5 Training protocol

We transform nominal predictions with sampled response operators and add scaled nominal prediction residuals as noise. The operator supplies the label for cross-entropy training of the belief ensemble. Encoder, world-model, and belief parameters remain fixed during evaluation. The experiments evaluate whether this synthetic supervision supports the complete planner within predefined families, without real shifted training trajectories; Appendix A.1 gives the construction and loss.

5 Experiments

5.1 Protocol

We use released frozen checkpoints, 50 physical steps per episode, 300 CEM candidates per iteration, 30 iterations, 10% elites, and prior weight 0.35. LeWM plans five blocks of five actions; Fast LeWorldModel (Gao & Xu, 2026) uses a 25-action prefix. Both execute 25 actions before replanning; beliefs update every five steps. Candidate/control budgets match, but predictor work and wall time do not: Recovery evaluates five rollouts per candidate for Reacher/PushT, three for Cube, and four for TwoRoom, plus evidence updates. Section 5.3 gives Proposal at least Full's predictor-example allowance per context length; Appendix A.2 verifies accounting and latency.

Each main/transfer arm has 300 scenarios from distinct source episodes per task and seeds $\{42, 43, 44\}$: 900 episodes. Methods share source, start, goal, and seeds. We average seeds within scenario before 20,000 paired bootstrap resamples for 95% confidence intervals (CIs). Macro-4 weights tasks equally and resamples within each task. Intervals are pointwise, without multiplicity adjustment; taskwise/secondary comparisons are exploratory. Supplemental comparisons reuse the fixed benchmark without selecting model parameters, operators, or routes.

5.2 Main result: planning stack under hidden shifts

Table 2 evaluates the complete planning stack under hidden shifts. All four tasks contribute equally to the primary aggregate.

Proposal Shift increases Macro-4 success by 17.1 percentage points; Recovery adds 7.9 points (95% CI $[6.2, 9.5]$). Recovery improves Cube by 22.0 points and TwoRoom by 9.2, while Reacher's $+3.4$-point interval includes zero and PushT decreases by 3.2 points ($[-5.6, -0.9]$). These are full-episode effects of complete planners. Median Recovery solves take 5.2–8.4 seconds across

Table 2: Hidden-shift success (%) on formal-test-v1: 300 scenarios and three planner seeds per task. Macro-4 averages all tasks. Bold marks the best non-privileged method; the privileged reference is diagnostic.

Method	Reacher	PushT	Cube	TwoRoom	Macro-4
Native LeWM	53.2	**80.9**	2.3	54.3	47.7
PROPOSAL SHIFT	50.3	72.8	54.1	82.0	64.8
PROPOSAL SHIFT + Recovery	**53.8**	69.6	**76.1**	**91.2**	**72.7**
Matched privileged reference	58.7	77.1	58.2	90.4	71.1

Table 3: Supplemental fixed-benchmark controls: success (%) over 300 scenarios and three seeds per task. Full is the unchanged modal-scoring method; soft scoring changes its score aggregation, not its solver. Predictor-example allowance is covered, not wall time. Appendix A.2 gives paired intervals, accounting, and reproduction checks.

Planner	Reacher	PushT	Cube	TwoRoom	Macro-4
Proposal	50.3	72.8	54.1	82.0	64.8
Higher-budget Proposal	51.9	73.2	58.4	82.0	66.4
Full (modal)	53.8	69.6	76.1	91.2	72.7
Full (soft)	54.1	70.3	76.6	92.0	73.3

tasks, versus approximately 1.0 second for Proposal; Section 5.3 tests additional ordinary search. Appendix Figure 5 retains the secondary Macro-3 diagnostic.

Event reachability is a separate effect. Cube starts permit a source trajectory to reach a stable grasp with control budget remaining. Native CEM reaches this event in 7.0% of episodes, compared with 79.7% for Proposal and 84.6% for Recovery. The increase from 2.3% Native success to 54.1% with Proposal accompanies much greater event coverage, not recovery alone. These conditional starts differ from the standard static Cube protocol, where a 50-episode local reproduction with the same checkpoint obtains 68%. Shared-prefix controls below separate post-trigger benefit from pre-event policy differences.

5.3 PREDICTION BUDGET AND SCORING CONTROLS

We test whether additional ordinary search closes the Recovery gap. Higher-budget Proposal retains its score and 300-candidate population but uses 151 CEM iterations on Reacher and PushT, 91 on Cube, and 121 on TwoRoom. No-shift workload profiling fixes these counts to cover Full's predictor examples at each input context length, including evidence updates. The four-arm experiment contains 14,400 episodes.

Higher-budget Proposal reaches 66.4% Macro-4, versus 72.7% for Full: a paired $+6.28$-point difference ($[4.47, 8.08]$). Full retains $+17.67$ points on Cube and $+9.22$ on TwoRoom, with intervals above zero; Reacher remains inconclusive and PushT negative. Increasing ordinary CEM iterations does not close the gap. This tests one search-budget allocation, not larger populations, restarts, or equal-wall-time search. Full still takes 1.68–1.76 times longer per solve.

The soft variant averages separately normalized class scores using belief probabilities, retaining allocation, priors, and solver. It is not an expected raw physical cost or a Bayes-optimal objective. It improves Macro-4 over modal scoring by 0.58 points ($[0.11, 1.06]$), an exploratory result. We retain the frozen modal method and report soft scoring as an alternative, not as evidence for modal superiority.

5.4 MATCHED NO-SHIFT BEHAVIOR

Keeping each scenario at its pre-event value yields Macro-4 success of 55.9%, 77.8%, and 79.1% for Native, Proposal, and Recovery. Recovery adds $+1.3$ points ($[-0.1, 2.6]$), compared with $+7.9$ un-

Table 4: Interface evaluation: hidden-shift success (%), 300 scenarios and three seeds per cell. Recovery is active on all four tasks without task fallback. PRISM uses released Cube/PushT heads and nominal-data extensions for Reacher/TwoRoom; IMWM is a paper-spec common-CEM reimplementation. Both retain the LeWM predictor. Fast uses official checkpoints.

Stack	Planner	Reacher	PushT	Cube	TwoRoom	Macro-4
PRISM prior	Common CEM	51.1	**83.1**	23.7	69.3	56.8
	+ Recovery	**56.2**	79.8	**56.9**	**82.6**	**68.9**
IMWM adapter (paper-spec)	Common CEM	51.1	**66.1**	43.2	64.0	56.1
	+ Recovery	**55.4**	62.6	**67.0**	**69.9**	**63.7**
Fast LeWM	Native CEM	**41.6**	**87.0**	3.4	68.9	50.2
	Proposal	41.0	84.9	51.3	84.9	65.5
	+ Recovery	40.7	81.0	**68.7**	**94.4**	**71.2**

der shifts, yielding a paired shifted-minus-static contrast of $+6.6$ ($[4.9, 8.4]$). This compares closed-loop policies, not identical post-trigger states. The suite includes nominal and fixed non-nominal dynamics on conditional starts; Appendix F separately reports the nominal-parameter subset.

5.5 Post-trigger and mechanism controls

A shared-action-prefix control executes source actions through activation before methods plan independently. Cube retains a $+4.6$-point Recovery gain ($[1.6, 7.7]$), also positive after excluding simulator replay mismatches (Appendix H). A separate identical-snapshot handoff yields $+2.0$ points ($[-1.7, 5.7]$) at the same 50-step total budget, with positive success effects at 60/70 steps (Appendix A.4). Cube's controlled success evidence therefore depends on handoff protocol and budget. TwoRoom's shared-prefix difference is -0.6 ($[-2.3, 1.2]$); its snapshot result and Reacher's are also inconclusive.

A residual cumulative-sum (CUSUM) adapter, based on cumulative change detection, reaches 68.6% Macro-4, below Recovery by 4.0 points ($[2.5, 5.6]$). It shares the predictor, retrieval backbone, and candidate budget but uses a fixed detector and single-response planning, making this a complete-adapter comparison (Appendix F).

Mechanism diagnostics. Six construction variants use 40 scenarios and three seeds (Appendix A.3). Cube Full reaches 70.0%, versus 58.3% for belief allocation with nominal scoring and 60.8% for nominal proposals with modal scoring; both paired differences are positive. Holding the mixture solver fixed, Full also exceeds fixed identity by 9.2 points on Cube and 6.7 on TwoRoom, with intervals above zero. These controls support response conditioning within this solver. They do not give additive allocation/ranking effects: the nominal-score variant also changes prior normalization, and nominal-proposal planning uses a different CEM solver. PushT instead favors nominal-scored allocation. The soft/modal comparison in Section 5.3 isolates score aggregation within the Full solver.

Cube belief agreement is 93.2% at 44 eligible abrupt-shift endpoints from 16 scenarios. This selected-endpoint diagnostic is distinct from decision value: Full minus randomized beliefs is $+4.2$ points ($[-2.5, 11.7]$). Correctly aligned inference is therefore not independently established as the cause of the planning gain.

5.6 Interface compatibility and predictor transfer

Table 4 tests PRISM-prior and paper-spec IMWM interfaces with the LeWM predictor, and architectural transfer to official Fast LeWorldModel's parallel predictor with a separate retrieval cache and belief encoder.

Recovery raises PRISM's Macro-4 from 56.8% to 68.9% and the IMWM adapter's from 56.1% to 63.7% ($+7.6$ points, $[5.4, 9.9]$). IMWM's Cube and TwoRoom differences have intervals above zero; Reacher is inconclusive, and PushT decreases by 3.6 points ($[-8.0, 0.9]$; Appendix G).

Fast Recovery raises Proposal's 65.5% to 71.2%, a paired $+5.7$ points ($[4.1, 7.3]$). Cube gains 17.3 points and TwoRoom 9.6, with intervals above zero; Reacher is inconclusive and PushT loses 3.9 ($[-6.1, -1.8]$). This demonstrates full-episode transfer of the interface to the tested Fast architecture; shared-prefix evidence remains specific to LeWM.

Fast matched no-shift success is 56.9%, 76.8%, and 76.6% for Native, Proposal, and Recovery. The static increment is -0.25 points ($[-1.44, 0.94]$); the shifted-minus-static contrast is $+5.92$ ($[4.53, 7.33]$). PushT and TwoRoom incur static losses (Appendix G).

