RESYNC: RE-ALIGNING THE TWO CLOCKS OF ASYNCHRONOUS WORLD-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Joint world–action models often denoise actions quickly while refining future video for many more steps. That asymmetry improves latency and world quality, but it also creates two inference clocks: an action can become executable before the world evidence that should justify it has resolved. We use the *commitment–evidence gap* to diagnose this mismatch. Across a controlled X-WAM schedule family, widening the gap produces an empirical ordering reversal: candidate selection loses value while advancing world evidence becomes the better use of test-time compute. **ReSync** turns that observation into an inference rule: it inserts world-only computation at a supported point where the action can still change, then resumes the native action schedule without adding action-denoising updates. On a frozen 960-rollout RoboCasa panel, ReSync improves X-WAM from 80.31% to 84.79% (+4.48 pp). At the same $3.40\times$ forward budget, five natural alternatives—waiting, resampling candidates, extending action denoising, synchronizing the streams, and warm-starting the world—remain between 80.52% and 82.92%. Even a $5.00\times$ fully synchronized schedule reaches only 83.23%. Success peaks inside the supported world-advancement range rather than growing indefinitely. Fixed on the native schedule, the rule selects points on two held-out schedules that remain competitive among the tested neighbours. It also improves RoboTwin 2.0 by 2.67 pp and a two-clock Cosmos Policy instantiation by 4.38 pp. Which clock the extra computation advances matters more than how much of it there is.

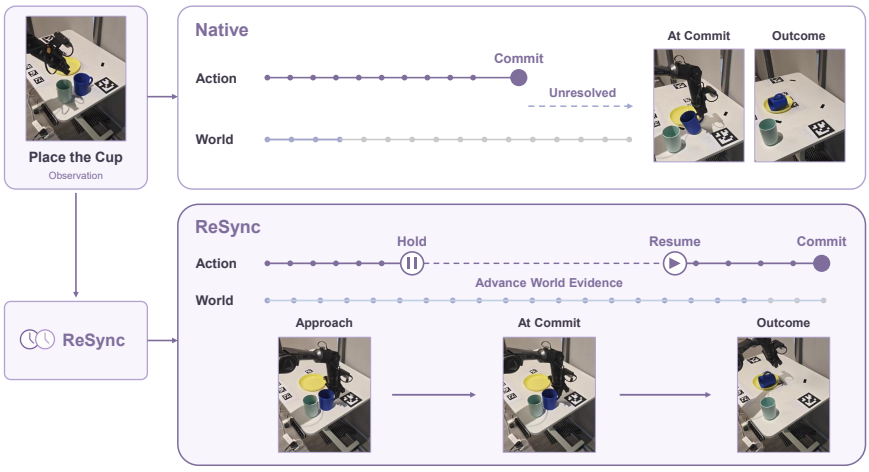

Figure 1: **Action commitment can precede useful world evidence.** Native X-WAM commits after the ten-step action schedule finishes; ReSync holds that action state, advances the world inside supported denoising time, and resumes native action denoising before commitment. Photographs are recorded execution frames; blur indicates evidence resolution schematically and is not model-decoded output.

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.

1 INTRODUCTION

World-action models make an appealing promise for manipulation: the policy predicts a future and acts inside that prediction with one generative model. The strongest systems often give those two outputs different denoising schedules. Actions are decoded quickly because control is latency-sensitive; future video is refined longer because a world prediction is useful only if it is sharp enough to carry task evidence. The price is easy to miss: action and world no longer run on the same clock.

The robot reaches for a cup. Its action is already executable, while the imagined future that should justify the reach is still taking shape. Nothing has failed to generate. The decision has simply arrived before its evidence.

X-WAM makes this timing structure explicit: the action stream finishes in ten updates while the world follows a 50-step trajectory (Guo et al., 2026). Native execution therefore commits after only about one fifth of the world schedule has elapsed. We call the mismatch the **commitment–evidence gap**. The commitment clock measures progress toward an executable action; the evidence clock measures progress toward a resolved future.

> **Two-clock principle.** When commitment and evidence run on separate clocks, useful test-time compute must make evidence available while the action can still change.

This principle separates three ideas that are easy to conflate. The *gap* is a diagnostic: it says how far the evidence clock lags at commitment. The *crossover* is an empirical finding: in a controlled family of schedules from the same checkpoint, candidate selection is the better use of compute at small gaps, while world-evidence advancement becomes better after the ordering reverses between the measured $G = 0.44$ and $G = 0.58$ schedules. *ReSync* is the method: once the timing problem is visible, it uses the native schedules plus one fixed recovery-margin measurement to decide where world-only computation can be inserted.

The matched-compute experiments show why this distinction matters. Simply running the action longer does not reproduce the gain. Neither does spending the same budget on more candidates, synchronizing the streams, or sharpening the world before joint denoising. Even Sync-50—the literal solution of denoising both streams for 50 steps—uses $5.00\times$ native compute and still trails ReSync at $3.40\times$. The world warm-start control is especially revealing: a sharper future by itself is not enough. Useful evidence has to arrive at a joint state the model was trained to couple, and early enough that the remaining action updates can absorb it.

That observation turns the problem from "how much more should we denoise?" into "when can extra world computation still change the decision?" Two placement constraints come directly from the sampler and training schedule: the hold must occur on an executed action state, and the advanced world/action pair must remain in the represented joint timestep regime. A third constraint—the number of native action updates needed for recovery—is fixed once by a coarse placement scan and then held constant. Together they give the ReSync operating region.

On the primary RoboCasa panel, this rule raises X-WAM from 80.31% to 84.79%. The gain is distributed across 17 of 24 tasks and every perturbation group. A world-advance sweep rises to the schedule-supported $M = 24$ point and then falls, showing that the useful region is internal rather than "more world compute is better." The same rule transfers unchanged to RoboTwin and, after applying the same action-recovery convention to its constructed two-clock interface, to Cosmos Policy. Across these results, the recurring pattern is simple: **world prediction is valuable when it becomes available while the action can still use it.**

2 RELATED WORK

Predictive policies couple control to future generation through video or joint world–action latents (Du et al., 2023; Wu et al., 2024; Cheang et al., 2024; Hu et al., 2024; Li et al., 2025; Zhang et al., 2025). X-WAM and Cosmos Policy expose the two-clock coupling studied here (Guo et al., 2026; Kim et al., 2026). Candidate and search methods spend test-time compute on alternatives or verification (Kwok et al., 2025; Zhao et al., 2026a;b; Chen et al., 2026); timestep and cache methods redistribute sampling effort (Pan et al., 2024; Sun et al., 2026; Huang et al., 2026; Wang et al., 2026; Li et al.,

2026). ReSync freezes the existing model, keeps the native number of action updates, and uses its joint timestep support to move world computation into a part of inference where the action can still respond. Appendix A compares the mechanisms directly.

3 THE TWO CLOCKS: DIAGNOSIS AND EMPIRICAL CROSSOVER

3.1 THE GAP AS A DIAGNOSTIC

A WAM that generates actions and future observations has two conceptually different progress variables. The *commitment clock* measures progress toward an executable action; the *evidence clock* measures progress toward a resolved world prediction. A synchronous sampler ties the two together. An asynchronous sampler does not.

Let τ_c denote action commitment. The semantic quantity of interest is how much decision-relevant world evidence has resolved by τ_c; the architecture gives us a direct timing proxy for it:

$$\hat{r}_{\mathrm{sched}}(\tau_c) = \frac{n_O(\tau_c)}{T_O}, \qquad G(\tau_c) = 1 - \hat{r}_{\mathrm{sched}}(\tau_c).$$

We refer to G as the **commitment–evidence gap**, with $\hat{r}_{\mathrm{sched}}$ understood throughout as normalized schedule progress rather than a measured fraction of semantic evidence. It requires no task reward, critic, or additional model. X-WAM executes ten action updates against a 50-step world trajectory (Guo et al., 2026), so native commitment occurs at $\hat{r}_{\mathrm{sched}} = 10/50 = 0.20$ and $G = 0.80$.

We measure whether that unresolved state is actually useful for choosing among candidate actions with a within-set identifiability statistic. For a representation score r, let $\mathrm{AUC}_{\mathrm{within}}(r)$ be the AUC against closed-loop candidate outcome computed only within candidate sets. For the tested representation family $\mathcal{R}$,

$$\kappa = \frac{\max_{r \in \mathcal{R}} \mathrm{AUC}_{\mathrm{within}}(r) - \mu_0}{1 - \mu_0},$$

where μ_0 is the mean of the familywise permutation-max null: each null draw permutes outcomes within candidate sets, recomputes every score in $\mathcal{R}$, and then takes the maximum AUC. Thus $\kappa = 0$ places the best tested representation at its familywise null level, while $\kappa = 1$ denotes perfect within-set separation. The screen fixes 80 candidate sets over 22 tasks (2,560 trajectories), the representation family, and 10,000 permutations across schedules; Appendix C.1 gives the full protocol.

