PACER: TRAINING-FREE PHASE-SCHEDULED MEMORY FOR LONG-HORIZON VISION-LANGUAGE-ACTION MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Frozen vision-language-action (VLA) policies re-encode every observation from scratch, so across a chain of subgoals they can lose track of the object they are acting on, yet fine-tuning such a policy for every deployment costs data, compute, and validation. We show that a frozen policy's own prefix key–value (KV) state can serve as memory. PACER is a training-free framework that maintains this state across replans as working memory and a capacity-one event snapshot, coordinated by a proprioceptive phase controller and periodic refresh. On LIBERO-Long with the frozen $\pi_{0.5}$ policy, PACER raises three-seed direct success from 92.3% to 95.5% (97.3% with an optional instruction-level router), including 91/150 to 149/150 on a repeated-object task, while median inference latency rises by only 0.6%. Both memories contribute to the success gain. On five physical-robot tasks, direct deployment improves aggregate success from 60% to 86% over 250 trials per condition.

1 INTRODUCTION

Generalist vision-language-action (VLA) policies now execute a broad range of language-conditioned manipulation skills from raw observations (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025). Fine-tuning a large policy for every workspace demands additional data, compute, and behavioral validation (Octo Model Team et al., 2024; Kim et al., 2024), motivating deployment with frozen weights. Long-horizon benchmarks chain several subgoals, and the policy must keep acting on the right object from one phase to the next, for example reaching for the second of two identical pots once the first is placed (Liu et al., 2023; Cherepanov et al., 2026; Zhou et al., 2026). In the frozen $\pi_{0.5}$ baseline studied here, each replan re-encodes a single current observation and discards its prefix KV state. Carrying that state forward offers a way to support consistent grounding without updating the policy.

Existing remedies follow two broad routes. The first learns memory with the policy (Shi et al., 2025; Fang et al., 2025; Cherepanov et al., 2026; Zeng et al., 2026b;a; Yang et al., 2026a; Koo et al., 2026). The second reuses internal key–value (KV) state at inference time without additional training (Xu et al., 2025; Yang et al., 2025b; Sun et al., 2026). PACER builds on this second route and trains neither the policy nor an event detector.

As a design intuition, execution alternates between moving (approach or transport) and acting (grasp or placement) phases, which proprioception can signal without learned parameters, since gripper changes and inter-replan end-effector displacement are inexpensive control cues. We present **PACER** (Phase-Adaptive Cache and Event-triggered Refresh), which uses these cues to coordinate four operations: rebuilding, partially updating, selecting a readout, and injecting stored prefix KV. A **working memory** maintains current grounding across replans; an **event memory** retains a capacity-one snapshot for action-head or backbone conditioning. An optional instruction-level router selects the memory route or the exact baseline path (Figure 1).

On LIBERO-Long, PACER reaches 95.5% directly and 97.3% with Router M, versus 92.3% for the frozen $\pi_{0.5}$ baseline (Table 1), with nearly unchanged median inference latency (+0.6%). The repeated-object task improves from 91/150 to 149/150 on each of the three seeds; direct application does not improve every task, and we report the regressions alongside the gains. Both memories

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.

contribute. Working memory lifts that task to 145/150, and event-snapshot conditioning lifts it to 149/150, raises a relative-placement task from 133/150 to 147/150 on every seed, and supplies 26 of the 48 additional direct successes. Quota-matched refresh-timing, injection-timing and snapshot-content variants all remain at 96–98/100, so neither memory needs exact in-quota alignment or historical snapshot content; readout choice is the main sensitivity (Section 4). On five physical-robot tasks, direct deployment without routing raises aggregate success from 60% to 86%.

Our contributions are: **(1) PACER prefix-state memory**, combining working-bank maintenance and snapshot conditioning; **(2) proprioceptive memory control**, an explicit implementation coordinating refresh, readout, and injection alongside a periodic channel; and **(3) an empirical account of the gain**: separating direct from routed deployment, we show that prefix-state maintenance and snapshot conditioning both contribute to the improvement, which changes little with in-quota timing or snapshot content yet benefits from phase-triggered refresh, identify readout as the main sensitivity, and validate direct deployment on five physical-robot tasks.

2 Related Work

Memory for VLA manipulation. Learning temporal state into a policy is a natural response to partial observability. MemoryVLA studies a perceptual-cognitive memory architecture, while KEMO detects event keyframes from kinematic and visual cues and EventVLA learns which frames to store (Shi et al., 2025; Zeng et al., 2026a; Yang et al., 2026a). HAMLET summarizes visual-language representations into learned moment tokens and aggregates them through history-aware fine-tuning (Koo et al., 2026). PACER addresses the complementary setting of adding memory to an already frozen policy. This distinction concerns the memory intervention only, since our physical-robot base policy was itself fine-tuned before being frozen.

KV reuse and test-time intervention. VLA-Cache and EfficientVLA study reuse and computational efficiency in VLA inference (Xu et al., 2025; Yang et al., 2025b); UAOR is a related training-free observation intervention (Yang et al., 2026b). VLA-Cache selectively reuses static visual-token KV, recomputes task-relevant tokens, and adapts reuse across layers, with faster inference as its goal. TempoFit is also training-free but builds a different memory (Sun et al., 2026): at every timestep it writes layer-wise prefix KV into FIFO buffers, retrieves entries by K-to-K similarity with a recency bias, and adds them back through residual conditioning. PACER keeps no per-layer history buffer and performs no retrieval: it maintains the current prefix state, re-encoding only changed tokens, and stores one full-prefix snapshot that is replaced only at phase events. We compare with TempoFit by the gain over each method's own baseline (Appendix L).