6 Related Work

Latent world models and planning. World models support visual control through latent prediction and imagination (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2020; Hansen et al., 2024). DINO-WM uses pretrained visual features; LeWM learns them end to end (Zhou et al., 2025; Maes et al., 2026). We study planning with frozen predictors.

Planner proposal and ranking. PETS combines predictive uncertainty with trajectory sampling (Chua et al., 2018); iCEM improves finite-budget search (Pinneri et al., 2021). IMWM uses intuition, retrieval, and hybrid costs; PRISM guides sampling with an action prior (Gao et al., 2026; Wang et al., 2026). Proposal Shift combines retrieval initialization with reliability-weighted prior scoring.

Planning with frozen predictors. Hi-LeWM studies latent subgoals (Caselli et al., 2026); other work modifies planning objectives (Singh, 2026; Pham & Bera, 2026). Concurrent Anchored Planning retrieves intermediate observations and ranks retrieved actions (Liu et al., 2026). Our retrieval instead initializes and regularizes CEM while retaining the final goal. Recovery conditions this search on recent response mismatch, evaluated under hidden state-triggered changes.

Dynamics adaptation. Dynamics augmentation and online adaptation address changing responses (Ball et al., 2021; Yu et al., 2017; Nagabandi et al., 2019; Kumar et al., 2021; Zhang et al., 2024). PEARL and VariBAD infer latent context (Rakelly et al., 2019; Zintgraf et al., 2020). SeCBAD jointly infers context and segment length; DynaMITE-RL models within-episode sessions (Chen et al., 2022; Liang et al., 2024). Unlike their trained context-conditioned policies, our synthetic-trained belief conditions finite-budget planning with a frozen predictor and predefined response operators.

7 Limitations and Future Work

Each task uses one shift family, confounding geometry and shift type; severity slices do not establish held-out generalization. Controlled Cube success depends on handoff protocol and budget. TwoRoom's gain is full-episode, Reacher is inconclusive, and PushT regresses. Predictor-budget and component controls establish neither equal-latency efficiency nor correct classification as the cause of gains. Crossed task/family tests, response holdouts, and matched-solver interventions are natural next steps. Simulator-specific operators provide no real-robot safety guarantee.

8 Conclusion

ShiftWM-Bench measures search coverage, full-episode success, and controlled post-trigger behavior for frozen visual planners. Proposal Shift strengthens search; Latent Belief Recovery conditions allocation and scoring on interaction. Full-episode Cube and TwoRoom gains and protocol-dependent controlled Cube gains support the interface. Stronger search changes event reachability; controlled recovery evaluates decisions after a shared handoff. More ordinary CEM iterations do not close the aggregate gap, and full-episode gains extend to Fast LeWorldModel. These findings motivate evaluating candidate search and response adaptation together, with separate controls for their contributions.

AI USE STATEMENT

We used AI tools to assist with experimental design, implementation and debugging, and manuscript drafting and polishing. Numerical results are computed from simulator evaluations and episode records rather than generated by a language model. The authors are responsible for the final methods, claims, artifacts, and verification.

REPRODUCIBILITY STATEMENT

Appendices specify operators, training, planners, protocols, and uncertainty estimates. The archive supplies manifests, source hashes, model identifiers, commands, and episode records for reproducing the comparisons.

REFERENCES

Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. In *International Conference on Machine Learning*, pp. 619–629, 2021.

Niccolò Caselli, Francesco Massafra, Samuele Punzo, Salvatore Lo Sardo, Ippokratis Pantelidis, and Sathya Kamesh Bhethanabhotla. Mind the gap: Promises and pitfalls of hierarchical planning in LeWorldModel. *arXiv preprint arXiv:2607.12547*, 2026.

Xiaoyu Chen, Xiangming Zhu, Yufeng Zheng, Pushi Zhang, Li Zhao, Wenxue Cheng, Peng Cheng, Yongqiang Xiong, Tao Qin, Jianyu Chen, et al. An adaptive deep rl method for non-stationary environments with piecewise stable context. *Advances in Neural Information Processing Systems*, 35:35449–35461, 2022.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

Baoqi Gao, Ruize Han, Miao Wang, and Song Wang. IMWM: Intuition models complement world models for latent planning. *arXiv preprint arXiv:2606.01626*, 2026.

Yuntian Gao and Xiangyu Xu. Fast LeWorldModel. *arXiv preprint arXiv:2606.26217*, 2026.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination, 2020.

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

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems*, 2021.

Anthony Liang, Guy Tennenholtz, Chih-Wei Hsu, Yinlam Chow, Erdem Biyik, and Craig Boutilier. Dynamite-rl: A dynamic model for improved temporal meta-reinforcement learning. *Advances in Neural Information Processing Systems*, 37:141390–141416, 2024.

Xvyuan Liu, Jianjie Fang, Chen Gao, and Yong Li. Aim short to reach far: Your frozen world model can plan better than you think. *arXiv preprint arXiv:2609.30036*, 2026.

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

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019.

Phu Pham and Aniket Bera. Latent energy action planning with world models. *arXiv preprint arXiv:2609.03294*, 2026.

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*, pp. 1049–1065, 2021.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning*, 2019.

Joyjeet Singh. The objective is the bottleneck: Latent world models encode what their planners cannot use. *arXiv preprint arXiv:2608.12959*, 2026.

Yuhai Wang, Jiawei Xia, Rongxuan Zhou, Xiao Hu, Yongliang Shi, Jing Du, and Yang Ye. PRISM: Prior-guided imagination sampling in world models. *arXiv preprint arXiv:2606.07974*, 2026.

Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Robotics: Science and Systems*, 2017.

Xinyu Zhang, Wenjie Qiu, Yi-Chen Li, Lei Yuan, Chengxing Jia, Zongzhang Zhang, and Yang Yu. Debiased offline representation learning for fast online adaptation in non-stationary dynamics. In *International Conference on Machine Learning*, 2024.

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

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*, 2020.

A Additional Experimental Details

This appendix specifies the response operators, training procedure, and control loop used in the main experiments. The LeWM Recovery arm uses a frozen three-model belief ensemble and the modal-cost mixture solver. Throughout, t indexes physical control steps, j completed five-action blocks, r fine-grid response values, and c coarse response classes. These indices refer to different objects: the fine grid constructs evidence, while the coarse classes define the planning branches.

Scenario and resampling units. The released manifests contain 300 distinct source episodes per task in formal-test-v1 and 40 per task in construction, with no source-episode overlap between the two splits. No formal source episode supplies multiple scenarios within a task, including across the two strata. Thus source-episode grouping and scenario grouping coincide here; the three planner seeds are repeated evaluations within that unit, not independent scenarios. This manifest check does not imply that the source dataset consists of independent draws from a broader deployment distribution. The reported intervals quantify paired variation across these benchmark scenarios and are not simultaneous confidence statements across all comparisons.

A.1 RESPONSE OPERATORS, TRAINING, AND EXECUTION

Response grids and labels. Reacher and PushT use the fine scale grid

$$\{0.25, 0.40, 0.60, 0.80, 1.00, 1.20, 1.40, 1.60, 1.80\}.$$

Values below, equal to, and above one map to reduced, identity, and amplified classes, with planning representatives $[0.6, 1.0, 1.4]$. Cube uses $[-60^\circ, 60^\circ]$ in 5° increments; negative, zero, and positive values map to representatives $[-45^\circ, 0^\circ, 45^\circ]$. TwoRoom uses integer delays $0, \ldots, 12$, grouped as 0, 1–4, 5–8, and 9–12, with representatives $[0, 2, 6, 10]$ physical control steps. The fine grid supplies residual features and synthetic labels; the coarse representatives define planning branches.

Response-conditioned task cost. We omit the current time subscript and write $z = z_t$ for the encoded observation context. The plan A contains $H = 25$ physical requested actions. $F_\theta(z, A)$ returns their predicted latent sequence under the frozen backbone, and $\mathcal{L}_g$ maps that sequence to a scalar goal cost. For the LeWM implementation,

$$\mathcal{L}_g(Z) = \|Z_{\text{end}} - e(g)\|_2^2, \qquad J_{\text{id}}(A) = \mathcal{L}_g(F_\theta(z, A)). \tag{6}$$

Here Z is a predicted latent sequence, Z_{end} its terminal feature vector, $e(g)$ the encoded goal image, and $\|\cdot\|_2^2$ the sum of squared feature-coordinate differences. The subscript id denotes the identity response. Transfer adapters retain their backbone's goal criterion.

For a dimensionless response scale κ, the operator and its cost are

$$F_\kappa(z, A) = F_\theta(z, 0) + \kappa\{F_\theta(z, A) - F_\theta(z, 0)\}, \qquad J_\kappa(A) = \mathcal{L}_g(F_\kappa(z, A)). \tag{7}$$

The zero in $F_\theta(z, 0)$ denotes an all-zero *physical* action sequence of the same length as A, not zeros in normalized action coordinates. F_κ is a transformed rollout, not a newly trained predictor. Subtracting the zero-action rollout supplies an approximate action-dependent displacement to scale; it is not a causal decomposition of latent dynamics. The transformation applies to the predicted sequence before the goal cost. $\kappa = 1$ recovers the nominal prediction, values below one reduce the displacement, and values above one amplify it.

For Cube, the signed angle α defines a planar rotation R_α:

$$R_\alpha(a_x, a_y) = (\cos\alpha\, a_x - \sin\alpha\, a_y,\ \sin\alpha\, a_x + \cos\alpha\, a_y), \qquad J_\alpha(A) = \mathcal{L}_g(F_\theta(z, R_\alpha A)). \tag{8}$$

Here a_x, a_y are the first two raw action coordinates; the remaining coordinates are unchanged. $R_\alpha A$ applies the same rotation to every action in A. Grid angles are specified in degrees and converted to radians for the trigonometric functions. Forward prediction uses $+\alpha$ because it models the environment's action mapping; compensating a desired command uses the inverse rotation $-\alpha$.