Holding candidates and scorer fixed, advancing only the world before commitment raises κ from 0.0187 to 0.063, 0.118, and 0.171. The schedule scan tests the same ordering.

3.2 AN EMPIRICAL CROSSOVER IN SCALING VALUE

We construct six inference schedules from one frozen X-WAM checkpoint, fixing the ten-step action schedule and evaluation protocol while changing only T_O. At the largest gap, candidate utility is nearly unidentifiable ($\kappa = 0.0187$). As the world resolves further before commitment, κ and selector gains rise, while the gain from advancing evidence falls.

The "Selector" column uses one fixed label-free scorer and 32-candidate bank in every row (distinct from Appendix B.3's learned selector). Table 1 is diagnostic; the equal-budget test appears in Table 2.

The ordering flips between Mid-18 and Mid-24: an empirical regime marker, not an exact threshold. All six rows use one checkpoint without retraining. BASE success is non-monotonic in T_O; shortening the world schedule closes the gap but sacrifices refinement. The intervention must preserve the world trajectory while changing when its evidence reaches the action.

Appendix C.1 reports a routing-map audit confirming that the implemented action-timestep map matches the executed schedule.

4 METHOD: RESYNC

The gap tells us whether evidence is late; the crossover tells us which compute axis paid off in the controlled schedule family. Neither specifies an intervention. ReSync is the step that turns those

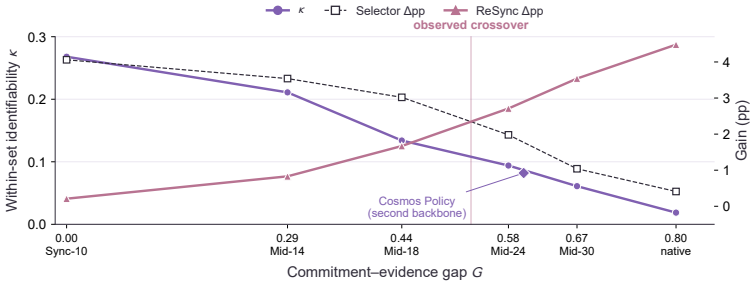

Figure 2: **The preferred compute axis reverses as the gap grows.** Larger gaps coincide with lower κ and smaller selector gains, but larger evidence-advancement gains. The reversal lies between the measured $G = 0.44$ and $G = 0.58$ schedules; the marker interpolates.

Table 1: **Empirical crossover across six schedules of one checkpoint.** Only T_O changes. The selector uses a fixed $N = 32$ bank; ReSync inserts $M = \{0, 2, 4, 8, 12, 24\}$ world updates. These budgets are not matched; Table 2 supplies the equal-budget comparison.

Setting	T_O	$\hat{r}_{\text{sched}}$	G	BASE	κ	Selector Δpp	ReSync Δpp
Sync-10	10	1.00	0.00	76.15	0.268	+4.06	+0.21[†]
Mid-14	14	0.71	0.29	79.48	0.211	+3.54	+0.83
Mid-18	18	0.56	0.44	81.04	0.134	+3.02	+1.67
Mid-24	24	0.42	0.58	81.25	0.094	+1.98	+2.71
Mid-30	30	0.33	0.67	80.73	0.061	+1.04	+3.54
X-WAM (native)	50	0.20	0.80	80.31	0.0187	+0.41	+4.48

[†] For SYNC-10, $M_{\max} = 0$: world and action finish together, so ReSync degenerates to an identity operation.

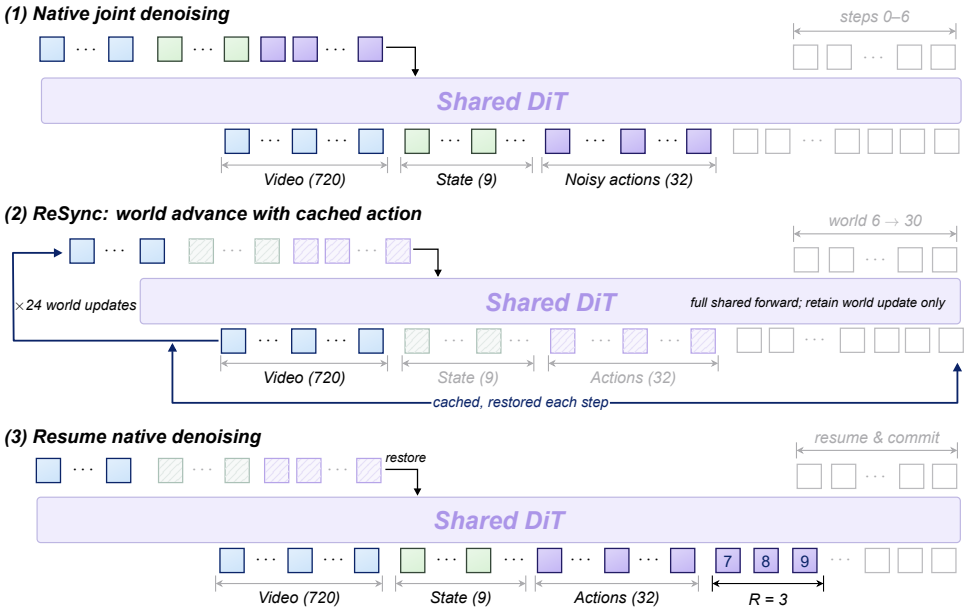

Figure 3: **ReSync advances only the evidence clock inside the usable window.** The action state is cached, the world advances, and native action denoising resumes while recovery budget remains.

observations into an inference rule. It asks where world-only computation can be inserted so that the resulting world/action state is supported and the action still has time to change.

4.1 FROM DIAGNOSIS TO AN ADMISSIBLE WINDOW

The commitment gap is the gate: if evidence is already resolved at commitment, there is little reason to move world computation earlier. Once the gap is present, three placement constraints determine where an intervention is usable.

Executed action support. The hold is placed only at action states visited by native inference, $j \in \mathcal{S}_{\text{exec}}$. Appendix C.1 audits this map directly.

Action recovery. A world update matters only if native action denoising still has room to absorb it. A coarse placement scan fixes $R_{\min} = 3$ once; the same margin is then used for RoboCasa, RoboTwin, and Cosmos.

Joint timestep support. X-WAM's asynchronous-noise sampling (Sec. 3.3 of Guo et al. (2026)) constructs the noisy world time as the action time plus a nonnegative residual, so the represented joint noisy regime obeys $t_O \geq t_a$ when both streams are denoised.[1] We therefore define

$$M_{\max}(j) = \max\{m : t_O^{\psi(j)+m} \geq t_a^j\}.$$

Together these constraints define

$$\mathcal{F} = \{(j, M) : j \in \mathcal{S}_{\text{exec}},\ J - 1 - j \geq R_{\min},\ 0 \leq M \leq M_{\max}(j)\}.$$

The schedule supplies executed support and joint timestep support; the placement scan supplies the single recovery margin. Once fixed, none of these quantities is retuned on the transfer evaluations.

Why not just shorten the world schedule? Table 1 shows why. BASE success is non-monotonic in T_O: shortening the world schedule reduces the gap, but beyond a point it also removes useful generative refinement. ReSync keeps the longer world trajectory and changes when a supported portion of it becomes available to the action.

4.2 The ReSync operator

The operator itself is almost disappointingly simple, and that is the point: the schedule has already made the hard allocation decision. At a feasible placement j, ReSync caches the action latent and its timestep, runs ordinary shared-DiT forwards while retaining only the world update, and restores the cached action after each forward. After M such updates, native action denoising resumes from the saved state. The action therefore sees a more resolved world without adding an action-denoising step, parameter, critic, selector, or verifier.

Algorithm.

1. Read the native world/action schedules and compute $M_{\max}(j)$.
2. Apply the fixed recovery margin $R_{\min} = 3$ and choose the latest feasible placement with the largest supported world advance.
3. Cache the action state; perform M world-only forwards, restoring the cached action after every forward.
4. Resume the native action sampler and commit after its remaining updates.