Event boundaries and phase-aware manipulation. Event boundaries have inspired ways to organize memory (Zacks et al., 2007; Zheng et al., 2022), while keyframes and temporal segmentation provide related structure in manipulation learning (Shridhar et al., 2022; Goyal et al., 2023; Zhang et al., 2023b). Long-VLA partitions manipulation into moving and interaction phases, learns a phase identifier with the policy on decomposed demonstrations, and uses phase-aware visual masking (Fan et al., 2025). PACER turns these event-boundary and phase ideas into online prefix-KV maintenance for a frozen policy, using deterministic proprioceptive cues and periodic refresh without additional training.

3 Method

3.1 Setup and Preliminaries

A frozen VLA policy π_θ operates in a closed replan loop. At replan t, the instruction ℓ and multi-view observation o_t are encoded into prefix tokens z_t, whose transformer key–value states are

$$\mathrm{KV}_t = F_\theta(z_t), \tag{1}$$

and the action head generates

$$a_t \sim p_\theta(\cdot \mid \mathrm{KV}_t, q_t), \tag{2}$$

where q_t is proprioception and a_t is an action chunk. The robot executes a prefix of this chunk before observing and replanning (Zhao et al., 2023b; Chi et al., 2024). The baseline recomputes a

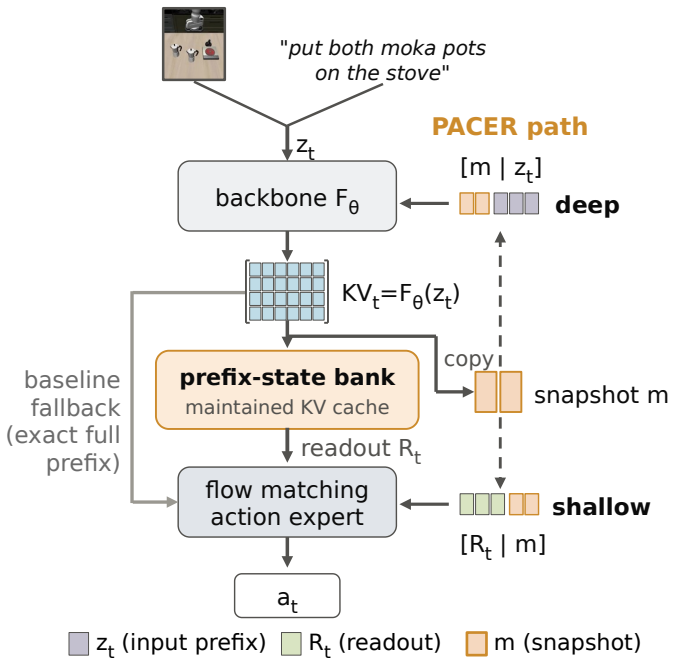


Figure 1: PACER's inference paths. A working bank supplies the action-head readout. Snapshot KV enters at the action head (shallow) or backbone (deep). The optional router selects PACER or the original full-prefix baseline; direct evaluation applies PACER to all tasks.

full forward at every replan and discards KV_t afterward. Evidence e observed at a replan t_0 may be absent at a later replan $t' > t_0$ when an instruction refers to it. Maintaining inference state across replans is thus a training-free intervention with θ fixed.

We distinguish a *working bank* B_t, a per-step *readout* R_t, and a stored *snapshot* m_t. The bank stores selected current or reused KV; the readout determines what the action head sees; the snapshot is copied from a full-prefix computation and retained until replaced. Even without a snapshot the bank carries state across steps; the snapshot re-exposes an earlier scene anchor.

PACER replaces the memoryless rebuild with a maintenance operator,

$$(B_t, R_t, m_t) = M(B_{t-1}, m_{t-1}, z_t, \sigma_t, h_t), \tag{3}$$

where σ_t is the execution-phase label and h_t collects the embedding history and controller state. M composes four primitives: *rebuild* the bank from a full forward, *partial update* of changed tokens, *mask* to expose a selected readout, and *inject* stored snapshot KV. Working memory combines the first three; event memory adds the snapshot path. These choices trade task success, per-step latency, and the live-token KV budget (Appendix I.1).

3.2 Phase-Adaptive Cache and Event-Triggered Refresh

Define the raw acting signal, which uses proprioception and no learned parameters,

$$b_t = \mathbb{1}[|g_t - g_{t-1}| > \tau_g \ \vee \ \|x_t - x_{t-1}\|_2 < \tau_x], \tag{4}$$

using gripper state g_t, end-effector position x_t, and thresholds τ_g and τ_x for gripper change and inter-replan displacement. A *moving* label denotes the default state and *acting* the signal-activated state; these are controller labels, not contact annotations. Parameter values are given in Appendix C.

The phase controller enters acting immediately when $b_t = 1$, holds it for at least d replans, and requests refresh on entry, on exit and at periodic checks while acting, so a controller refresh can occur without a phase transition (Appendix C; Figure 3 shows one rollout).

Independently, a periodic channel refreshes every p replans to limit drift ($p = 2$, written p2; p1 refreshes every replan). A full-prefix forward is used at initialization or when either channel requests

Algorithm 1 Deep configuration: execution order. Algorithm 2 gives the full conditions and settings.

1: Once per episode, optionally route the instruction; fallback runs the exact baseline for the whole episode.
2: At each replan, embed the inputs; initialize empty state on the first observation, otherwise update the phase controller and periodic refresh request.
3: At initialization or refresh, run a full-prefix forward and rebuild the bank; select L, S initially and reselect S on controller requests.
4: Otherwise, partially forward changed tokens, patch their bank entries, and retain out-of-bank changes as temporary extra KV.
5: Choose full-prefix readout at refresh or initialization; otherwise select the phase-dependent bank view plus temporary extra KV.
6: At eligible later reads, deep-condition the readout using the stored snapshot before any replacement, leaving the bank unchanged.
7: At eligible writes, replace the snapshot from unconditioned full-prefix KV; initialization then applies the same-observation snapshot.
8: Denoise actions, execute the action chunk prefix, and update embedding history.