For TwoRoom, d is a hypothesized nonnegative integer delay in physical steps, and $Q_d(A)$ is the command sequence predicted to execute over the next H steps:

$$J_d(A) = \mathcal{L}_g(F_\theta(z, Q_d(A))). \tag{9}$$

The operator Q_d prepends the last d previously requested commands, in execution order, then appends A and truncates the result to H actions. Unavailable history entries are zero-padded. Thus the planner reasons about the pending queue using its own command history, without reading the simulator's queue or event state.

Class-conditioned priors. Let U be the retrieved reference sequence in raw action coordinates. Reacher initializes branch κ from U/κ, increasing the requested magnitude when the hypothesized response is weak. Cube initializes branch α from $R_{-\alpha}U$. TwoRoom advances the desired sequence by d positions and repeats its last action to fill the tail, anticipating command delay. These transformed priors are clipped to the environment action bounds and mapped back to the backbone's normalized action coordinates. PushT uses the same retrieved prior in every branch: its response dependence enters the cost and branch allocation, not a friction-specific inverse action map.

Denote the resulting normalized prior for class c by U_c. The penalty $D_c(A)$ in Eq. 4 is the mean of squared coordinate differences between the normalized candidate sequence and U_c, averaged over all H actions and action coordinates. The block packing used by LeWM does not change this average. The reliability ρ_t is supplied by the frozen retrieval procedure from the similarity of the current/goal query to stored examples; it is separate from the response probabilities $b_t(c)$.

Population score normalization. For a vector v containing one scalar cost per candidate in the current population, let $\mathrm{Quant}_p(v)$ be its empirical quantile at fraction p. The normalization used separately for task costs and prior distances is

$$\mathcal{N}[v] = \frac{v - \mathrm{Quant}_{0.1}(v)}{\max\{\mathrm{Quant}_{0.9}(v) - \mathrm{Quant}_{0.1}(v), 10^{-6}\}}. \tag{10}$$

Subtraction and division act on every entry of v. The notation $\mathcal{N}[J_c](A)$ in the main text means the entry corresponding to plan A after normalizing the vector of class-c task costs; $\mathcal{N}[D_c](A)$ is defined analogously. The denominator floor prevents division by a near-zero spread. This affine transformation neither clips values nor replaces them with ranks.

Residual features. For completed block j and fine-grid index r, let $e_{j,r}$ be the mean squared difference between its predicted and observed latent endpoints, averaged over feature coordinates. If the grid has G values, then $r = 1, \ldots, G$. We compute

$$\ell_{j,r} = \log(e_{j,r} + 10^{-8}), \qquad u_{j,r} = \ell_{j,r} - \min_{1 \le k \le G} \ell_{j,k}, \tag{11}$$

$$\epsilon_{j,r} = \frac{u_{j,r}}{\max\{\mathrm{median}\{u_{j,k} : u_{j,k} > 10^{-10}\}, 10^{-4}\}}. \tag{12}$$

Here $\ell_{j,r}$ is the natural-log error, $u_{j,r}$ subtracts the best grid value's log error within the same block, and $\epsilon_{j,r}$ is the normalized feature. The dummy index k ranges over the same fine grid as r. The constant 10^{-8} makes the logarithm finite; 10^{-10} excludes effectively zero entries from the median, and 10^{-4} floors the normalization scale. An empty positive set has median one. These operations represent relative support across hypotheses rather than the absolute magnitude of one prediction error.

The vector $\epsilon_j = (\epsilon_{j,1}, \ldots, \epsilon_{j,G})$ is the block feature used in the main text. The last $K = 5$ vectors are left-padded with zeros, flattened, and concatenated with the K binary validity masks m_j. Thus the input dimension is $KG + K$: 50 for Reacher/PushT ($G = 9$), 130 for Cube ($G = 25$), and 70 for TwoRoom ($G = 13$). Each encoder is a multilayer perceptron (MLP) with two hidden layers of width 128 and rectified linear unit (ReLU) activations. A softmax converts each model's output scores to class probabilities; three independently seeded models supply their arithmetic mean.

Synthetic supervision. Each nominal evidence episode supplies fine-grid endpoint predictions and the residual between the observed endpoint and the identity prediction. For each of 12 augmentations, we sample a nonidentity grid value and a block at which it becomes active. The synthetic label is identity before that block and the sampled response afterward.

For synthetic training window i, let $\hat{z}_{i,r}$ be its latest block's endpoint prediction under grid value r, let r_i^* be the active grid index, and let η_i be the observed-minus-identity residual from the corresponding nominal block. The synthetic endpoint is

$$z_i^{\mathrm{syn}} = \hat{z}_{i,r_i^*} + \nu_i \eta_i, \qquad \nu_i \sim \mathrm{Uniform}[0.75, 1.25]. \tag{13}$$

The hat distinguishes a prediction from an observation, the superscript syn denotes synthetic data, and the scalar ν_i randomly rescales the nominal residual. Residual features are recomputed against *all* grid predictions using this endpoint. Their history forms h_i, and the active grid value maps to coarse label c_i. Each ensemble member independently minimizes

$$\mathcal{L}_{\mathrm{belief}}^{(s)} = -\frac{1}{M} \sum_{i=1}^{M} \log q_{\phi_s}(c_i \mid h_i). \tag{14}$$

Here M is the number of training windows, i their index, $s \in \{1, 2, 3\}$ the ensemble-member index, and ϕ_s that member's parameters. $q_{\phi_s}(c_i \mid h_i)$ is its probability for the target class; the negative natural log is the cross-entropy loss. The main-text q_ϕ denotes the averaged prediction of these three members, with ϕ collecting their frozen parameters.

Training uses AdamW, learning rate 10^{-3}, weight decay 10^{-4}, batches of 256, and 80 epochs. Each task's archived fit uses 40 nominal evidence episodes and reserves 20 nominal episodes for calibration; these are episode records, not 60 distinct benchmark scenarios. The generated training-window counts are 2,484 (Reacher), 2,772 (PushT), 2,820 (Cube), and 1,932 (TwoRoom). The main mixture arm selects the largest class probability directly; inherited confidence-streak thresholds do not gate its class selection. The same residual representation is used on every task.

Mixture CEM. Branch means start at their class priors, with unit standard deviation in normalized action coordinates. Largest-remainder rounding first takes the integer part of each $Nb_t(c)$ and then distributes the remaining slots in descending order of the fractional remainders. This gives exactly $N = 300$ candidates at every iteration. Each nonempty branch includes its mean as one sample and updates from its lowest-cost $\lceil 0.1N_c \rceil$ local elites, where $\lceil \cdot \rceil$ rounds up to the next integer and N_c is that branch's sample count. Standard deviations have floor 10^{-4}; a single-elite branch halves its previous standard deviation. After 30 iterations, a final comparison rescores all class means, including zero-allocation branches, and the retained sampled candidate. The retained sample is tracked using its iteration-normalized score and rescored in this final population. Full therefore differs from nominal Proposal in solver behavior as well as response context.

Control cadence and computation. LeWM plans five latent blocks, each containing five physical actions. Fast plans one 25-action prefix. Both execute 25 actions before the next CEM call in the ordinary 50-step evaluation. Belief features and retrieval context update every five steps, but a new belief does not replace actions already buffered. Shared-prefix protocols can initialize a new planner at handoff. Each CEM call samples $30 \times 300 = 9{,}000$ candidates, followed by the small final comparison in the mixture solver. The implementation evaluates every coarse-class cost before choosing the modal score: Reacher/PushT require five rollout evaluations per candidate (two for each nonidentity scale plus identity), Cube three, and TwoRoom four. These counts exclude belief-feature prediction, encoding, retrieval, and final-candidate evaluation; they count rollout evaluations, not GPU kernels. The comparison therefore matches search populations and control budgets while allowing additional computation for response conditioning.

Table 5: Execution order of the evaluated Recovery policy. Belief updates are causal and can occur while an existing plan is executing.

Inputs: frozen encoder/predictor and belief ensemble, retrieval cache, response operators, goal.
Initialize identity belief, empty history, and empty action buffer.
At each five-step boundary, encode the observation; if a block has completed, append its response-error vector and update the belief. Refresh the retrieved prior.
If the action buffer is empty, form class priors, allocate 300 candidates across branches, and run 30 CEM iterations using the modal response score. Select the final plan and fill the 25-action buffer.
Take the next buffered action. At a five-step boundary, predict the next five-action block under the fine response grid for the next residual update.
Execute the action and append the requested command to history. Repeat until success or budget exhaustion.

Statistical unit. Confidence intervals first average the three planner seeds within each scenario and then use 20,000 paired scenario-bootstrap resamples. Macro intervals resample scenarios independently within each task before taking the unweighted task mean. Intervals are pointwise; no multiplicity adjustment is applied. Supplemental comparisons reuse the fixed splits and do not select new model parameters, thresholds, or routes from formal outcomes.

A.2 Prediction-budget and score-aggregation controls

Question and fixed protocol. The supplemental experiment tests two alternatives: whether more ordinary Proposal search closes the gap to Full, and whether selecting the modal response score is preferable to averaging class scores. It reuses all 300 formal scenarios per task with seeds $\{42, 43, 44\}$, for 900 episodes per task/arm and 14,400 total. The four arms are original Proposal, higher-budget Proposal, Full/modal, and Full/soft. The model, belief ensemble, retrieval cache, response operators, prior weight 0.35, 300-candidate population, 10% elite fraction, 25-action execution buffer, and 50-step environment budget are unchanged. The new comparisons were specified before their measurements on an already observed fixed benchmark; they are not a newly blinded holdout.

Soft score. The soft arm changes the score aggregation in the Full mixture solver to

$$\bar{J}_{\mathrm{soft}}(A) = \sum_{c \in \mathcal{C}} b_t(c) \bar{J}_c(A), \tag{15}$$

where A is a candidate plan, $\mathcal{C}$ the registered class set, $b_t(c)$ the current class probability, and $\bar{J}_c$ the individually normalized task-plus-prior score in Eq. 4. The subscript soft identifies this alternative. Allocation, class priors, normalization, final candidate comparison, and rollout evaluation are unchanged. This is an average of normalized class scores, not an expected raw physical cost. One-hot beliefs produce identical hard and soft scores; semantic tests check that equivalence and the weighted-sum implementation. Different scores can subsequently change CEM refinements, executed actions, and observed histories.