4.3 Operating point

For X-WAM, $J = 10$ and

$$M_{\max}(j) = [0, 4, 8, 12, 16, 20, 24, 28, 32], \qquad j = 0, \ldots, 8.$$

The recovery constraint removes $j \geq 7$, leaving $(j^\star, M^\star) = (6, 24)$ as the latest feasible point with maximum supported advancement. Native X-WAM commits after 10 of 50 world updates, so the schedule-progress proxy is $\hat{r}_{\text{sched}} = 10/50 = 0.20$. ReSync inserts 24 world-only updates while preserving the same ten action updates; at commitment the world has completed 34 of 50 updates and $\hat{r}_{\text{sched}} = 34/50 = 0.68$.

The same action-recovery convention also defines our Cosmos interface. We construct a 14-update action stream against its 35-step future-state schedule; with $R_{\min} = 3$, the latest safe cache is $j_C = 10$. Cosmos uses a different training construction from X-WAM, so the transfer interface fixes a ten-step world-only advance, $M_C = 10$, before closed-loop evaluation. All Cosmos results below refer to this constructed $(J_C, j_C, M_C) = (14, 10, 10)$ interface. BASE and ReSync execute the same 14 action updates.

[1] Released implementation: `https://github.com/sharinka0715/X-WAM/blob/main/runners/xwam_runner.py`, `training_step`, joint-timestep sampling.

4.4 COMPUTE

Each inserted X-WAM world-only update is one shared-DiT forward, so $C_{\mathrm{RS}} = 1 + M/10$ and the default point costs $3.40\times$ native forward compute. Section 5.2 spends that same 34-forward budget on five other allocations, making *where* the compute goes the controlled variable.

5 EXPERIMENTS

Benchmarks and protocol. The primary evaluation is a frozen clustered paired RoboCasa panel (Nasiriany et al., 2024): 192 physical-state clusters with five policy-noise replicates each, for 960 rollouts across 24 tasks. Every method receives the same physical states and policy-noise contract. Transfer uses a 300-rollout paired RoboTwin 2.0 panel (Chen et al., 2025) and a 320-rollout RoboCasa panel with Cosmos Policy (Kim et al., 2026). Clustered bootstrap intervals are the primary inferential quantity; exact paired McNemar summaries are in Appendix E.

5.1 DOES RESYNC IMPROVE THE POLICY?

On the primary panel, native X-WAM succeeds on 80.31% of rollouts. ReSync raises success to 84.79%, a +4.48 pp change with 82 rescues and 39 breaks (95% cluster-bootstrap CI [+2.17,+6.74]). The gain is not carried by one task family: 17 of 24 RoboCasa tasks improve, four are effectively unchanged, and three decrease. Positive deltas appear under language, lighting, robot placement, camera, and background perturbations. ReSync also composes with candidate allocation, reaching 85.94% at $4.15\times$ compute.

The main question is therefore not whether the intervention can help, but whether its advantage is specific to *where* the extra forwards are spent.

5.2 WOULD THE SAME COMPUTE HELP ANYWHERE ELSE?

We spend the same 34 shared-DiT forwards in five obvious ways. **wait-only** executes the extra forwards but restores both streams; **matched resampling** buys additional candidate trajectories; **Action-34** spends the budget on action denoising; **Sync-34** uses a common 34-step schedule for both streams; and **world warm-start** advances the world for 24 forwards before the native ten joint updates. We also include **Sync-50** as the literal fully-synchronized schedule. It costs $5.00\times$ and is intentionally over-budget.

Table 2: **Where the compute goes matters.** Every $3.40\times$ row uses 34 shared-DiT forwards; Sync-50 is shown as a higher-cost reference. Confidence intervals are clustered over physical-state clusters.

Method	SR	Δpp	Rescue/Break	95% CI	Compute
BASE	80.31	0.00	–	–	1.00×
allocation $r25, N4$	82.71	+2.40	64/41	[+0.18,+4.61]	1.75×
wait-only	80.52	+0.21	43/41	[-1.84,+2.19]	3.40×
matched resampling	82.08	+1.77	66/49	[-0.27,+3.83]	3.40×
Action-34	81.56	+1.25	50/38	[-0.84,+3.35]	3.40×
Sync-34	82.29	+1.98	57/38	[-0.10,+4.05]	3.40×
world warm-start	82.92	+2.60	61/36	[+0.31,+4.88]	3.40×
ReSync	**84.79**	**+4.48**	82/39	**[+2.17,+6.74]**	3.40×
Sync-50	83.23	+2.92	68/40	[+0.71,+5.13]	5.00×
ReSync + allocation	85.94	+5.63	95/41	[+3.17,+8.12]	4.15×

The equal-budget alternatives top out at 82.92%, 1.87 pp below ReSync. Relative to matched resampling, the paired difference is +2.71 pp (95% CI [+0.86,+4.57]); relative to world warm-start, it is +1.87 pp ([+0.09,+3.68]). The controls separate several superficially similar ideas. Action-34 shows that extra action refinement is not enough. Warm-start shows that a sharper world *before* joint denoising is not enough either: the world is moved ahead while the action is still near its initial noisy state, creating a relative timestep pairing unlike the native asynchronous coupling. ReSync waits until the action has progressed, advances the world inside the supported joint region, and still leaves three native action updates to absorb that evidence.

Sync-50 is the most direct way to remove the gap: let the action denoise for as long as the world does. It costs $5.00\times$ native compute and still finishes 1.56 pp below ReSync at $3.40\times$. Removing the gap

is not the same as spending computation where the gap says it is useful. The future has to become informative while the action is in a state that can still use it.

5.3 WHY IS THERE A USEFUL RANGE AT ALL?

If ReSync worked simply because more world denoising is better, success should keep rising with M. It does not. The sweep climbs while advancement remains inside joint training support, reaches 84.79% at $M = 24$, and then turns down.

Table 3: **World-side compute has an interior optimum.** $M = 24$ is the largest supported advancement at $j^\star = 6$; $M = 32$ and 40 deliberately cross that boundary.

M	0	8	16	**24**	32	40
SR (%)	80.31	81.67	83.33	**84.79**	83.96	82.29
Δpp	0.00	+1.35	+3.02	**+4.48**	+3.65	+1.98

The full span makes the shape clear. By $M = 40$ the schedule-progress proxy reaches 1.00 ($\hat{r}_{\text{sched}} = 1.00$), yet success is 2.50 pp below the $M = 24$ point. The turning point is the same joint-support boundary obtained from the native schedule before this sweep. The first step beyond it is already lower: $M = 24$ exceeds $M = 32$ by +0.83 pp (95% CI [+0.11,+1.57]). Fully resolving the world is therefore not the objective; resolving useful evidence while the world/action pair remains supported is.

Placement creates the other side of the window. Success remains stable through $t_i = 7$ and drops sharply at $t_i = 8, 9$, where only two or one native action updates remain. This coarse scan fixes $R_{\min} = 3$ once. The complete placement–advancement surface then shows a broad high-return region inside the two constraints: the schedule-selected $(6, 24)$ point reaches 84.79%, while the best observed safe cell reaches 85.10%. The point lands on the same high-performing plateau without being selected from that surface.

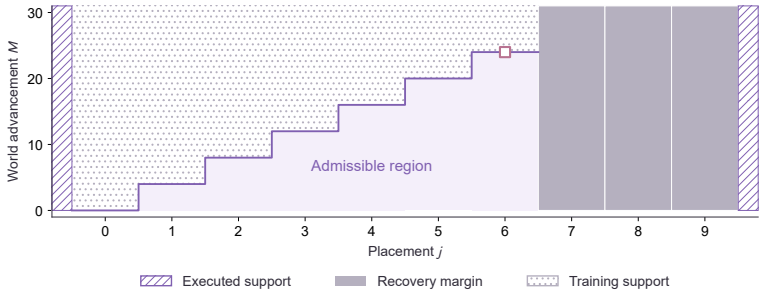

Figure 4: **Schedule constraints carve out the high-return region.** Shading marks support constraints rather than success. The schedule-selected $(6, 24)$ point lies on the same high-performing plateau as the best safe cells.

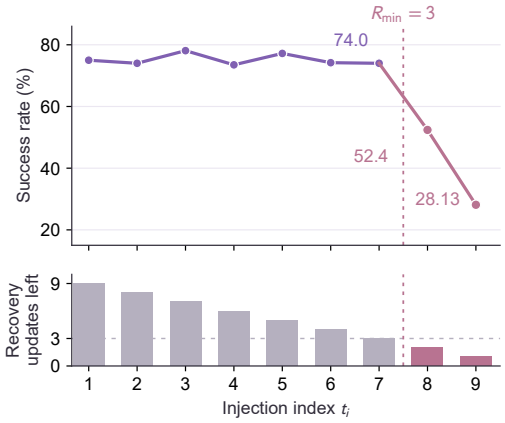

Figure 5: **Late placement loses the recovery margin.** Performance collapses when only one or two native action updates remain.