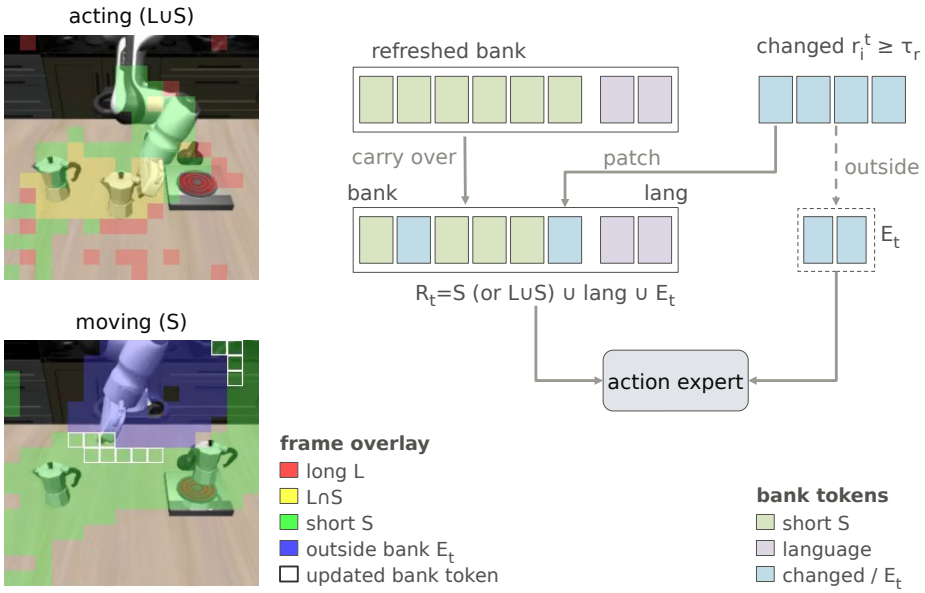


Figure 2: Selected-bank maintenance and non-refresh readout. Left: token selection on two frames of a Task 8 rollout; colors mark L, S, their overlap and changed tokens outside the bank (E_t), and white outlines mark bank tokens updated at that replan. Right: between refreshes the bank carries over unchanged entries, patches changed ones ($r_i^t \geq \tau_r$) and passes changed tokens outside it to the readout; moving reads S, acting $L \cup S$, and refresh steps use the full prefix.

refresh, and the action head then reads the full prefix. Controller-triggered refresh also reselects the short-term token set; a purely periodic refresh retains its indices. Algorithm 1 summarizes the deployed ordering; Algorithm 2 in Appendix C gives the full conditions.

3.3 State-Maintenance Mechanisms

3.3.1 Token Selection and Partial KV Updates

The bank keeps language-token slots with their padding mask and two base-camera index sets (Figure 2). The fixed set $L = \{i : \bar{A}_i \geq \tau(\bar{A})\}$ is selected once at initialization from language-to-image attention $\bar{A}_i$ with an Otsu threshold τ (Otsu, 1979). The tracking set $S_t = \{i : \tilde{s}_i^t \geq Q_{\alpha_t}(\tilde{s}^t)\}$ ranks image tokens by their smoothed similarity $\tilde{s}_i^t$ to the language prefix and is reselected on

controller refreshes, so it follows the target as the scene changes. Between refreshes, a change score r_i^t (the largest normalized embedding distance of position i from recent observations) selects $U_t = \{i : r_i^t \geq \tau_r\}$; changed positions outside the bank membership $\mathcal{B}_t = L \cup S_t \cup \mathcal{L}$, where $\mathcal{L}$ holds the language positions, form $E_t = U_t \setminus \mathcal{B}_t$ and join the current readout without being stored. The bank evolves as

$$B_t = \begin{cases} F_\theta(z_t)[\mathcal{B}_t], & \text{initialization or refresh,} \\ \mathrm{Patch}(B_{t-1}, z_t[U_t]), & \text{otherwise,} \end{cases} \tag{5}$$

where Patch re-encodes only the changed tokens through the backbone against the bank KV, keeps their original prefix positions (Su et al., 2024), and writes their new keys and values back into the bank, while unchanged entries are carried over, so the bank holds a stabilized present. The readout decides what the action head sees:

$$R_t = \begin{cases} B_t[S_t \cup \mathcal{L}] \,\|\, \mathrm{KV}_t^E, & \text{moving,} \\ B_t[L \cup S_t \cup \mathcal{L}] \,\|\, \mathrm{KV}_t^E, & \text{acting,} \end{cases} \tag{6}$$

where KV_t^E is the temporary changed-token KV and $\|$ denotes concatenation. Moving exposes the tracking view S_t; acting also exposes the fixed scene positions in L; refresh steps use the full prefix. We write a readout as moving→acting, so Eq. (6) is $S \to L \cup S$. Thresholds, change detection and the partial-forward computation are given in Appendix C.