Prediction-work accounting. We count examples passed to the frozen predictor separately for context lengths $\ell \in \{1, 2, 3\}$, measured in latent time positions. A batched call with B examples contributes B at its context length, whereas the number of calls counts the batched invocations themselves. This distinction matters because the mixture solver also evaluates small final populations.

Four fixed matched-no-shift construction scenarios per task (two per stratum, seed 42) determine the iteration allowance without success-based selection. Let w_ℓ be the examples used by one original Proposal iteration, R_ℓ the largest observed Full solve count including the final comparison, and E_ℓ the largest outside-solve prediction count in one policy action. Five evidence boundaries fit in a 25-action buffer. The higher-budget iteration count is

$$I_{\mathrm{high}} = \max\left\{30,\ \max_{\ell\in\{1,2,3\}} \left\lceil \frac{R_\ell + 5E_\ell}{w_\ell} \right\rceil \right\}. \tag{16}$$

Here I_{high} is the integer CEM iteration count and $\lceil\cdot\rceil$ rounds upward. The outer maximum retains at least the original 30 iterations. Proposal has no predictor work outside these iterations in its solve. In length order $(1, 2, 3)$, w_ℓ is $(300, 300, 900)$; the rule yields 151 iterations for Reacher/PushT, 91 for Cube, and 121 for TwoRoom. Population size and the nominal scoring rule remain fixed.

Table 6: Realized predictor-example allowance per 25-action buffer. Triples are counts at context lengths $(1, 2, 3)$. Full's requirement conservatively combines its largest solve and five largest evidence-boundary counts; higher-budget Proposal covers it at every length. This excludes encoding and is not FLOP, memory, or wall-time equality.

Task	Iterations	Full requirement	Higher-budget allowance
Reacher	151	$(45030, 45020, 135060)$	$(45300, 45300, 135900)$
PushT	151	$(45030, 45020, 135060)$	$(45300, 45300, 135900)$
Cube	91	$(27137, 27012, 81036)$	$(27300, 27300, 81900)$
TwoRoom	121	$(36085, 36020, 108060)$	$(36300, 36300, 108900)$

The allowance check is repeated on the complete evaluation records, comparing the smallest higher-budget allowance with the largest Full requirement at every context length (Table 6). All four tasks pass. The unit is predictor examples, not distinct candidate plans or GPU kernels: the baseline spends its allowance on more ordinary search iterations, whereas Full spends most of its work on multiple responses.

Paired outcomes. Table 7 gives the prespecified primary Macro-4 comparison, Full minus higher-budget Proposal, and the secondary Full/modal minus Full/soft comparison. We average seeds within each scenario and use 20,000 paired scenario-bootstrap draws, with independent within-task resampling for Macro-4 and random-number seed 20260923. Taskwise and secondary intervals are exploratory pointwise 95% intervals without multiplicity adjustment. All rates and differences are computed before rounding.

Increasing ordinary search improves Cube, but Full retains a larger success rate on Cube and TwoRoom. Reacher remains inconclusive and PushT remains negative. Soft scoring has a small aggregate advantage over modal scoring; no evaluated task shows a clear modal advantage. These findings support the complete interface relative to the tested search-budget alternative, not the superiority of hard response selection or an isolated benefit from correctly identifying the response class. All comparisons are full-episode and do not add shared-state post-trigger evidence.

Hardware and timing. Workers use one NVIDIA GeForce RTX 4090 (24 GB) each on an Intel Xeon Gold 6330 host, with Python 3.10.20, PyTorch 2.5.1, and CUDA 12.1. GPU process occu-

Table 7: Supplemental success differences in percentage points with paired 95% CIs. High denotes higher-budget Proposal; Full denotes the original modal method. The primary supplemental endpoint is Macro-4 Full−High. Full−Soft retains the prespecified subtraction direction.

Task	Full−High	High−Proposal	Full−Soft
Reacher	$+1.89$ $[-1.67, 5.44]$	$+1.56$ $[-0.78, 4.00]$	-0.33 $[-1.44, 0.89]$
PushT	-3.67 $[-7.00, -0.33]$	$+0.44$ $[-2.22, 3.11]$	-0.78 $[-1.78, 0.22]$
Cube	$+17.67$ $[13.11, 22.22]$	$+4.33$ $[1.22, 7.44]$	-0.44 $[-1.33, 0.44]$
TwoRoom	$+9.22$ $[6.33, 12.22]$	0.00 $[-2.56, 2.56]$	-0.78 $[-1.56, -0.11]$
Macro-4	$+6.28$ $[4.47, 8.08]$	$+1.58$ $[0.25, 2.92]$	-0.58 $[-1.06, -0.11]$

Table 8: Supplemental mean task error at episode termination; lower is better. High is higher-budget Proposal. Differences are Full/modal minus High with paired 95% CIs. Reacher uses radians, Cube simulator length units, and PushT/TwoRoom environment coordinate units, as in Appendix A.4.

Task	High	Full/modal	Full/soft	Full−High [95% CI]
Reacher	0.1018	0.0991	0.0991	-0.0027 $[-0.0077, 0.0021]$
PushT	19.5404	22.0643	21.6633	$+2.5240$ $[0.7917, 4.3426]$
Cube	0.0763	0.0683	0.0676	-0.0080 $[-0.0128, -0.0031]$
TwoRoom	19.3747	17.0969	16.9286	-2.2778 $[-3.4661, -1.0933]$

pancy is sampled every ten seconds; accepted profiles contain no recorded competing process on the assigned GPU, although shorter overlap is not ruled out. Timers synchronize CUDA at solve and policy-call boundaries. The first episode of each process segment is excluded from latency summaries, but retained for success and work counts. Table 9 pools the remaining full-evaluation calls.

Table 9: Measured supplemental computation. Solve times are median and 95th percentile in seconds. Calls/examples are median predictor invocations/batched examples per solve, excluding evidence outside the solve. Memory is peak PyTorch-allocated MiB, not total process/device memory. P, High, Full, and Soft denote Proposal, higher-budget Proposal, Full/modal, and Full/soft.

Task	Arm	Median	P95	Calls	Examples	Memory
Reacher	P	0.985	1.002	150	45000	454.2
	High	5.003	5.117	755	226500	454.2
	Full	8.418	8.636	775	225100	457.7
	Soft	8.374	8.736	775	225100	457.7
PushT	P	0.990	1.013	150	45000	469.7
	High	4.824	4.963	755	226500	469.7
	Full	8.278	8.389	775	225100	472.2
	Soft	8.379	8.496	775	225100	471.4
Cube	P	0.987	1.020	150	45000	454.2
	High	2.951	3.065	455	136500	454.2
	Full	5.185	5.405	465	135060	456.1
	Soft	5.262	5.384	465	135060	456.5
TwoRoom	P	0.980	0.995	150	45000	253.0
	High	3.915	4.048	605	181500	253.0
	Full	6.798	6.844	620	180100	254.9
	Soft	6.754	6.874	620	180100	254.6

Full's median solve latency is 1.68–1.76 times the higher-budget baseline despite the latter's predictor-example allowance. Python/solver work, batched-call structure, and scoring overhead are not equalized; the experiment is not a wall-time or FLOP match. Mean policy-call times, including

buffered calls, retrieval, and evidence updates, are 0.233/0.388 seconds for High/Full on Reacher, 0.214/0.369 on PushT, 0.132/0.236 on Cube, and 0.200/0.359 on TwoRoom. These workload means depend on policy trajectories and early termination. Retained batch timers sum to 44.66 worker-hours across the 16 arms, not elapsed makespan or total project GPU time; profiling, initialization, and unrecorded failed-batch work are not included. Full timing distributions, reserved-memory peaks, and process-segment records accompany the analysis.

Data verification and reproduction. All 14,400 records are present and pass scenario/seed pairing, source/start, initial/goal, dynamics-schedule, and recorded leakage checks. The 111 recorded source/artifact integrity checks pass, and independent recomputation agrees with every arm's success rate and planned paired success interval. Resumed jobs preserve their completed prefixes exactly; no failed episode or unfavorable outcome is dropped.

All 7,200 repeated Proposal/Full episodes reproduce the archived success outcomes. Six Cube trajectories differ in requested/executed action hashes, with a maximum absolute final-error difference of approximately 2.77×10^{-5} simulator length units and unchanged success outcomes. Both archived and repeated records are retained.

A.3 Planner components and belief diagnostics

Six planner variants compare the complete wrapper with nominal scoring, nominal proposal initialization, and fixed response context. The protocol uses `construction_dev_v0.3_fixed`: 20 official-anchor and 20 cross-regime scenarios per task, each with seeds $\{42, 43, 44\}$. This gives 120 episodes per arm and 2,880 across four tasks. All arms use a 50-step environment budget, five blocks of five actions, 300 CEM samples, 30 iterations, a 10% elite fraction, and prior weight 0.35. The sampling budget is shared; solver details are specified below. The comparison protocol was fixed before these runs on previously used construction scenarios. The results serve as closed-loop component diagnostics, with no model or task-route selection from their outcomes.

Compared interventions. **Proposal** is the locked retrieval/prior-aware backbone. **Alloc./nom.** retains belief-weighted branch sampling but scores every candidate with the nominal world-model cost and nominal identity-prior penalty, including retrieval reliability. **Nom./modal** retains nominal retrieval-initialized CEM and uses modal response scoring; scoring also changes subsequent CEM refinement. **Fixed identity** uses the Full mixture solver with a one-hot identity posterior at every replanning boundary. **Full** is the complete Recovery planner in Section 4. **Oracle context** substitutes a one-hot registered response class obtained from privileged event context, retaining the Full solver and configured budgets.

Fixed identity retains the mixture solver and score normalization; its final comparison includes every branch mean, even branches receiving zero sampled candidates. It therefore tests nominal response context within the Full implementation rather than reproducing Proposal. Oracle context selects identity before activation and maps the current response to a registered representative afterward. Cube uses the sign of the angle change, Reacher the actuator ratio, and TwoRoom the current delay bin. PushT uses friction direction as an unvalidated response-scale proxy. Oracle–Full consequently changes event access, class choice, and posterior uncertainty together; it evaluates privileged context in this solver, not an optimal policy.