The compute–quality tradeoff is explicit: one policy decision takes 2.68 s natively and 8.97 s at $M = 24$ on one H800. These are policy-decision intervals, not low-level servo periods, and applications that value faster replanning can move left on the same curve rather than changing the rule. Appendix D gives the measurement protocol and the full latency table.

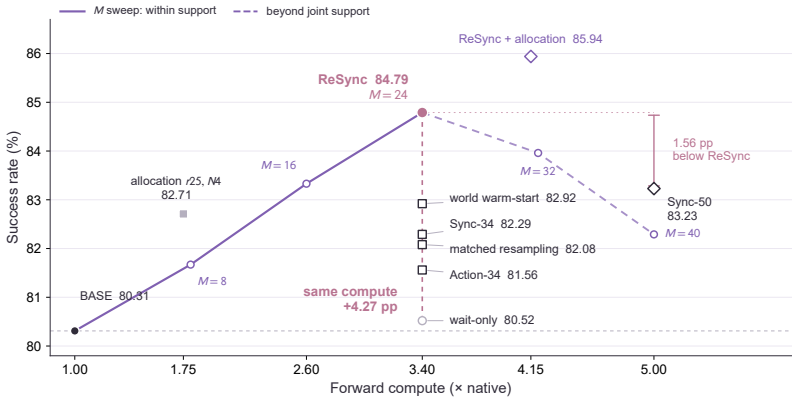

Figure 6: **Evidence advancement improves until the supported boundary, then gives the gain back.** Candidate allocation saturates earlier, while ReSync peaks at the interior $M = 24$ point.

5.4 Does the rule work on schedules it has not seen?

The rule was fixed on the native 10/50 schedule. We now apply it unchanged to two X-WAM schedule pairs that played no part in deriving it, using the same frozen 960-rollout RoboCasa panel. With $R_{\min} = 3$, the 8/40 pair gives the latest safe cache $j^\star = 4$ and the native timestep inequality gives $M^\star = 16$; the 12/60 pair gives $j^\star = 8$ and $M^\star = 32$. We record those operating points before evaluating either local sweep.

Table 4: **Frozen-rule validation on two unseen schedule pairs.** For each pair we evaluate the rule-selected point, its $M \pm 8$ neighbors, and an equal-budget world warm-start; the $M + 8$ point deliberately crosses the schedule-derived support boundary. CIs are cluster-bootstrap intervals for the selected point relative to that schedule's BASE.

Schedule (J, T_O)	Rule $(j^\star, M^\star)$	BASE	$M^\star-8$ SR	Rule SR	$M^\star+8$ SR	Warm-start SR	Rule Δpp	95% CI
$(8, 40)$	$(4, 16)$	79.58	82.29	**83.65**	83.02	82.19	**+4.06**	[+1.72,+6.35]
$(12, 60)$	$(8, 32)$	80.94	84.06	**85.21**	84.58	83.65	**+4.27**	[+1.96,+6.61]

In both schedules the frozen rule lands at the highest of the three tested ReSync advancement points, and the equal-budget warm-start remains lower. The 8/40 rule point uses $1 + 16/8 = 3.00\times$ native forward compute; the 12/60 point uses $1 + 32/12 = 3.67\times$. The same rule that explains the native 10/50 schedule therefore selects a high-return operating point before either new local sweep is revealed.

5.5 Does a fixed rule transfer?

RoboTwin uses the same X-WAM asynchronous schedule, so the operating point remains $(6, 24)$. Cosmos changes both backbone and training construction. We therefore transfer the action-recovery convention rather than X-WAM's training-support inequality: the constructed 14/35 interface fixes $(j_C, M_C) = (10, 10)$ before evaluation. The recovery margin and action-update count remain unchanged.

On RoboTwin, success rises from 90.33% to 93.00% (+2.67 pp; 95% CI [+0.24,+5.06]). The 60-cluster estimate gives $\kappa = -0.079$ with an interval spanning zero, so both RoboTwin and native RoboCasa fall in the practically non-identifiable regime. Across benchmarks we use G to identify that regime, while κ remains a panel-specific estimate. RoboTwin also starts from a much higher baseline and therefore has less absolute headroom.

Table 5: **The same timing rule transfers across benchmark and backbone.** RoboTwin reuses the native X-WAM schedule; Cosmos uses the fixed constructed 14/35 interface, with BASE and ReSync executing the same 14 action updates.

Backbone	Benchmark	G	κ	$(j^\star, M)$	BASE	ReSync	Δpp
X-WAM	RoboCasa	0.80	0.0187	$(6, 24)$	80.31	84.79	+4.48
X-WAM	RoboTwin 2.0	0.80	$-0.079^{\ddagger}$	$(6, 24)$	90.33	93.00	+2.67
Cosmos Policy	RoboCasa	0.60	0.082	$(10, 10)$	66.25	70.63	+4.38

‡ Matched 60-cluster subset; its uncertainty interval spans zero.

On Cosmos, the constructed 14/35 interface gives $G = 0.60$ and $\kappa = 0.082$. Its own released sampler supplies the 35-step future-state trajectory; on our constructed 14-update action interface, the same three-update recovery convention places the cache at $j_C = 10$, and we fix $M_C = 10$ world-only forwards before evaluation. ReSync raises success from 66.25% to 70.63% (+4.38 pp; 95% CI [+1.06,+7.72]) while keeping the same 14 action updates. The forward-equivalent cost is $1 + 10/14 = 1.71\times$.

GeoBoN, the closest candidate-allocation reference on both backbones, reports single-rollout RoboCasa baselines of 80.8% for X-WAM and 66.3% for Cosmos Policy, with roughly two-point gains at $N = 8$ (Zhao et al., 2026b). Those public numbers anchor the scale of our paired baselines; the transfer result is that the same schedule rule continues to improve the policy when the benchmark or backbone changes.

As a physical feasibility check, we run the same inference change on Mobile ALOHA (Fu et al., 2024). BASE and ReSync share one Mobile-ALOHA-tuned X-WAM checkpoint and the same observation/action wrapper; ReSync receives no additional fine-tuning. Across 10 matched resets on each of cup placement, block placement, and cluttered reaching, BASE succeeds on 19/30 trials (63.3%) and ReSync on 22/30 (73.3%). Appendix B.2 reports task-wise counts and the fixed-episode visualization protocol.

6 Analysis

The results separate diagnosis, empirical regularity, and intervention. The commitment–evidence gap measures how far world evidence lags at commitment, the controlled schedule family shows where candidate selection gives way to evidence advancement, and ReSync places world computation where the action can still respond to it.

Under a matched budget, where the computation lands matters more than how much of it there is. A world warm-start sharpens evidence before the joint pass, but the action is then too noisy to absorb it at a supported joint state. Removing the gap is not the same as spending computation where the gap says it is worth spending.

The M sweep and the placement scan close the two sides of the same window: more world progress helps only up to the support boundary, and evidence that arrives with one or two action updates left comes too late to change the decision. World prediction has value only when it becomes available during the portion of the action trajectory that can still use it.

The held-out schedules turn that explanation into a prospective check, since the frozen rule picks operating points on two unseen schedule pairs that beat both their neighbors and their equal-budget controls. Reconfiguring the sampling schedule is not equivalent: a shorter world schedule gives up generative refinement and a longer one lets commitment outrun evidence, whereas ReSync keeps the native trajectory and moves only the supported portion of it.

7 Conclusion

Asynchronous world–action models do not have a single test-time compute budget; they have two clocks. The commitment–evidence gap diagnoses when the action is running ahead of its world evidence, the controlled schedule family reveals an empirical crossover in how extra compute should be spent, and ReSync turns that structure into an inference rule with no new parameters. The amount of extra computation is not the decisive variable. **World prediction is valuable when it becomes available while the action can still change.**

REPRODUCIBILITY STATEMENT

Evaluation uses frozen RoboCasa (960), RoboTwin (300), and Cosmos (320) rollout panels, plus 30 matched physical-robot trials per condition. The 8/40 and 12/60 held-out schedule operating points were recorded from the frozen rule before their local sweeps were evaluated. Appendix E specifies inference; Appendix D gives schedules, pseudocode, and controls. The supplementary video shows the mechanism and paired robot and simulator executions.

ETHICS STATEMENT

The work studies robot-manipulation inference and does not use human-subject data. Physical-robot policy rollouts are executed in a bounded laboratory workspace under operator supervision; operators provide resets and safety intervention but do not teleoperate the evaluated trajectories.