3.3.2 Snapshot Conditioning

The working bank exposes only selected tokens, so the snapshot adds a full-prefix view of the scene taken at a phase event. The snapshot m stores the valid base-camera and language KV of one full-prefix computation; it is written at initialization, replaced on entry to acting, and retained while the bank keeps changing. Conditioning on it improves both the repeated-object and relative-placement tasks (Section 4.3), and the benefit does not require past content (Section 4.6). Shallow conditioning appends the snapshot to the action-head context,

$$a_t \sim p_\theta(\cdot \mid R_t \| m, q_t), \tag{7}$$

and deep conditioning uses it as backbone past KV in one extra full-prefix forward,

$$\widehat{\mathrm{KV}}_t = F_\theta(z_t \mid m), \tag{8}$$

so current tokens are encoded jointly with the stored scene before the action head reads them; the persistent bank is not overwritten. The deep configuration reads at phase changes in either direction before replacing the snapshot, and initialization writes and applies the first snapshot (Algorithm 2; positional alignment in Appendix C).

Cost. Partial encoding and selected readout offset the extra computation of carrying and conditioning on retained KV; median inference latency rises by only 0.6% (Section 4.7), and Appendix I.1 gives a cost decomposition.

4 Experiments

4.1 Evaluation Setup

We study a frozen $\pi_{0.5}$ baseline implemented in PyTorch on LIBERO-Long's ten tasks (Physical Intelligence et al., 2025; Liu et al., 2023), with action horizon 10, replan interval 5, and 50 episodes per task for each seed in $\{7, 42, 123\}$. Policy weights remain frozen. Direct application to all ten tasks is our primary measure. We also report an optional instruction-level router as a deployment configuration. It enables PACER for multi-object placements with a shared goal (Router A), and additionally for multi-target placements (Router M). Other instructions run the baseline (Section 4.7).

All configurations share this protocol and one complete baseline series whose 30 task–seed result files were verified (Appendix A); seeds 42 and 123 are also reported on their own (Section 4.2). The timing and content studies in Sections 4.4 and 4.6 additionally use protocols frozen before evaluation, with matched initial states.

Table 1: LIBERO-Long results with the frozen $\pi_{0.5}$ policy, three seeds. Suite columns give success (%) over 1,500 episodes; routed columns apply PACER where the router enables it and the baseline elsewhere. Task columns give successes out of 150. L1 retains all base-camera and language KV between periodic refreshes without patching bank entries; L2 uses a selected bank with partial updates and phase-dependent readout; PACER is L2 plus deep snapshot conditioning. Changing p2 to p1 changes refresh and full-prefix denoising, not injection to every replan. The shallow and deep rows differ in schedule and initialization as well as depth.

Configuration	Direct	Router A	Router M	Task 8	Task 6
$\pi_{0.5}$ (baseline)	92.3	92.3	92.3	91/150	137/150
L1: periodic continuity	90.9	96.1	93.9	144/150	118/150
L1 without stale base-camera readout	90.0	96.3	94.8	145/150	124/150
L2: selected bank, no snapshot	93.8	96.4	95.9	145/150	133/150
L2 + shallow snapshot	94.2	96.5	96.8	146/150	145/150
$\pi_{0.5}$ + PACER (L2 + deep snapshot)	95.5	96.7	97.3	149/150	147/150
PACER, refresh every replan (p1)	92.4	92.6	92.5	92/150	136/150

4.2 Direct and Routed Results

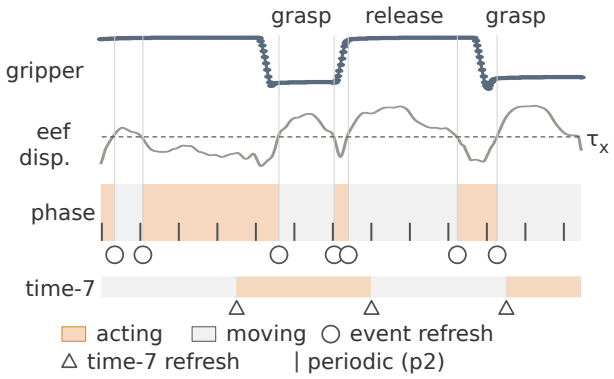
Direct PACER reaches 1433/1500 versus 1385/1500, a gain of 48 successes (3.20 percentage points). Router M reaches 1459/1500 (97.3%) and Router A 1451/1500 (Table 1); two-sided Fisher tests give $p \approx 10^{-9}$ for Router M over the suite and $p \approx 10^{-19}$ on Task 8 (Appendix H); Router M's additional 26 successes are exactly the direct losses on the four instructions without multi-object or multi-target dependencies, on which it runs the baseline (Tasks 2, 3, 5 and 9: -8, -1, -3 and -14), so routing adds safety, not a stronger memory effect. Measured against each method's own baseline, routed PACER gives the largest reported LIBERO-Long gain among training-free methods we know of (+5.0 points, removing 65% of baseline failures, versus +4.0 for TempoFit and +2.0 for UAOR); direct PACER's +3.2 points is below TempoFit's. Each figure follows its method's own protocol; neither method has released code, and only our routed row uses instruction-level gating (Appendix L).

The gain concentrates where the baseline fails. Tasks 1, 3, 4 and 7 start at 147–149/150 and leave little room for improvement. Among the multi-object and multi-target tasks, only Tasks 0, 6 and 8 have at least 13 baseline failures, and PACER recovers failures on all three: 6 of 13 on Task 0, 10 of 13 on Task 6 and 58 of 59 on Task 8. On Task 6, a relative-placement task, the gain comes from snapshot conditioning (Section 4.3). Task 8 ("put both moka pots on the stove") accounts for 59 of the baseline's 115 failures; PACER raises it from 91/150 to 149/150, improving every seed (31→50, 28→49 and 32→50 out of 50), and it contributes 58 of Router M's 74 additional successes (78.4%). All five physical tasks also improve (Section 4.8). Without Task 8, direct PACER reaches 1284/1350 against the baseline's 1294/1350, and Router M reaches 1310/1350. The two largest direct losses, on Task 9 (137 to 123) and Task 2 (146 to 138), occur on instructions without multi-object or multi-target dependencies; additional runs reproduce both (Appendix G). Task 4, a relative-placement instruction, drops from 147 to 145.