Table 10: Six construction-only planner comparisons: success (%) over 40 scenarios and three seeds per arm. Alloc./nom. retains the nominal prior-aware score; Nom./modal uses nominal proposal initialization and modal response scoring. Oracle context is privileged and is not a deployable competitor.

Task	Proposal	Alloc./nom.	Nom./modal	Fixed identity	Full	Oracle
Reacher	56.7	61.7	58.3	55.0	57.5	56.7
PushT	66.7	77.5	67.5	63.3	66.7	68.3
Cube	55.0	58.3	60.8	60.8	70.0	76.7
TwoRoom	82.5	81.7	85.8	83.3	90.0	91.7

Pairing and reproducibility. All six arms contain the same 40 scenario identifiers and all three seeds. Paired metadata checks cover pre/post dynamics, profile, source episode/start, direction, severity, and subset; recorded initial/goal-hash and schedule audits pass. Reexecuted Proposal, Nom./modal, and Full match all 1,440 historical construction success outcomes. Cube has one requested-action hash mismatch in Proposal and one in Nom./modal, with final-error differences below 6.4×10^{-6}; other task/arm action hashes match. This supports success-level reproduction without claiming bitwise simulator reproducibility. Tables 11–12 average seeds within scenarios and use 20,000 paired scenario-bootstrap resamples, seed 20260919. Intervals are pointwise percentile 95% intervals without multiplicity adjustment.

Table 11: Construction paired success differences (percentage points), with pointwise 95% scenario-bootstrap CIs. These compare planner variants, not isolated effects on a fixed candidate population.

Task	Full−Proposal	Full−Alloc./nom.	Full−Nom./modal
Reacher	$+0.8$ $[-10.8, 11.7]$	-4.2 $[-18.3, 9.2]$	-0.8 $[-10.0, 7.5]$
PushT	0.0 $[-5.0, 5.0]$	-10.8 $[-19.2, -3.3]$	-0.8 $[-5.8, 3.3]$
Cube	$+15.0$ $[5.8, 24.2]$	$+11.7$ $[3.3, 21.7]$	$+9.2$ $[1.7, 16.7]$
TwoRoom	$+7.5$ $[-1.7, 16.7]$	$+8.3$ $[-1.7, 18.3]$	$+4.2$ $[-4.2, 11.7]$

Table 12: Response-context and nominal-score controls on the same construction pairs. Differences are percentage points with pointwise 95% scenario-bootstrap CIs. Fixed identity retains the mixture implementation; oracle context also changes event access and posterior uncertainty.

Task	Full−Fixed identity	Oracle−Full	Alloc./nom.−Proposal
Reacher	$+2.5$ $[-7.5, 10.8]$	-0.8 $[-5.8, 4.2]$	$+5.0$ $[-5.8, 15.8]$
PushT	$+3.3$ $[-0.8, 9.2]$	$+1.7$ $[-5.0, 9.2]$	$+10.8$ $[3.3, 19.2]$
Cube	$+9.2$ $[2.5, 17.5]$	$+6.7$ $[0.8, 14.2]$	$+3.3$ $[-3.3, 10.0]$
TwoRoom	$+6.7$ $[0.8, 14.2]$	$+1.7$ $[0.0, 4.2]$	-0.8 $[-10.8, 8.3]$

Component comparisons. On Cube, Full exceeds each simplified deployable variant, including fixed identity, while oracle context provides a further positive difference. Full–Fixed identity success differences are positive for all three seeds ($+5.0, +10.0, +12.5$ points). The corresponding mean termination-time error difference is -0.00067 simulator length units ($[-0.00353, 0.00246]$). Thus the construction comparison supports higher success for the complete response-conditioned wrapper, with an inconclusive difference in average final error.

TwoRoom improves over fixed identity by 6.7 points, but its differences from Proposal, Alloc./nom., and Nom./modal include zero. This construction result concerns full-episode behavior, complementing the shared-prefix comparison in Appendix H. On Reacher, all reported success-difference intervals include zero.

PushT favors Alloc./nom. over both Proposal and Full by 10.8 points ($[3.3, 19.2]$ for each comparison). Its gains over Proposal are positive for seeds 42/43/44 ($+15.0, +7.5, +10.0$ points), while the final-error difference is inconclusive: -1.8057 environment units ($[-5.2067, 1.4505]$). PushT branch priors are identical at initialization, so this variant changes branch allocation and CEM refinement without a friction-specific action transform. This identifies nominal scoring as a useful alternative on the construction distribution; Full's formal contact-friction regression remains the main result.

What changes between variants. Alloc./nom. uses Proposal's nominal score

$$J_{\mathrm{id}}(A) + \lambda \rho_t s_J \frac{D_{\mathrm{id}}(A) - D_{\min}}{\max\{\mathrm{Quant}_{0.9}(D_{\mathrm{id}}) - D_{\min}, 10^{-6}\}}, \tag{17}$$

where $J_{\mathrm{id}}(A)$ is the raw nominal goal cost, $D_{\mathrm{id}}(A)$ the mean-squared distance to the identity prior, and $D_{\min}$ the minimum of that distance over the current candidate population. Quant_p is the empirical population quantile defined above, $\lambda = 0.35$ is the prior weight, and ρ_t is retrieval reliability.

The task-cost scale is

$$s_J = \max\{\mathrm{Quant}_{0.9}(J_{\mathrm{id}}) - \mathrm{Quant}_{0.1}(J_{\mathrm{id}}), 10^{-6}\}. \tag{18}$$

Thus s_J converts the normalized prior penalty to the numerical scale of the nominal task cost. Full uses the class-conditioned task and prior normalization in Eq. 4. The Alloc./nom.–Full comparison therefore changes both response scoring and prior normalization. Nom./modal also differs from Full in proposal implementation and prior treatment. These coupled changes make the experiments comparisons of planner variants, not a factorial estimate of allocation and ranking effects. The following diagnostics examine response inference and the value of its alignment with interaction history.

An auxiliary mixture/raw-cost variant obtains 60.0% on Reacher, 76.7% on PushT, 36.7% on Cube, and 81.7% on TwoRoom. It removes the prior-aware term as well as response scoring, whereas Alloc./nom. retains nominal prior-aware scoring. The two variants are kept distinct in the experiment archive.

Belief endpoint agreement. Diagnostics use logs from the same Full implementation as the component comparison. The model has three classes on Reacher, PushT, and Cube, and four on TwoRoom. Each observation is the final available belief-update boundary in an abrupt episode whose immediately preceding five-step block is wholly post-activation. We count one endpoint per eligible episode. Gradual episodes, untriggered episodes, and episodes without a completed post-event block are excluded. History windows may still contain pre-event blocks, and delay queues may retain earlier commands. These endpoints measure belief updates, which need not coincide with new CEM decisions.

Reference classes follow the registered evaluator convention: actuator ratio below 0.95, within $[0.95, 1.05]$, or above 1.05 for Reacher; negative/zero/positive mapping change for Cube; and absolute delay bins 0, 1–4, 5–8, and 9–12 for TwoRoom. Cube prediction and scoring apply R_α, while prior compensation applies $R_{-\alpha}$. For these scored Cube episodes, the sign of the final angle equals the sign of its change from the pre-event angle. PushT friction direction is only a proxy for reduced/amplified latent response, not a validated response-class label, so it is not reported as classification accuracy.

Table 13: Construction belief diagnostics. Agreement (%) uses one eligible abrupt-shift endpoint per episode; n is the number scored out of 120 episodes per task. Identity (%) counts all five-step belief-update boundaries, including initialization, across abrupt and gradual episodes. It is neither a CEM-decision rate nor a nominal-policy fallback rate. PushT lacks a validated physical-to-response class label.

Task	n	Agreement	Top-2	Identity
Reacher	49	65.3	83.7	42.1
PushT	41	—	—	77.1
Cube	44	93.2	97.7	58.2
TwoRoom	21	71.4	85.7	58.3

Each task has 60 abrupt episodes. Scored endpoints span 17 scenarios in Reacher, 14 in PushT, 16 in Cube, and 9 in TwoRoom. Agreement counts are 32/49 for Reacher, 41/44 for Cube, and 15/21 for TwoRoom. Registered-class counts are $(28, 0, 21)$ in reduced/identity/amplified order for Reacher, $(20, 0, 24)$ in negative/identity/positive order for Cube, and $(3, 1, 7, 10)$ in increasing delay-bin order for TwoRoom; the first two diagnostics contain no identity-reference examples. Top-2 agreement means that the reference class is among the two largest predicted probabilities. Ten-bin equal-width expected calibration error (ECE) on these selected endpoints is 0.282, 0.087, and 0.275. ECE averages the absolute gap between confidence and class agreement within probability bins, weighted by their sample counts. Calibration is consequently assessed on small, policy-dependent subsets rather than full trajectories. PushT agreement with the unvalidated friction-direction proxy is 4/41 (9.8%). Identity frequencies use 900, 837, 838, and 666 belief-update boundaries for Reacher, PushT, Cube, and TwoRoom. Early termination and late triggering affect eligibility, so endpoint agreement alone cannot establish the effect of correct classification on planning.

Table 14: Construction-only negative controls: success (%) over 40 scenarios and three seeds. Shuffled history permutes stored residual history before appending the latest block and selecting the input window. Randomized belief permutes class identities within the current probability vector. Brackets are paired scenario-bootstrap 95% CIs for Full minus each control.

Task	Full	Shuffled hist.	Randomized belief	Full−Shuffled	Full−Randomized
Reacher	57.5	56.7	52.5	$+0.8$ $[-2.5, 4.2]$	$+5.0$ $[-2.5, 12.5]$
PushT	66.7	68.3	68.3	-1.7 $[-6.7, 3.3]$	-1.7 $[-9.2, 4.2]$
Cube	70.0	67.5	65.8	$+2.5$ $[0.0, 6.7]$	$+4.2$ $[-2.5, 11.7]$
TwoRoom	90.0	91.7	85.0	-1.7 $[-5.0, 0.0]$	$+5.0$ $[0.0, 11.7]$

The controls match scenarios, seeds, CEM settings, and frozen models. Full has positive point estimates on Cube, but intervals touch or include zero. The current comparisons therefore leave its advantage over perturbed beliefs unresolved; they also do not establish equivalence. Reacher and TwoRoom lack clear separation from randomized beliefs, and both controls slightly exceed Full on PushT. Shuffling preserves interaction information and the newest block while changing older-history order and window membership. Class permutation preserves the current vector's probabilities but can change subsequent trajectories and confidence. In contrast, fixed identity removes nonidentity belief usage within the mixture solver. Together, these comparisons distinguish the value of response conditioning from the unresolved value of correctly matched histories. Independently reserved fixed-belief and cross-episode history-mismatch tests would address the latter directly.