AI USE STATEMENT

AI tools were used only for language polishing.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, et al. π_0: A vision-language-action flow model for general robot control. *arXiv:2410.24164*, 2024.

Anthony Brohan et al. RT-1: Robotics transformer for real-world control at scale. *arXiv:2212.06817*, 2022.

Anthony Brohan et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv:2307.15818*, 2023.

Chi-Lam Cheang et al. GR-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv:2410.06158*, 2024.

Kewei Chen, Yayu Long, and Mingsheng Shang. Drift is a sampling error: SNR-aware power distributions for long-horizon robotic planning. *arXiv:2605.09537*, 2026.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv:2506.18088*, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *arXiv:2302.00111*, 2023.

Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile ALOHA: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4D world action modeling from video priors with asynchronous denoising. *arXiv:2604.26694*, 2026.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv:2412.14803*, 2024.

Wen Huang, Haoran Sun, Yongjian Guo, Yunxuan Ma, Haoran Li, Jing Long, Zhouying Mo, Zhong Guan, Yucheng Guo, Shuai Di, and Junwu Xiong. NoiseGate: Learning per-latent timestep schedules as information gating in world action models. *arXiv:2605.07794*, 2026.

Yunfan Jiang et al. VIMA: General robot manipulation with multimodal prompts. *arXiv:2210.03094*, 2022.

Alexander Khazatsky et al. DROID: A large-scale in-the-wild robot manipulation dataset. *arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations (ICLR)*, 2026. URL `https://research.nvidia.com/labs/cosmos-lab/cosmos-policy/`.

Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. RoboMonkey: Scaling test-time sampling and verification for vision-language-action models. *arXiv:2506.17811*, 2025.

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

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. In *Robotics: Science and Systems (RSS)*, 2025.

Xi Lin, Kangyi Wu, Jiayi Li, Jiaqiao Tang, Qingrong He, and Lin Zhao. BAT-Nav: Belief-based arbitration and termination via remaining discoverability in multi-goal semantic navigation. *arXiv:2605.16932*, 2026.

Ajay Mandlekar et al. MimicGen: A data generation system for scalable robot learning using human demonstrations. *arXiv:2310.17596*, 2023.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems (RSS)*, 2024.

Open X-Embodiment Collaboration et al. Open X-Embodiment: Robotic learning datasets and RT-X models. *arXiv:2310.08864*, 2023.

Zizheng Pan, Bohan Zhuang, De-An Huang, Weili Nie, Zhiding Yu, Chaowei Xiao, Jianfei Cai, and Anima Anandkumar. T-Stitch: Accelerating sampling in pre-trained diffusion models with trajectory stitching. *arXiv:2402.14167*, 2024.

Yirui Sun, Guangyu Zhuge, Keliang Liu, Jie Gu, Shiqin Dai, Xinyu Bing, Zhongxue Gan, and Chunxu Tian. SANTS: A state-adaptive scheduler for world action models. *arXiv:2605.27947*, 2026.

Siying Wang, Kangye Ji, Di Wang, and Fei Cheng. Evolving cache schedules for fast diffusion policy inference. *arXiv:2607.20293*, 2026.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations (ICLR)*, 2024.

Kangyi Wu, Pengna Li, Kailin Lyu, Xi Lin, Lin Zhao, Qingrong He, Jinjun Wang, and Jianyi Liu. Dual-anchoring: Addressing state drift in vision-language navigation. *arXiv:2604.17473*, 2026.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, et al. DreamVLA: A vision-language-action model dreamed with comprehensive world knowledge. *arXiv:2507.04447*, 2025.

Chen Zhao, Zhuoran Wang, Haoyang Li, Shifeng Bao, Guanlin Li, Youhe Feng, Yang Li, Jie Tang, and Jing Zhang. Action draft and verify: A self-verifying framework for vision-language-action model. *arXiv:2603.18091*, 2026a.

Zesen Zhao, Minkyoung Cho, Hui Shen, Boyuan Zheng, Kunxiao Gao, Yulong Cao, and Z. Morley Mao. Test-time scaling for world action models via zero-shot geometric evaluation. *arXiv:2607.17454*, 2026b.

Appendix

A Allocation Controls and Positioning

Table 6: **The neighboring methods act on different decision axes.** The final column describes the evidence state assumed or modeled at commitment.

Method	Allocated object	Signal	Primary axis	Evidence at commit
RoboMonkey	candidates	VLM verifier	candidate	resolved (implicit)
ADV	drafts	self-verification	candidate	resolved (implicit)
GeoBoN	WAM trajectories	geometry	candidate	resolved (implicit)
CAPS	search	SNR	search depth	unmodeled
SANTS	video steps	scheduler	stopping time	explicit noise
NoiseGate	latent timesteps	learned gate	schedule	explicit noise
EVO	cache refresh	schedule search	block×time	unmodeled
T-Stitch	model capacity	fixed switch	timestep	unmodeled
ELASTIC	seq./parallel	meta-policy	allocation	unmodeled
ReSync	**decision timing**	**schedule gap**	**commit time**	**explicit**

SANTS, NoiseGate, EVO, and ELASTIC correspond to adaptive stopping, per-latent schedule gating, cache scheduling, and learned sequential/parallel allocation, respectively (Sun et al., 2026; Huang et al., 2026; Wang et al., 2026; Li et al., 2026).

Table 7: **Allocation operating points on the updated paired panel.**

Method	SR	Δpp	95% CI	Rescue/Break	Compute
BASE	80.31	0.00	–	–	1.00×
allocation $r25, N4$	82.71	+2.40	[+0.18,+4.61]	64/41	1.75×
ReSync + allocation	85.94	+5.63	[+3.17,+8.12]	95/41	4.15×

B Simulation and Real-Robot Executions

B.1 Paired Simulation Trajectories

B.2 Real-Robot Executions

The physical feasibility check uses one Mobile-ALOHA-tuned X-WAM checkpoint for both BASE and ReSync. The same three-RGB/proprioceptive observation wrapper, action-chunk controller, and checkpoint are used in both conditions; ReSync changes only inference timing and receives no additional fine-tuning. We run 10 matched resets on each of three tasks. Cup placement changes from 6/10 to 7/10, block placement from 6/10 to 8/10, and cluttered reaching from 7/10 to 7/10, for an aggregate change from 19/30 (63.3%) to 22/30 (73.3%). The visualization uses fixed episode indices rather than outcome-selected examples.

Table 8: **Physical-robot policy rollouts.** BASE and ReSync share the same checkpoint, resets, and action interface.

Task	BASE	ReSync	Δ
Cup placement	6/10	7/10	+10 pp
Block placement	6/10	8/10	+20 pp
Cluttered reaching	7/10	7/10	0 pp
Overall	19/30	22/30	+10.0 pp

Figure 10 uses the same fixed episode index for each task rather than choosing examples by outcome. All 30 rollouts per condition are retained in the experiment log.