On seeds 42 and 123 alone, direct PACER and Router M still lead the baseline, 949 and 967 versus 920 out of 1,000 (Appendix G).

4.3 Prefix-State Continuity

What explains the repeated-object improvement? Working memory explains most of the repeated-object gain, and event-snapshot conditioning adds to it. Periodic continuity already reaches 144/150 on Task 8, and removing its stale base-camera readout still gives 145/150 (Table 1). Direct access to that old visual block is therefore not what drives the improvement. The latter still encodes current tokens against maintained state, so it is not history-free. A mid-episode intervention confirms that the second-half benefit does not depend on memory stored earlier in the episode (Section 4.6). Shallow and deep conditioning raise Task 6 from 133/150 to 145/150 and 147/150, improving every seed, and deep conditioning raises Task 8 from 145/150 to 149/150 and supplies 26 of PACER's 48



	Deployed (deep)	$L \cup S$ readout (no snapshot)
Task 8	143 / **149**	143 / **145**
Task 6	146 / **147**	129 / **133**
Direct	1413 / **1433**	1381 / **1406**
Router M	1448 / **1459**	1433 / **1434**

Figure 3: Phase-triggered versus time-7 refresh. Left: one Task 8 rollout; gripper state (high: open) and end-effector displacement (threshold τ_x) set the moving/acting labels, circles mark controller refreshes and ticks mark p2. Time-7 refreshes every seven replans (triangles) and, in the deployed configuration, also flips the labels that drive readout, snapshot writes and deep reads. Right: three-seed successes, time-7 / phase-triggered, for the deployed configuration ($S \to L \cup S$, deep) and the $L \cup S \to L \cup S$ readout (tasks out of 150, totals out of 1,500).

additional direct successes (Table 1). Appendix J shows six diagnostic rollouts, and Section 4.6 tests whether the snapshot must hold past content.

Refreshing every replan (p1) returns Task 8 to 92/150, consistent with the maintained state needing to persist; p1 changes refresh and full-prefix denoising cadence, not injection to every step.

4.4 Refresh Schedule and Timing

We first compare phase-triggered refresh with a fixed schedule at a matched refresh count. The time-7 control refreshes every seven replans, matching the mean controller refresh interval measured on Task 8 (about 6.7 replans); both schedules keep p2. Phase-triggered refresh is ahead in every comparison (Figure 3; Table 10, Appendix E): with the same $L \cup S \to L \cup S$ readout, direct success is 1406 versus 1381/1500, and on the deployed configuration, clock-driven moving/acting labels lower it from 1433 to 1413/1500, behind on every seed.

Within a matched quota, exact timing is not critical. On Task 8 with the same $L \cup S \to L \cup S$ readout and no snapshots, one full refresh per two-replan window placed at the first phase change, at the window end or at a random slot gives 97, 96 and 98/100; with the snapshot fixed at frame 0, one injection per seven-replan window placed the same three ways gives 97, 97 and 98/100 (Table 25, Appendices O and Q). All six stay far above the main-table baseline's 91/150 on Task 8.

4.5 Readout and Snapshot Conditioning

Injection gains depend on the readout. With the main readout, shallow and deep conditioning raise Task 6 from 133/150 to 145/150 and 147/150 on every seed; the readout $L \cup S \to \mathrm{res}$, whose acting view is language KV plus current residual-output KV and which also uses maintained state, already reaches 150/150 without a snapshot (Table 16, Appendix F). Readout choice therefore sets how much conditioning adds and is the main configuration sensitivity. Writing the snapshot on entry to acting gives the best tested write schedule: 1433/1500 direct and 296/300 on Tasks 6+8, versus 1419 when writing on completion (ACT) and 1415 with the write/read direction reversed (STA; Appendix F). All seven snapshot configurations in Table 17 exceed the baseline in direct success (1404–1433 versus 1385/1500; Appendix G), so the gain does not hinge on one configuration.

Initialization versus later snapshot use. With PACER's all-base snapshot selector, complete snapshot use gives 149/150 on Task 8, 147/150 on Task 6 and 1433/1500 direct, versus 147/150, 143/150 and 1417/1500 with frame-0-only conditioning, so later event snapshots add to what same-scene initialization provides (Appendix F); Section 4.6 examines their content.

Table 2: Memory-content studies, 100 episodes per condition (two seeds $\times$ 50 matched initial states). (a) Historical (H) versus current-frame (C) snapshot content; prespecified Holm-adjusted $p = 1.0$. (b) Task 8 continued from the state after the first pot is placed: PACER keeping its memory, with the memory cleared or replaced by another episode's memory, and the baseline. Differences are in percentage points with 95% initial-state-cluster bootstrap intervals. Protocols are in Appendices P and R.

(a) Snapshot content

Task	H / C	H−C (pp)	95% CI
Task 8	98 / 98	0	$[-4, +4]$
Task 6	98 / 96	$+2$	$[-2, +6]$

(b) Mid-episode intervention, Task 8

Condition	Success	Keep− (pp)	95% CI
Keep memory	96/100		
Clear memory	93/100	$+3$	$[-2, +9]$
Swap memory	97/100	-1	$[-5, +2]$
Baseline	79/100	$+17$	$[+9, +26]$

4.6 What the Memory Carries

We ask whether replacing historical snapshots (H) by current-frame content (C) costs performance on Tasks 8 and 6, holding scheduling, readout, deep conditioning and maintenance rules fixed. At each application, C reads KV from an unconditioned encoding of the current observation at the snapshot indices; H performs the same preparation forward but retains the historical payload. Neither overwrites the bank or stored snapshot (Appendix P).