A.4 Severity and continuous task error

We stratify the formal results by registered severity and report continuous outcomes alongside binary success. All severity groups belong to the same evaluation protocol; these descriptive slices are not held-out-severity tests and are not used for parameter selection.

Table 15: Recovery–Proposal Shift success difference (percentage points) by registered severity on the locked formal rows. This table is a stratified diagnostic, not a held-out-severity experiment.

Task	Mild	Moderate	Severe
Reacher	-4.3	$+8.7$	$+6.0$
PushT	-1.3	-5.7	-2.7
Cube	$+12.3$	$+28.0$	$+25.7$
TwoRoom	$+4.0$	$+11.0$	$+12.7$

Cube's point estimates are positive across all three severities, PushT's are negative across all three, and Reacher's are mixed. TwoRoom's full-episode gains increase with severity. This stratification characterizes closed-loop behavior; the shared-prefix interval for TwoRoom still includes zero.

Continuous error is measured at episode termination (success or budget exhaustion), not at a common terminal time. Reacher uses maximum absolute wrapped joint-angle error in radians. PushT uses Euclidean object-center distance in environment coordinate units, Cube uses object-to-goal Euclidean distance in simulator length units, and TwoRoom uses agent-to-goal Euclidean distance in environment coordinate units. These task-specific errors are not averaged across tasks. Table 16 reports all 900 pairs per task. PushT's binary decline is accompanied by worse final error; Cube and TwoRoom improve on both measures. The matched privileged reference reaches 16.9830 mean PushT error, but is not an oracle-class substitution within the identical Recovery implementation.

Strict-snapshot controls restore one canonical simulator state at event activation and reconstruct the same prefix history for each method and seed. They reuse the formal starts and give each planner the remaining control budget. At 50 total steps, Cube reaches 59.1% with Proposal and 61.1% with Recovery: the $+2.0$-point interval includes zero, although final error decreases (Table 17). TwoRoom reaches 59.9/58.9%, a paired -1.0 points ($[-3.2, 1.2]$); its final-error difference is -0.5989 environment units ($[-1.2131, 0.0048]$). The archived snapshot-hash and protocol audits pass for all 900 pairs per task. These results are distinct from replaying shared actions separately in Appendix H.

Table 16: Formal final task error at termination. Lower is better. Differences are Recovery minus Proposal, with paired scenario-bootstrap 95% CIs over 300 scenarios and three seeds. Units are task-specific, as defined in the text.

Task	Proposal	Recovery	Difference [95% CI]
Reacher	0.1018	0.0991	-0.0027 $[-0.0076, 0.0023]$
PushT	20.4040	22.0643	$+1.6604$ $[0.4597, 2.9501]$
Cube	0.0818	0.0683	-0.0136 $[-0.0179, -0.0093]$
TwoRoom	19.0320	17.0969	-1.9351 $[-2.9301, -0.8726]$

Reacher's strict-snapshot control gives Proposal/Recovery success of 36.2/34.8%, a paired change of -1.4 points ($[-5.7, 2.9]$), with higher Recovery final error by 0.0238 radians ($[0.0042, 0.0438]$). Cube's budget extensions increase the *total* environment-control budget to 60 or 70 physical steps, including the prefix. Each budget uses 300 scenarios and seeds $\{42, 43, 44\}$ (900 pairs), with the same 300-sample, 30-iteration CEM configuration. The 50-step baseline and longer-budget results are reported together to distinguish inconclusive success at the original budget from gains under an extended protocol.

Table 17: Cube strict-snapshot handoff at the original 50-step budget and 60/70-step extensions. Success rates are percentages; gains are percentage points. Error differences use simulator length units. Both differences are Recovery minus Proposal with paired 95% CIs over 300 scenarios and three seeds.

Total steps	Proposal	Recovery	Success gain [95% CI]	Error change [95% CI]
50	59.1	61.1	$+2.0$ $[-1.7, 5.7]$	-0.0046 $[-0.0090, -0.0002]$
60	59.6	78.3	$+18.8$ $[14.1, 23.6]$	-0.0115 $[-0.0153, -0.0079]$
70	61.1	90.3	$+29.2$ $[24.4, 34.1]$	-0.0310 $[-0.0370, -0.0251]$

B Per-Task Event Construction

Figure 3 summarizes the state-triggered construction used by SHIFTWM-BENCH. Each row starts with pre-event execution, marks the physical trigger, and continues under a hidden response change. The family is predefined for each task; the event time and realized response parameters are withheld from the planner.

C Three-Task and Routed-Transfer Diagnostic

The primary aggregate in Tables 2–4 is Macro-4. Figure 5 retains a descriptive Macro-3 view over Reacher, Cube, and TwoRoom, with PushT shown separately. This subset is an analytical grouping, not an independently validated task suite. Panel B uses routed Fast rather than the all-task raw Recovery evaluation; it does not replace the four-task transfer table.

D Supplementary PushT Results

PushT contributes equally to the primary four-task aggregate. Native LeWM obtains 80.9% success, while Proposal Shift and Proposal + Recovery obtain 72.8% and 69.6%. Trigger rates are approximately 97–98% across all four arms, so event avoidance does not explain the decline. The matched privileged reference reaches 77.1%; supplying the shift parameter to its fixed correction also does not improve on Native. The tested latent scale approximation does not explicitly model contact geometry or friction-dependent motion. The observed regression motivates contact-conditioned scoring, rather than treating friction direction as a validated latent response label.

The success decline is accompanied by an increase in continuous final error (Table 16), and all three severity groups have negative Recovery deltas. These observations establish a failure case for the evaluated interface, rather than an artifact of the binary success threshold.

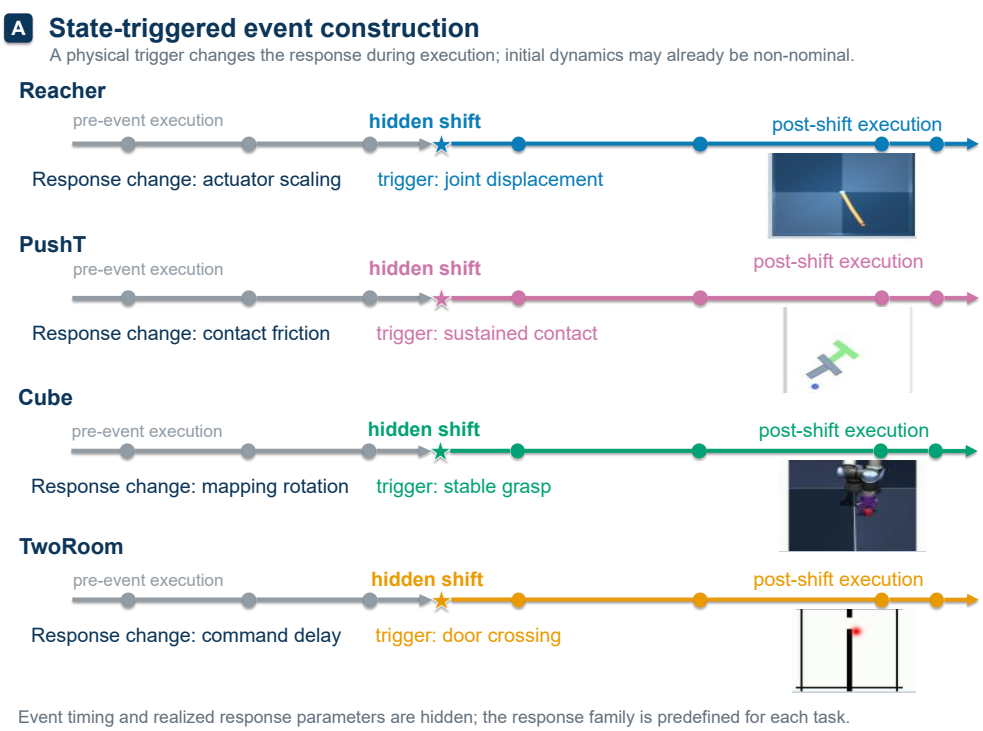

Figure 3: State-triggered event construction in SHIFTWM-BENCH. Rows show Reacher actuator scaling, PushT contact friction, Cube action rotation, and TwoRoom command delay. Screenshots are representative observations; timeline spacing is schematic, not measured duration. All formal scenarios retain the same task-specific trigger semantics.

Table 18: PushT diagnostic on the locked formal split. Trigger rate is the fraction of episodes in which the state-triggered contact event was confirmed. The nearly unchanged trigger rates show that the difference is not caused by one method simply avoiding the event.

Method	Success (%)	Trigger rate (%)
Native LeWM	80.9	97.2
PROPOSAL SHIFT	72.8	98.0
PROPOSAL SHIFT + Recovery	69.6	97.7
Matched privileged reference	77.1	98.0

E TRIGGER-COVERAGE DIAGNOSTIC

Table 19 reports the rate at which each arm reaches and confirms the hidden event. This diagnostic is important for interpreting the low Native LeWM score on Cube: the native candidate generator reaches the stable-grasp trigger in only 7.0% of episodes, whereas the proposal and complete systems reach it in 79.7% and 84.6%. Improved trigger coverage and a positive shared-prefix effect are distinct observations; these rates do not give an additive causal decomposition of the main success gain. The trigger rates for Reacher and PushT are already above 95%, so their smaller recovery deltas are not explained by missing the event.