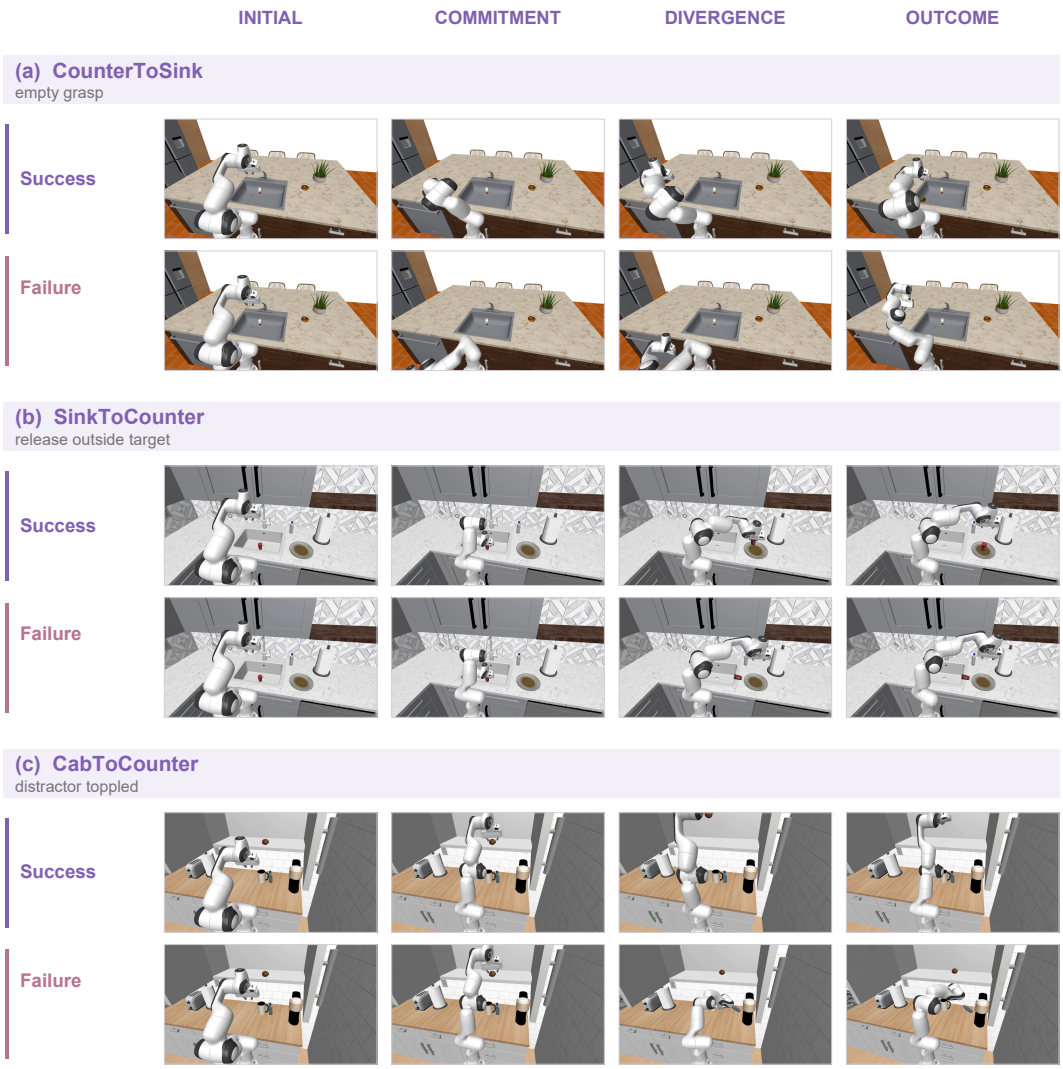

Figure 7: **Paired RoboCasa simulator executions.** Each pair starts from the same reset state and shows successful and failed executions. Frames are aligned at the initial state, gripper closure, divergence, and stable outcome.

B.3 Learned Selector Stress Test

The learned selector screen is separate from the fixed geometric scorer used in Table 1. Here, a learned state–action-residual (SAR) selector averages -2.16 pp across 14 configurations (2/14 positive), while random selection averages -0.24 pp across 12 (5/12 positive). Across matched schedule/depth pairs, random wins 9/12 (one-sided binomial $p = 0.073$). The two selector results answer different questions: the geometric scorer measures whether resolved candidate worlds become rankable under a fixed label-free rule, while this learned screen tests whether a separate trained state–action selector can exploit the same candidate bank.

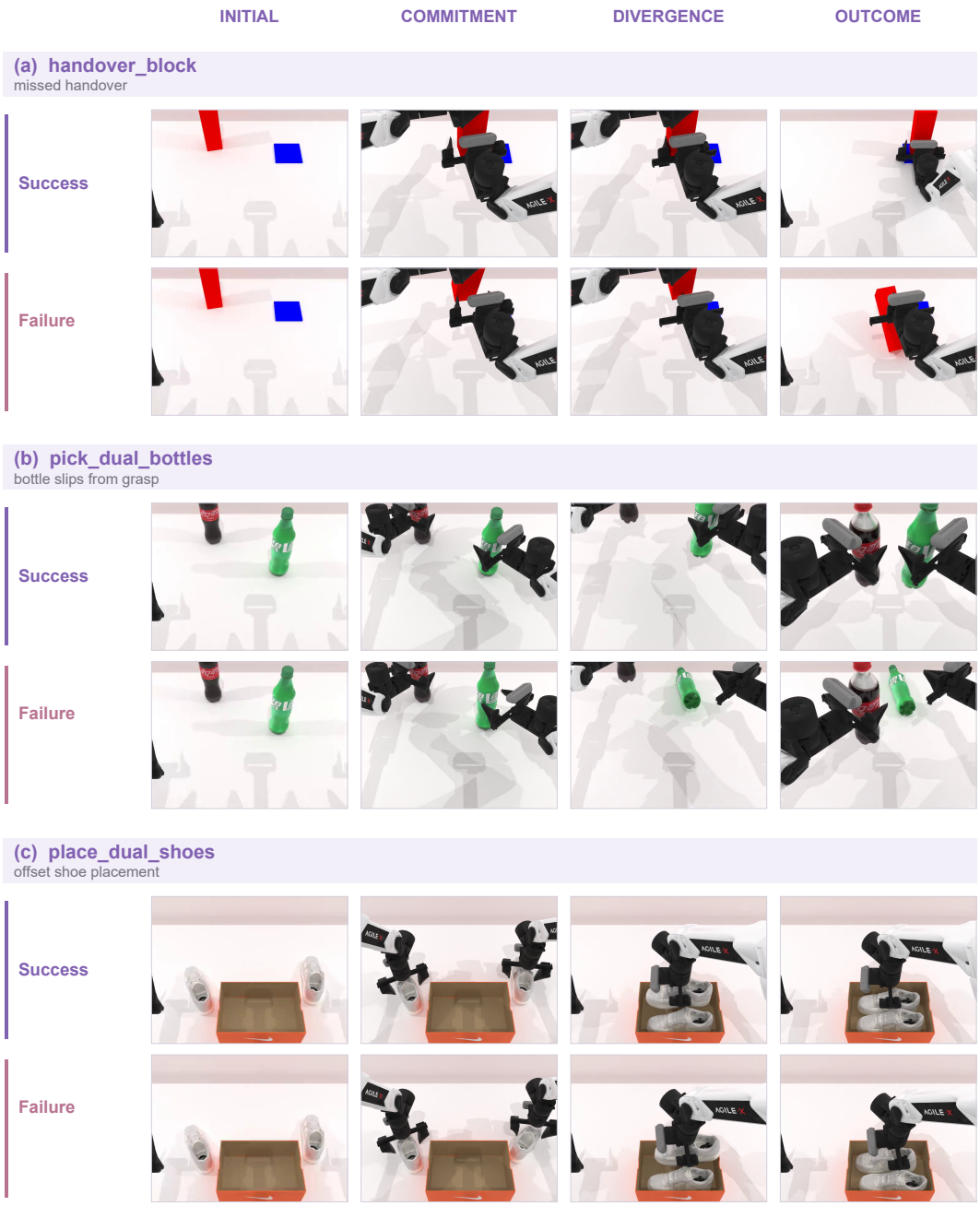

Figure 8: **Paired RoboTwin 2.0 simulator executions.** Each pair shares the initial state and shows successful and failed executions. Frames are aligned at gripper closure and subsequent task events. Source frames retain their native 640×480 resolution.



Figure 9: **Schematic schedule-progress ladder.** A rendered future frame is degraded by controlled downsampling and blur to visualize the intuition behind increasing schedule progress. L0 is the reference frame; L1–L5 are annotated with schedule-progress values 0.20, 0.32, 0.44, 0.56, and 0.68. These panels are an illustration of the proxy and are not model-decoded world states.



Figure 10: **Paired physical-robot policy rollouts.** BASE/ReSync pairs share task and reset conditions and are aligned at commitment and subsequent task events.

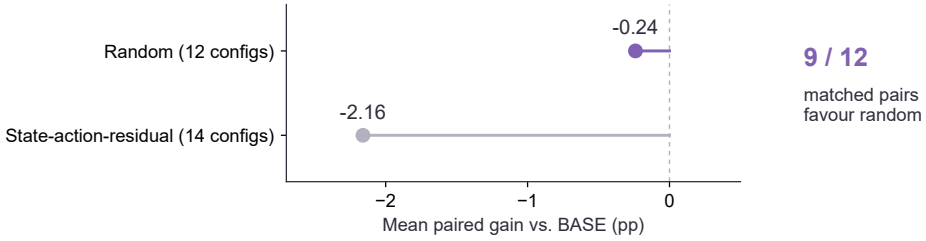

Figure 11: **The selector screen is directional rather than a headline result.** Mean paired deltas are shown for 14 SAR and 12 random-selector configurations. Random selection wins 9 of the 12 matched schedule/depth pairs.