Current-frame content leaves Task 8 unchanged at 98/100 and Task 6 at 96/100 against 98/100 (Table 2), so the conditioning benefit comes from the snapshot pathway and does not require past content. Both conditions keep the same working state and deep conditioning, so the comparison isolates snapshot content (Appendix P).

Observed failures in the content comparison. The ten failures are prolonged or repeated moka-pot grasps on Task 8 and incomplete transfers or one toppled mug on Task 6; none is a wrong-target error (Appendix P.1).

Mid-episode intervention. Content aside, does the maintained state carry information from earlier in the episode? On Task 8 we run PACER until the first pot rests on the stove, save the simulator, random-number and memory state, and continue the same episode from that state in four ways: keeping the memory, clearing it, replacing it with the memory another episode had at the same point, or switching to the baseline (Appendix R). After this point PACER still adds 17 successes over the baseline (Table 2b). Clearing or swapping the memory stays within three successes of keeping it, so the second-half benefit comes from state PACER rebuilds after the intervention rather than from information stored before it; no condition lifts the placed pot again.

4.7 Routing and Inference Cost

The router is a rule-based classifier designed from PACER's mechanism. It enables memory only for instructions carrying the dependencies PACER targets, namely multi-object placements with a shared goal (repeated-instance continuity) or multi-target placements, and abstains otherwise, for example on appliance grounding or a single terminal closure. Read once per episode and free of task identifiers, it runs exactly the baseline on abstention. In a dry audit of 1,061 instructions it also activates on matching structures in other LIBERO suites and CALVIN, so it does not depend on LIBERO-Long task identities. Under 50 meaning-preserving paraphrases of the ten instructions it keeps all 20 abstentions and 27 of 30 activations, and the three changes are conservative abstentions. Rules and per-task reasons are listed in Appendix B. All direct results and the physical study run without it.

In a same-machine profile over Tasks 6/7/8, median inference latency is 253.0 ms for the baseline and 254.5 ms for deep PACER (+0.6%); mean latency rises by 4.3% and peak GPU memory by 0.6% (7.15 to 7.19 GiB). Per-configuration timing and live-token counts are in Appendix I.

Table 3: Five-task physical evaluation: 50 cumulative episodes per condition and task, including the initial ten. Both conditions use the same frozen Piper-adapted policy and unchanged setup; PACER runs without routing.

Task	$\pi_{0.5}$ (baseline)	$\pi_{0.5}$ + PACER
Second drawer	21/50	31/50
Two yellow blocks	24/50	44/50
Pen into cup	16/50	45/50
Stack three bowls	44/50	49/50
First and third bowls to the middle	45/50	46/50
Total	150/250	215/250

4.8 Real-Robot Experiments

Direct deployment without routing raises aggregate physical-robot success from 60% to 86% over 250 trials per condition (Table 3). We evaluate a Piper 6-DoF arm with fixed head/base and wrist cameras. The $\pi_{0.5}$ base was fine-tuned on seven in-house tasks (236 demonstrations, approximately 134k frames), then frozen for baseline and PACER. PACER adds no training, and all five tasks use the $S \to L \cup S$ readout, deep conditioning and a gripper-only phase signal (Appendix M).

PACER improves success on every task; the largest gains are pen transfer (16 to 45) and two yellow blocks (24 to 44). Figure 4 shows a baseline block dropping after an insecure grasp (left) and wrong-color selections (middle and right), whereas PACER completes the target placements. Success criteria are in Appendix M.

5 Discussion and Conclusion

Scope and limitations. The equal-quota timing and content studies show that exact alignment and past snapshot content are not required. The main simulation study uses LIBERO-Long with $\pi_{0.5}$; adapted to OpenVLA-OFT, PACER also improves the two memory-dependent LIBERO-Mem tasks evaluated so far, from 8/50 to 16/50 and from 10/50 to 15/50, with the remaining tasks in progress (Appendix S). The physical study covers five tasks on one platform.

Conclusion. PACER turns a frozen VLA policy's prefix KV state into training-free working and event memory, raising direct LIBERO-Long success from 92.3% to 95.5%; both memories, refreshed at phase changes, carry the gain.

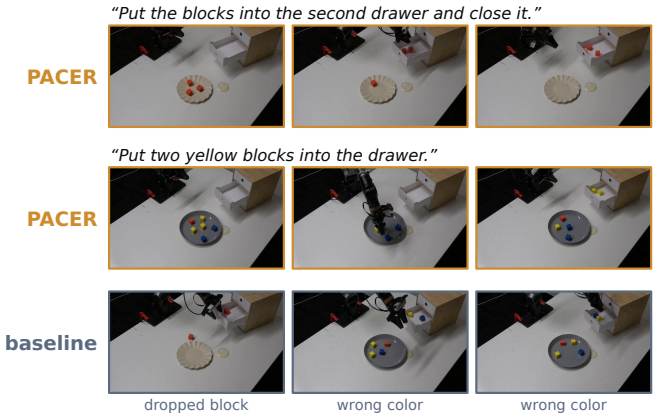


Figure 4: Drawer-task examples. Cumulative success over 50 trials per condition rises from 21/50 to 31/50 for the second drawer and from 24/50 to 44/50 for two yellow blocks.