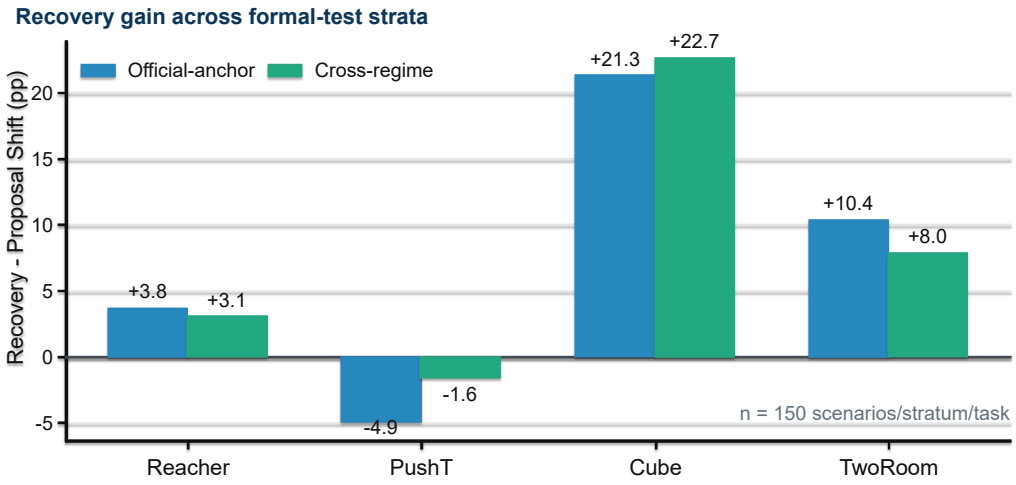

Figure 4: Descriptive full-episode differences across the two formal-test strata. Bars show Recovery–Proposal success differences for official-anchor and cross-regime subsets, each containing 150 scenarios per task. Bars have no stratum-specific uncertainty intervals; pooled task intervals in Figure 5 do not quantify the uncertainty of these individual bars. The split view describes distribution sensitivity, not a causal recovery effect.

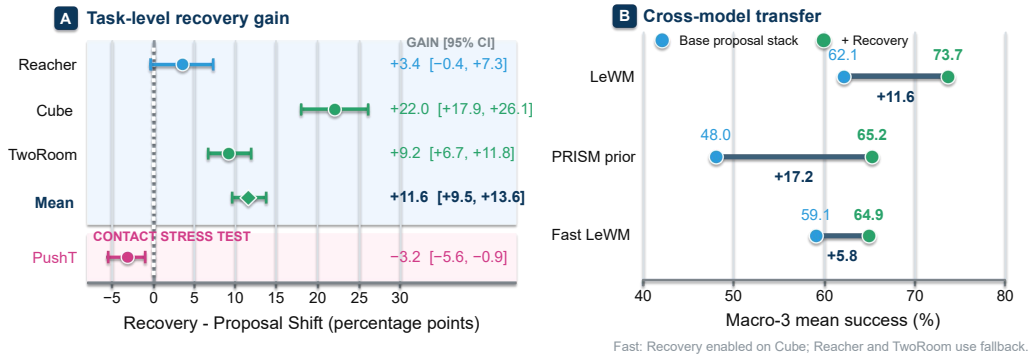

Figure 5: Secondary full-episode diagnostic, not post-trigger attribution. (A) The plotted "recovery gain" is the full-episode LeWM Recovery–Proposal success difference with pointwise 95% scenario-bootstrap CIs. Macro-3 averages Reacher, Cube, and TwoRoom; PushT remains visible separately. The shared-prefix Cube gain (+4.6 points) is a different endpoint (Table 28). (B) The three-task diagnostic for LeWM, PRISM, and routed Fast, with Recovery on Cube only and Proposal elsewhere. Table 4 instead reports the primary four-task comparison with active Recovery on every task.

F Matched No-Shift and Adaptation Controls

In Tables 20 and 25, Δ is the Recovery-minus-Proposal success difference within the condition named by the column. $\Delta\Delta$ subtracts the no-shift Δ from the shifted Δ on matched scenarios. Both are reported in percentage points, not relative percent improvements, and are computed from unrounded episode-level outcomes.

Table 21 reports the complete matched no-shift evaluation. It uses the same formal scenario sources, starts, goals, and planner seeds as the shifted evaluation, but sets the effective post-event parameter equal to the pre-event parameter. The original scenario metadata retains its registered shift. Static behavior is therefore measured on the same conditional start distribution without a within-episode dynamics change.

Table 19: Confirmed event-trigger rates (%) on the locked formal split. Rates aggregate 300 scenarios and three planner seeds.

Task	Native	PROPOSAL SHIFT	PROPOSAL SHIFT + Recovery	Privileged ref.
Reacher	96.0	95.7	95.4	95.7
PushT	97.2	98.0	97.7	98.0
Cube	7.0	79.7	84.6	79.7
TwoRoom	73.3	93.3	93.4	93.3

Table 20: Shifted and matched no-shift comparisons. Values are paired Recovery–Proposal success differences in percentage points. The final column subtracts the no-shift difference from the shifted difference, with pointwise scenario-bootstrap 95% CIs. This is a condition contrast, not an identical-state post-trigger intervention.

Task	Shifted Δ	No-shift Δ	Shift-specific $\Delta\Delta$
Reacher	$+3.4$	-0.1	$+3.6$ $[-1.3, 8.4]$
PushT	-3.2	-2.7	-0.6 $[-2.9, 1.9]$
Cube	$+22.0$	$+8.1$	$\mathbf{+13.9}$ $[\mathbf{10.6}, \mathbf{17.3}]$
TwoRoom	$+9.2$	-0.3	$\mathbf{+9.6}$ $[\mathbf{7.2}, \mathbf{12.0}]$
Macro-4	$+7.9$	$+1.3$	$\mathbf{+6.6}$ $[\mathbf{4.9}, \mathbf{8.4}]$
Macro-3 (descriptive)	$+11.6$	$+2.6$	$+9.0$ $[6.9, 11.2]$

Table 21: Matched no-shift success rates (%). Macro-4 is computed from unrounded episode-level rates. Each cell aggregates 300 scenarios and three planner seeds, including both nominal and fixed non-nominal pre-event dynamics.

Method	Reacher	PushT	Cube	TwoRoom	Macro-4
Native LeWM	**68.7**	**82.1**	3.3	69.7	55.9
PROPOSAL SHIFT	63.9	77.3	72.9	**97.1**	77.8
PROPOSAL SHIFT + Recovery	63.8	74.7	**81.0**	96.8	**79.1**

Nominal-parameter subset. Table 22 isolates the 150 official-anchor scenarios per task, whose pre-event parameter is the task's nominal value. With the event disabled, these runs use nominal dynamics throughout. They retain the benchmark's conditional starts and goals, so they are not a reproduction of an official benchmark's independently sampled replay protocol. The subset makes the distinction between nominal dynamics and matched fixed non-nominal dynamics explicit.

Table 22: Nominal-parameter, no-shift subset of the LeWM evaluation: success (%) over 150 official-anchor scenarios and three seeds per task. Starts and goals are those of ShiftWM-Bench.

Method	Reacher	PushT	Cube	TwoRoom
Native LeWM	84.7	80.4	3.6	76.2
PROPOSAL SHIFT	74.7	76.2	74.0	98.7
PROPOSAL SHIFT + Recovery	74.0	72.2	80.9	97.1

No-shift schedule verification. The archived no-shift checker compared some recorded parameters with the original shifted manifest rather than the effective fixed-pre schedule. All saved mismatch values equal the scenario's pre-event parameter. A deterministic diagnostic replay of the first scenario in each stratum, seed 42, and all three arms (24 episodes in total) records the full physical schedule: all 781 recorded steps remain at the pre-event value, and success, requested/executed action hashes, and final errors exactly match the corresponding archived episodes. This supports reuse of the archived results and identifies the checker mismatch. Full physical traces were not retained for every historical episode; the audit archive preserves that provenance limit and the original flags without overwriting them.

Table 23: Non-privileged online adaptation on the three-task diagnostic. Residual-CUSUM uses a fixed residual test with single-response planning; Recovery uses learned beliefs and mixture CEM. Both use the same frozen predictor, retrieval data, and candidate budget.

Method	Reacher	Cube	TwoRoom	Mean
PROPOSAL SHIFT	50.3	54.1	82.0	62.1
Residual-CUSUM	50.9	63.9	87.0	67.3
PROPOSAL SHIFT + Recovery	**53.8**	**76.1**	**91.2**	**73.7**

Residual-CUSUM implementation. This cumulative-sum (CUSUM) baseline accumulates evidence that a nonidentity response predicts recent observations better. At completed block j, it subtracts the smallest nonidentity normalized error from the identity normalized error and clips the difference below at zero. Denote this nonnegative evidence by E_j. The detector state is

$$S_j = \max(0, 0.95S_{j-1} + E_j - 0.5), \qquad S_0 = 0. \tag{19}$$

Here S_j is accumulated evidence through block j, S_{j-1} its previous value, 0.95 a decay factor, and 0.5 a fixed per-block drift penalty. The outer maximum prevents negative accumulated evidence. When $S_j \geq 1$ and two consecutive blocks have $E_j > 0.5$, the detector selects the coarse representative nearest the lowest-error nonidentity grid value; otherwise it selects identity. Thus 1 is the accumulated-evidence threshold, and the two-block rule requires persistent support. The selected response conditions the prior and correction in retrieval-initialized CEM. It shares response operators and evaluation budgets with Recovery but not belief-weighted allocation or the mixture solver, making it a complete-adapter baseline rather than an encoder-only ablation.

The four-task shifted mean of Residual-CUSUM is 68.6%, compared with 64.8% for PROPOSAL SHIFT and 72.7% for Recovery. The paired CUSUM–Proposal gain is $+3.8$ points (95% CI $[3.0, 4.7]$), while Recovery exceeds CUSUM by $+4.0$ points ($[2.5, 5.6]$). On PushT, CUSUM matches the 72.8% Proposal success rate and Recovery falls to 69.6%. These results compare the decision policies produced by the complete adapters.