C IN-PROCESS IDENTIFIABILITY AND CANDIDATE REPLACEMENT

C.1 SCHEDULE AND IDENTIFIABILITY DETAILS

Controlled schedule family. To isolate timing from architecture, we construct a controlled schedule family from the same frozen X-WAM checkpoint. The action schedule remains fixed at ten steps. We change only the world schedule length T_O. We refer to these settings by their schedule lengths—SYNC-10, MID-14, MID-18, MID-24, MID-30, and X-WAM (NATIVE)—rather than as different models. The checkpoint, observations, action head, candidate protocol, and evaluation panel are identical across all six rows.

κ protocol. The identifiability screen uses 80 candidate sets covering 22 tasks, with 32 candidates per set (2,560 candidate trajectories). The representation family $\mathcal{R}$ contains four frozen scores (consensus, state–action residual, cross-chunk consistency, and a random control), four learned scores (absolute/relative targets with BCE or pairwise training), visual world-latent scores based on z_O and pooled relative forms, and geometric centroid, nearest-neighbor, and median-distance scores. For each of 10,000 within-set outcome permutations we recompute every score and the familywise maximum AUC; μ_0 is the mean of those maxima. Candidate-set bootstrap intervals use the same set as the resampling unit. Native RoboCasa gives $\kappa = 0.0187$ (familywise $p = 0.3408$, 95% CI $[-0.052, +0.086]$); the matched RoboTwin screen gives $\kappa = -0.079$ ($p = 0.8905$, 95% CI $[-0.188, +0.031]$).

The $N = 1$ oracle value of 53.8% belongs to this separate 80-set candidate-bank screen. The main 80.31% BASE result is measured on the frozen 960-rollout, 24-task RoboCasa policy panel, so the two numbers estimate different quantities under different evaluation protocols.

A falsifiable consequence of unresolved evidence. Consider a masked-evidence channel

$$P_y^r = rP_y^\star + (1 - r)Q,$$

where $P_y^\star$ denotes fully resolved evidence for utility class y and the unresolved component Q is independent of that class. Then

$$\mathrm{TV}(P_1^r, P_0^r) = r\,\mathrm{TV}(P_1^\star, P_0^\star) \le r.$$

Unresolved mass contracts the discrimination available at commitment in direct proportion to the resolved fraction. This identity motivates a directional diagnostic for the real WAM: with candidates and scorer fixed, advancing only the world should increase within-set identifiability, while widening the gap should reduce the value of candidate discrimination.

Candidate-bank and evidence controls. The candidate bank has ample oracle headroom: nested-prefix oracle success rises from 53.8% at $N = 1$ to 77.6% at $N = 32$. Yet native X-WAM commitment exposes almost none of it to a deployment-time selector. Across frozen, learned, visual, and geometric representations, within-set discrimination stays near chance ($\kappa = 0.0187$ on RoboCasa; -0.079 on the matched RoboTwin subset), and replacing one quarter of the candidate pool substantially reorders rankings while barely moving closed-loop success.

The matched world-only intervention in Section 3.2 changes only when the decision reads the world. Varying T_O from the opposite direction produces the same ordering: larger commitment–evidence gaps reduce selector value and increase the return to evidence advancement (Figure 2, Table 1).

Schedule length and native quality. The BASE column reveals the second half of the story. Native success is itself non-monotonic in T_O: aggressively shortening the world schedule closes the gap but sacrifices generation quality, while a long world trajectory preserves prediction capacity but leaves more of it unresolved at commitment. ReSync keeps the rich trajectory and reallocates only its useful inference-time portion before commitment.

Routing-map consistency check. This check audits which routed bins the native sampler actually visits; it is an implementation consistency check, not an independent predictive result. A small frozen-backbone coupling adapter routes with $k(t) = \lfloor (1 - t/1000)\, 10 \rfloor$. The ten executed X-WAM action timesteps map to $[0, 0, 0, 0, 1, 1, 2, 3, 4, 6]$, giving reachable support $\{0, 1, 2, 3, 4, 6\}$ and dead support $\{5, 7, 8, 9\}$ *before training*.

The executed routing map matches 10/10 bins across all three seeds. The non-monotonic hole is a useful implementation fingerprint: $k = 5$ stays at initialization while $k = 6$ learns. The dead-$k5$/live-$k6$ pattern follows directly from the routed timestep map.

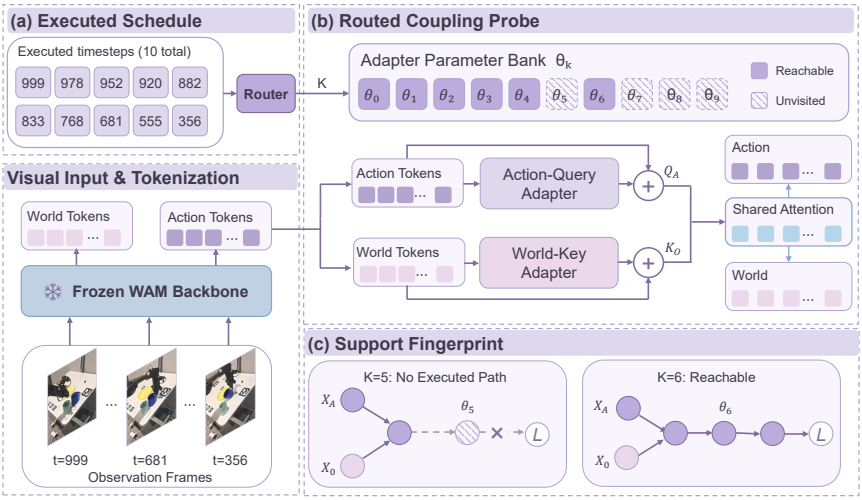

Figure 12: **Routing-map consistency check.** Reachable and unvisited bins follow the executed timestep map across three seeds, including the dead-$k5$/live-$k6$ pattern.

Preservation and intervention. Reachability alone is not sufficient: modifying the reachable bins degrades the competent native path. These controls make train-time coupling modification unattractive and motivate a parameter-free temporal intervention. Full coupling-adapter optimization, native-path success change, variance trajectories, and seedwise controls are in Appendix D.

C.2 Additional identifiability controls

A label-free world-particle geometry screen evaluates six geometric scores at all ten executed action steps. Every within-set AUC remains near chance and the familywise max-statistic gives $p = 1.0$.

Table 9: **In-process geometry remains weak at every executed action position.** Values are within-set AUC.

Metric	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
kNN density	.435	.424	.428	.433	.436	.426	.431	.431	.428	.432
Mahalanobis	.433	.483	.505	.479	.462	.460	.453	.493	.500	.471
KDE	.516	.516	.513	.514	.516	.516	.516	.516	.516	.516
Dist.-center	.489	.489	.486	.486	.486	.486	.489	.486	.486	.486
Local agree.	.482	.491	.485	.497	.488	.491	.488	.482	.483	.494
Full dist.-center	.449	.449	.449	.449	.449	.449	.449	.449	.449	.449

Candidate replacement. Replacing one quarter of the candidate pool reorders 38.7% of cases; 50% replacement reorders 61.4%. The 25% intervention moves success by +0.52 pp, 95% CI $[-1.22, +2.31]$, with OR ≈ 1.08 and $\phi \approx 0.013$.

Proposition 1 (Masked-evidence contraction)**.** *If unresolved evidence is class-independent,* $P_y^r = rP_y^\star + (1 - r)Q$*, then*

$$\mathrm{TV}(P_1^r, P_0^r) = r\,\mathrm{TV}(P_1^\star, P_0^\star) \le r.$$

The proposition is a compact masked-evidence identity used to motivate the identifiability test in Section 3.2; the empirical question is whether the real WAM follows the same direction when only the world clock is advanced.

D ReSync Implementation, Schedule Rule, and Controls

Implementation details. ReSync is always-on once the schedule-derived operating point is selected. ReSync+allocation gates only residual candidate continuations after the shared ReSync prefix; it does not alter the ReSync placement or world-advance rule.

Index convention. The schedule-rule index j denotes the last completed action update; the placement-scan index $t_i = j + 1$ denotes the next action update. With $J = 10$, the remaining budget is $R = J - 1 - j = 10 - t_i$, including the update at t_i. Therefore $j^* = 6$ corresponds to $t_i = 7$ and leaves three updates, whereas $j = 8$ leaves one.

Cosmos convention. Cosmos uses a different training construction from X-WAM, so its transfer test is defined on the constructed inference interface itself. We use a 14-update action stream against the 35-step future-state schedule and carry over the same recovery convention: $J_C - 1 - j_C \geq R_{\min} = 3$, giving $j_C = 10$. The interface fixes $M_C = 10$ world-only forwards before evaluation. ReSync caches after action update 10, performs those ten world-only forwards while restoring the cached action state, and completes native action updates 11–13. BASE and ReSync both execute the same 14 action updates, and compute is $1 + 10/14 = 1.71\times$.