LLM USAGE STATEMENT

Large language model (LLM) assistants were used under the authors' direction for four purposes; for each we state what the authors did and what the assistants did. *Writing*: the authors drafted the related work (Section 2) and wrote the method (Section 3), the experiments (Section 4) and the corresponding appendices; the assistants aided and polished the writing, including rewording, grammar and consistency of terminology. *Retrieval and discovery*: the assistants searched for and summarized related work; the authors read every cited work before citing it. *Research ideation and execution*: the authors designed PACER and all experiments and made every methodological decision; the assistants discussed experimental designs and interpretations of results, implemented evaluation and analysis scripts, launched and monitored evaluation runs, and tabulated results under the authors' instructions. *Drafting*: the assistants drafted the remaining parts of the paper from the authors' related-work, method and experiment sections; the authors revised and approved all text. The authors checked every experiment, script output and reported number against the underlying evaluation records and take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix C specify the method and implementation, including attention masks, positional alignment and the phase controller. Appendix A describes the evaluation protocol and the verified main-table result files, and Appendices O, P and Q report the controlled studies, which were run under frozen protocols with recorded source, checkpoint and initial-state hashes. Appendix N lists the software and hardware environment. An anonymized code supplement with evaluation entry points and selected simulation and physical-robot rollout videos, with and without overlays, accompany the submission, and we will release the code and evaluation configurations upon publication.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control, 2024. URL https://arxiv.org/abs/2410.24164.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token Merging: Your ViT But Faster, 2023. URL https://arxiv.org/abs/2210.09461.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control, 2023. URL https://arxiv.org/abs/2307.15818.

Hao Chen, Jiaming Liu, Zhonghao Yan, Nuowei Han, Renrui Zhang, Chenyang Gu, Jialin Gao, Ziyu Guo, Siyuan Qian, Yinxi Wang, Peng Jia, Shanghang Zhang, and Pheng-Ann Heng. LaST-R1: Reinforcing Robotic Manipulation via Adaptive Physical Latent Reasoning, 2026. URL https://arxiv.org/abs/2604.28192.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online RL Fine-tuning for Flow-based Vision-Language-Action Models, 2025. URL https://arxiv.org/abs/2510.25889.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An Image is Worth 1/2 Tokens After Layer 2: Plug-and-Play Inference Acceleration for Large Vision-Language Models, 2024. URL https://arxiv.org/abs/2403.06764.

Jintao Cheng, Weibin Li, Haozhe Wang, Gang Wang, Yipu Zhang, Xiaoyu Tang, Jin Wu, Xieyuanli Chen, Yunhui Liu, and Wei Zhang. Training-Free Interaction-Aligned Visual Token Pruning for Efficient Embodied Manipulation, 2026. URL https://arxiv.org/abs/2603.22991.

Egor Cherepanov, Nikita Kachaev, Daniil Zelezetsky, Aydar Bulatov, Artem Pshenitsyn, Yuri Kuratov, Alexey Skrynnik, Aleksandr I. Panov, and Alexey K. Kovalev. μVLA: On Recurrent Memory for Partially Observable Manipulation in VLA Models, 2026. URL https://arxiv.org/abs/2606.12497.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. RoboMME: Benchmarking and Understanding Memory for Robotic Generalist Policies, 2026. URL https://arxiv.org/abs/2603.04639.

Bradley Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1): 1–26, 1979. doi: 10.1214/aos/1176344552.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.

Yiguo Fan, Shuanghao Bai, Xinyang Tong, Pengxiang Ding, Yuyang Zhu, Hongchao Lu, Fengqi Dai, Wei Zhao, Yang Liu, Siteng Huang, Zhaoxin Fan, Badong Chen, and Donglin Wang. Long-VLA: Unleashing Long-Horizon Capability of Vision Language Action Model for Robot Manipulation. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2018–2037, 2025. URL https://proceedings.mlr.press/v305/fan25a.html.

Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. SAM2Act: Integrating Visual Foundation Model with A Memory Architecture for Robotic Manipulation, 2025. URL https://arxiv.org/abs/2501.18564.

Ronald A. Fisher. On the Interpretation of χ^2 from Contingency Tables, and the Calculation of P. *Journal of the Royal Statistical Society*, 85(1):87–94, 1922. doi: 10.2307/2340521.

Ronald A. Fisher. *The Design of Experiments*. Oliver and Boyd, Edinburgh, 1935.

Yonatan Geifman and Ran El-Yaniv. Selective Classification for Deep Neural Networks, 2017. URL https://arxiv.org/abs/1705.08500.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt Cache: Modular Attention Reuse for Low-Latency Inference, 2024. URL https://arxiv.org/abs/2311.04934.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic View Transformer for 3D Object Manipulation, 2023. URL https://arxiv.org/abs/2306.14896.

Sture Holm. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success, 2025. URL https://arxiv.org/abs/2502.19645.

George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot Learning from Demonstration by Constructing Skill Trees. *The International Journal of Robotics Research*, 31 (3):360–375, 2012. doi: 10.1177/0278364911428653.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. HAMLET: Switch Your Vision-Language-Action Model into a History-Aware Policy. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=KcJ9U0x6kO.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. SimpleVLA-RL: Scaling VLA Training via Reinforcement Learning, 2025a. URL https://arxiv.org/abs/2509.09674.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-Time Intervention: Eliciting Truthful Answers from a Language Model, 2024a. URL https://arxiv.org/abs/2306.03341.

Runhao Li, Wenkai Guo, Zhenyu Wu, Changyuan Wang, Haoyuan Deng, Zhenyu Weng, Yap-Peng Tan, and Ziwei Wang. MAP-VLA: Memory-Augmented Prompting for Vision-Language-Action Model in Robotic Manipulation, 2025b. URL https://arxiv.org/abs/2511.09516.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM Knows What You are Looking for Before Generation, 2024b. URL https://arxiv.org/abs/2404.14469.