G Cross-Model Evaluation Details

G.1 Fast Recovery on all tasks

All Fast arms share official checkpoints, latent retrieval caches, formal scenarios, seeds, and CEM settings. Raw Recovery is enabled on every task without an additional task gate. Shifted and matched no-shift conditions each contain 300 scenarios and three seeds per task, for 7,200 raw-Recovery episodes. Native and Proposal contribute a further 7,200 matched no-shift episodes, giving 10,800 static episodes across three evaluated planners. Routed static results apply the fixed task rule to these rows and require no additional rollouts. Configuration, scenario/seed, source/start, initial/goal-hash, and schedule checks verify the intended pairing. The raw Cube run reproduces all 900 success outcomes from the routed Cube evaluation.

Table 24: Fast matched no-shift success (%): 300 scenarios and three seeds per task and evaluated planner. Dynamics remain at each scenario's pre-event value, including nominal and fixed non-nominal values. Recovery is enabled on every task.

Planner	Reacher	PushT	Cube	TwoRoom	Macro-4
Native CEM	49.0	88.1	4.8	85.7	56.9
Proposal	48.4	87.9	71.9	99.1	76.8
+ Recovery	50.2	84.2	74.0	97.9	76.6

Table 24 measures static performance on the benchmark's conditional starts, not the official static replay protocol. Relative to Proposal, raw Recovery changes static success by -0.25 points ($[-1.44, 0.94]$) in Macro-4. This interval does not establish equivalence: PushT and TwoRoom have negative task-level intervals, whereas the Reacher and Cube intervals include zero.

Table 25: Fast raw-Recovery–Proposal success differences in percentage points, with pointwise paired 95% CIs. The final column subtracts the no-shift difference from the shifted difference within each scenario. Seeds are averaged within scenario before 20,000 bootstrap resamples; Macro-4 independently resamples scenarios within each task.

Task	Shifted Δ [95% CI]	No-shift Δ [95% CI]	$\Delta\Delta$ [95% CI]
Reacher	-0.33 $[-3.56, 2.89]$	$+1.78$ $[-1.33, 4.89]$	-2.11 $[-5.56, 1.33]$
PushT	-3.89 $[-6.11, -1.78]$	-3.67 $[-5.78, -1.89]$	-0.22 $[-1.56, 1.00]$
Cube	$+17.33$ $[13.11, 21.56]$	$+2.11$ $[-0.78, 4.89]$	$+15.22$ $[11.89, 18.67]$
TwoRoom	$+9.56$ $[6.78, 12.44]$	-1.22 $[-2.44, -0.22]$	$+10.78$ $[8.11, 13.56]$
Macro-4	$+5.67$ $[4.08, 7.25]$	-0.25 $[-1.44, 0.94]$	$+5.92$ $[4.53, 7.33]$

The positive Cube and TwoRoom condition contrasts in Table 25 show that their shifted gains are not matched by equally large static gains. PushT loses success under both conditions, so its regression is not specific to the friction event. These are descriptive closed-loop condition contrasts: matched starts do not force identical trajectories or post-trigger states.

Raw Recovery also reduces shifted termination-time error on Cube by 0.00799 simulator length units ($[0.00394, 0.01205]$) and on TwoRoom by 1.7610 environment units ($[0.9089, 2.5909]$), relative to Fast Proposal. PushT error increases by 2.1890 units ($[0.6858, 3.8549]$); Reacher's error difference includes zero. Thus the two positive shifted success effects also appear in continuous outcomes. Under no shift, however, final error increases on PushT by 2.3130 units ($[0.7806, 3.9877]$) and on TwoRoom by 0.9370 units ($[0.3563, 1.6473]$), consistent with their static success losses.

Historical task-route diagnostic. The following archived route analysis is separate from the all-task Recovery results in Tables 4 and 24; no routed score contributes to their Macro-4 means. The Fast task route admits a task when a one-sided 95% lower bootstrap bound for raw Recovery minus Proposal on matched no-shift construction data is nonnegative. The original calibration used 100,000 resamples of the 120 episode–seed pairs per task. Clustering the three seeds within each of the 40 scenarios and recomputing the same rule leaves the route unchanged (Table 26): Cube is

enabled, and the other tasks retain Proposal. The two-sided Cube interval contains zero, which is compatible with a positive one-sided 95% lower bound.

Table 26: Fast route sensitivity on 40 matched-static construction scenarios per task. Differences and bounds are percentage points for raw Recovery minus Proposal. The scenario-clustered admission decision agrees with the historical gate.

Task	Difference	Two-sided 95% CI	One-sided lower	Route
Reacher	-5.8	$[-14.2, 2.5]$	-12.5	Proposal
PushT	-1.7	$[-8.3, 3.3]$	-6.7	Proposal
Cube	$+6.7$	$[-0.8, 14.2]$	$+0.8$	Recovery
TwoRoom	-1.7	$[-4.2, 0.0]$	-3.3	Proposal

This fixed route yields 69.9% shifted Macro-4 success, versus 71.2% for raw Recovery. It avoids the observed PushT decline but omits the TwoRoom benefit, illustrating the trade-off from a task route selected on static construction behavior. The rule provides an empirical admission criterion, not a guarantee of test-time non-regression.

G.2 PRISM AND PAPER-SPEC IMWM INTERFACES

The PRISM common-CEM interface uses released Cube and PushT prior heads and locally trained nominal-data extensions of the same architecture for Reacher and TwoRoom. The extension checkpoints retain their nominal validation histories and training arguments in the supplementary material. They were saved with the upstream trainer's diagnostic gates overridden: their validation improvement and predicted-standard-deviation spread did not meet those gates. These are fixed-prior interface experiments, not reproductions of the complete native PRISM planner. The two rows in Table 4 report the base adapter and raw Recovery on all four tasks. The raw four-task Recovery mean is 68.9%. A separate routed variant that retains the base adapter on PushT gives 69.7%; it is not mixed into the raw table.

The IMWM experiment uses a paper-spec reimplementation under common CEM, not an official IMWM checkpoint. Nominal construction diagnostics select an intuition-dominant recipe on Reacher and TwoRoom and a trusted-hybrid recipe on PushT and Cube. Its latent predictor remains LeWM. Table 4 includes active Recovery on every task. The original PushT `imwm_recovery_shift` arm used task-level fallback in all 900 episodes; its requested-action hashes matched the base adapter. We retain this historical route separately in Table 27.

The active PushT completion uses the same 300 formal-test-v1 scenarios and seeds 42–44, frozen predictor, inverse scorer, belief, and trusted-hybrid recipe. It directly invokes the existing Recovery policy without task routing, retaining identity scoring inside the mixture solver. No training or hyperparameter selection is performed. The budget remains 50 physical steps, 300 candidates per CEM iteration, 30 iterations, and 30 elites. Before the full evaluation, four base-adapter episodes reproduce the archived success and action hashes, and a construction smoke check verifies active-policy execution. All 900 new episodes pass the manifest, schedule, and leakage checks; their policy diagnostics contain no task-level fallback.

Table 27: Paper-spec IMWM common-CEM adapter on formal-test-v1: success (%) over 300 scenarios and three seeds per task. The active row enables Recovery on every task and is used in Table 4. The historical route instead retains the base adapter on PushT.

Planner	Reacher	PushT	Cube	TwoRoom	Macro-4
IMWM adapter	51.1	66.1	43.2	64.0	56.1
+ Recovery (active)	55.4	62.6	67.0	69.9	63.7
+ Historical route	55.4	66.1	67.0	69.9	64.6

For active Recovery, the paired Macro-4 increase is 7.6 points ($[5.4, 9.9]$). Task differences are $+4.3$ ($[-0.7, 9.3]$) for Reacher, -3.6 ($[-8.0, 0.9]$) for PushT, $+23.8$ ($[19.2, 28.3]$) for Cube, and $+5.9$ ($[1.8, 10.0]$) for TwoRoom. These intervals use 20,000 scenario-bootstrap draws after averaging

seeds, with independent within-task resampling for Macro-4. Active PushT Recovery succeeds in 563 of 900 episodes, versus 595 for the base; its final-error difference is $+0.414$ environment units ($[-2.119, 2.930]$). Both PushT intervals include zero. All comparisons use complete 900-episode arms joined by scenario and seed, including the separately completed Reacher and PushT files. The historical routed mean of 64.6% is excluded from the all-active comparison.

H SHARED-ACTION-PREFIX CONTROL

Table 28: Scripted-prefix post-trigger control. All methods execute the same actions through event activation and then plan independently. The final column is the paired Recovery–Proposal difference with a scenario-bootstrap 95% CI.

Task	Native	PROPOSAL SHIFT	Recovery	Rec.–Prop.
Cube	44.6	58.9	**63.4**	$+\mathbf{4.6}$ $[\mathbf{1.6}, \mathbf{7.7}]$
TwoRoom	59.2	**59.9**	59.3	-0.6 $[-2.3, 1.2]$

The scripted-prefix evaluation executes the source action sequence through the manifest's registered activation step. No evaluated arm succeeds before activation. Prefix lengths range from 10 to 15 steps on Cube and from 10 to 17 on TwoRoom, with a median of 12 for both tasks. Requested and executed action hashes match across all three methods for every episode–seed pair. State-prefix hashes match for all 900 TwoRoom pairs and for 857/900 Cube pairs. The 43 Cube mismatches occur in 19 scenarios and begin at reset despite identical action prefixes, indicating simulator-level replay nondeterminism rather than policy-dependent pre-event actions. Dropping all 19 affected scenarios changes the Cube Recovery–Proposal estimate from $+4.6$ ($[1.6, 7.7]$) to $+5.2$ points ($[2.0, 8.4]$).

I REPRODUCIBILITY ARTIFACTS

The anonymous supplementary archive contains construction and formal manifests, portable evaluation code, small trained components, code/model checksums, evaluation instructions, paired episode records, and frozen training and routing configurations. It documents that model parameters, IMWM recipes, and task routes were not fit using real shifted or formal-test trajectories. Supplemental protocols, source hashes, output hashes, budget accounting, and statistical verification accompany Appendix A.2. Included scripts recompute the reported success rates and paired comparisons offline. Fresh simulator runs additionally require the documented public dependencies, datasets, and large backbone checkpoints. Cache-building scripts reconstruct retrieval features from these public resources; author-private caches are not a required input. The archive records training/evaluation seeds, environment versions, and third-party licenses.