Native schedule and ψ. The runner uses independent FlowUniPC schedules with 50 world and ten action steps and time-shift 5. During the first ten joint iterations it indexes world and action schedules with the same loop index, so $\psi(j) = j$. The action timesteps are

$$[999, 978, 952, 920, 882, 833, 768, 681, 555, 356].$$

The 50-step world schedule is

$$[999, 995, 991, 987, 982, 978, 973, 968, 963, 957, 952, 946, 940, 934, 927, 920, 913, 906, 898, 890, 882, 873, 863, 854, 843, 833, 821, 809, 796, 783, 768, 753, 737, 720, 701, 681, 660, 636, 611, 584, 555, 522, 487, 448, 405, 356, 302, 241, 172, 92].$$

Following X-WAM's joint-timestep sampling (Guo et al., 2026) and the released runner cited in Section 4, we define the represented joint noisy regime by $t_O \geq t_a$; hence $M_{\max}(j) = \max\{m : t_O^{\psi(j)+m} \geq t_a^j\}$.

Table 10: **The schedule makes the X-WAM operating point directly auditable.** $R_{\min} = 3$ removes $j = 7, 8$ and leaves $j^\star = 6$ as the latest safe maximizer.

j	t_a^j	$\psi(j)$	$t_O^{\psi(j)}$	$M_{\max}$	recovery
0	999	0	999	0	9
1	978	1	995	4	8
2	952	2	991	8	7
3	920	3	987	12	6
4	882	4	982	16	5
5	833	5	978	20	4
6	**768**	**6**	**973**	**24**	**3**
7	681	7	968	28	2
8	555	8	963	32	1

Frozen-rule schedule pairs. Applying the same scheduler construction and the already fixed $R_{\min} = 3$ to two unused schedule pairs gives $(J, T_O) = (8, 40) \Rightarrow j^\star = 4,\ M^\star = 16$ and $(12, 60) \Rightarrow j^\star = 8,\ M^\star = 32$. These operating points were recorded before the corresponding neighbor and warm-start evaluations in Table 4; no value from those sweeps enters the rule.

The compute–quality tradeoff is explicit. On one H800, native X-WAM takes 2.68 s per policy decision (0.37 Hz), $M = 8$ takes 4.74 s (0.21 Hz), and $M = 24$ takes 8.97 s (0.11 Hz). These are policy-decision intervals, not low-level servo periods. The released RoboCasa evaluator blocks on the policy-server response and then executes the returned 32-action chunk before issuing the next query.[2]

[2]See `https://github.com/sharinka0715/X-WAM/blob/main/evaluation/robocasa_client.py`, where the client waits on `socket.recv()` and uses `action_length=32`.

Applications that value faster replanning can move left on the same curve rather than changing the rule itself. Appendix D reports the full latency

Latency and overshoot. Wall-clock growth is slightly sub-linear relative to forward-equivalent compute because batching, model dispatch, and compiled-graph overhead are partly fixed. The $M = 32, 40$ rows intentionally exceed the $M_{\max}(6) = 24$ training-support boundary and expose the descending side of the interior optimum.

Table 11: **World advancement peaks at the training-supported boundary.** Wall-clock measurements use one NVIDIA H800 80GB GPU with identical batching and compilation state; $M > 24$ deliberately crosses joint training support.

M	Schedule progress proxy	SR	Δpp	Compute	Latency / decision	Effective Hz
0	0.20	80.31	0.00	1.00×	2.68 s	0.37
8	0.36	81.67	+1.35	1.80×	4.74 s	0.21
16	0.52	83.33	+3.02	2.60×	6.83 s	0.15
24	0.68	84.79	+4.48	3.40×	8.97 s	0.11
32	0.84	83.96	+3.65	4.20×	11.14 s	0.09
40	1.00	82.29	+1.98	5.00×	13.32 s	0.08

Preservation control. The coupling-adapter support test separates preservation from reachability. Uniform capacity yields native-path success change $(-0.83, -1.07, -2.29)$ pp across seeds; reachable-only and dead-only closed-loop reallocations yield $(-0.46, -0.71, -1.18)$ pp and $(+0.04, -0.03, +0.02)$ pp respectively. The distinct metrics are kept separate because they answer different questions: whether the native path is preserved, and whether train-time reallocation remains a viable policy intervention.

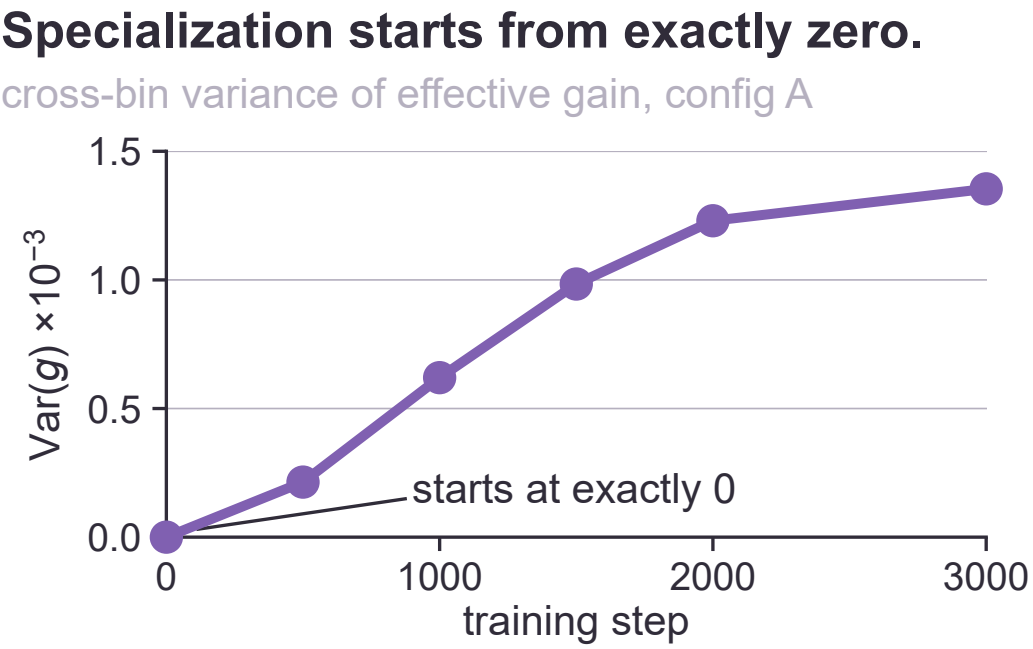

Figure 13: **Specialization is learned, not inherited.** Cross-bin variance of the effective gain during the coupling-adapter support test. It is zero at initialization and rises monotonically; the unvisited bins remain at 0.50.

E Reproducibility and Statistics

The evaluation grid, selector, trigger masks, seed contract, paired estimands, cluster key, and tests are fixed before evaluating the 960-case panel. The panel is 24 tasks × eight state IDs × five rollout IDs. Cluster bootstrap operates on 192 physical states; exact McNemar uses paired rescue/break counts. The RoboTwin ReSync panel is 60 clusters × five replicates.

Schedule-shaping contrasts. A random-placement schedule is the direct reference for early-, late-, and decayed-placement variants. Cluster-bootstrap CI is the primary inference unit; case-wise exact McNemar is secondary. Table 12 reports the four contrasts.

Table 12: **Schedule shaping changes point estimates, not clustered inference.** McNemar is a secondary diagnostic.

Contrast	Δpp	cluster-bootstrap 95% CI	McNemar p
random placement vs BASE	+0.94	[-1.16,+3.07]	0.39
early vs random	+0.28	[-1.51,+2.11]	0.76
late vs random	-0.86	[-2.84,+1.13]	0.35
decay vs random	+1.31	[-0.58,+3.22]	0.17

Historical protocols. Published X-WAM reports 79.2%; an early 24-task$\times$40-episode reproduction gives 79.79%; the frozen clustered paired baseline used here is 80.31%. These are separate protocols and are not merged.

F Evidence Lineage

The experiment records retain the frozen 960-case panel, case identifiers, task/state/rollout mapping, candidate seeds, candidate-replacement controls, coupling-adapter checks, placement scans, selector screens, ReSync ablations, and the pre-evaluation operating-point records for the 8/40 and 12/60 schedule pairs. Quantitative tables are generated from the corresponding frozen evaluation records. The supplementary package contains the narrated video and a short README.

Training-support extension. ReSync's ceiling follows joint timestep support represented during training. Widening that support directly expands the admissible evidence-advancement region.