LIBERO-Mem. LIBERO-Mem: memory-dependent sequential manipulation tasks. https://github.com/libero-mem/libero-mem, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning, 2023. URL https://arxiv.org/abs/2306.03310.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022. doi: 10.1109/LRA.2022.3180108.

Sebastian Michelmann, Uri Hasson, and Kenneth A. Norman. Evidence That Event Boundaries Are Access Points for Memory Retrieval. *Psychological Science*, 34(3):326–344, 2023. doi: 10.1177/09567976221128206.

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*, 2024.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards Stable Test-Time Adaptation in Dynamic Wild World, 2023. URL https://arxiv.org/abs/2302.12400.

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

Nobuyuki Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979. doi: 10.1109/TSMC.1979.4310076.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering Llama 2 via Contrastive Activation Addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-Cognitive Memory in Vision-Language-Action Models for Robotic Manipulation, 2025. URL https://arxiv.org/abs/2508.19236.

Lucy Xiaoyang Shi, Archit Sharma, Tony Z. Zhao, and Chelsea Finn. Waypoint-Based Imitation Learning for Robotic Manipulation, 2023. URL https://arxiv.org/abs/2307.14326.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-Actor: A Multi-Task Transformer for Robotic Manipulation, 2022. URL https://arxiv.org/abs/2209.05451.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Jun Sun, Boyu Yang, Jiahao Zhang, Ning Ma, Chencheng Wu, Siqing Zhang, Yiou Huang, Qiufeng Wang, Shan Liang, and Yaran Chen. TempoFit: Plug-and-Play Layer-Wise Temporal KV Memory for Long-Horizon Vision-Language-Action Manipulation, 2026. URL https://arxiv.org/abs/2603.07647.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. Tent: Fully Test-Time Adaptation by Entropy Minimization. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Kuancheng Wang, Seungho Yeom, Jinglin Cao, Yuheng Zhi, Nikhil Shinde, and Michael Yip. Remember what you did?: Learning Behavioral Memories for Partially Observable Object Manipulation, 2026. URL https://arxiv.org/abs/2606.21188.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks, 2024. URL https://arxiv.org/abs/2309.17453.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-Cache: Efficient Vision-Language-Action Manipulation via Adaptive Token Caching, 2025. URL https://arxiv.org/abs/2502.02175.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, Wengang Zhou, Yao Mu, and Tai Wang. EventVLA: Event-Driven Visual Evidence Memory for Long-Horizon Vision-Language-Action Policies, 2026a. URL https://arxiv.org/abs/2606.20092.

Jiabing Yang, Yixiang Chen, Yuan Xu, Peiyan Li, Zichen Wen, Bowen Fang, Tao Yu, Xiangnan Wu, Qisen Ma, Kai Wang, Ziheng He, Yingda Li, Zhengbo Zhang, Jing Liu, Nianfeng Liu, Yan Huang, and Liang Wang. UAOR: Uncertainty-aware Observation Reinjection for Vision-Language-Action Models, 2026b. URL https://arxiv.org/abs/2602.18020.

Siyuan Yang, Yang Zhang, Haoran He, Ling Pan, Xiu Li, Chenjia Bai, and Xuelong Li. Steering Vision-Language-Action Models as Anti-Exploration: A Test-Time Scaling Approach, 2025a. URL https://arxiv.org/abs/2512.02834.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. EfficientVLA: Training-Free Acceleration and Compression for Vision-Language-Action Models, 2025b. URL https://arxiv.org/abs/2506.10100.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. DeeR-VLA: Dynamic Inference of Multimodal Large Language Models for Efficient Robot Execution, 2024. URL https://arxiv.org/abs/2411.02359.

Jeffrey M. Zacks, Nicole K. Speer, Khena M. Swallow, Todd S. Braver, and Jeremy R. Reynolds. Event Perception: A Mind-Brain Perspective. *Psychological Bulletin*, 133(2):273–293, 2007. doi: 10.1037/0033-2909.133.2.273.

Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Peihong Wang, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, Zhihao Liu, Kang Chen, Wenhao Tang, Quanlu Zhang, Weinan Zhang, Chao Yu, and Yu Wang. RLinf-VLA: A Unified and Efficient Framework for Reinforcement Learning of Vision-Language-Action Models, 2025. URL https://arxiv.org/abs/2510.06710.

Yihan Zeng, Minghao Ye, Yiyuan Chen, Yide Shentu, Philipp Wu, Zike Yan, and Zhongyu Li. KEMO: Event-Driven Keyframe Memory for Long-Horizon Robot Manipulation with VLA Policies, 2026a. URL https://arxiv.org/abs/2606.23589.

Zijian Zeng, Fei Ding, Huiming Yang, and Xianwei Li. HELM: Harness-Enhanced Long-horizon Memory for Vision-Language-Action Manipulation, 2026b. URL https://arxiv.org/abs/2604.18791.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. VLABench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. *arXiv preprint arXiv:2412.18194*, 2024.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models, 2023a. URL https://arxiv.org/abs/2306.14048.

Zichen Zhang, Yunshuang Li, Osbert Bastani, Abhishek Gupta, Dinesh Jayaraman, Yecheng Jason Ma, and Luca Weihs. Universal Visual Decomposer: Long-Horizon Manipulation Made Easy, 2023b. URL https://arxiv.org/abs/2310.08581.

Hao Zhao, Yuejiang Liu, Alexandre Alahi, and Tao Lin. On Pitfalls of Test-Time Adaptation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 42058–42080. PMLR, 2023a.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware, 2023b. URL https://arxiv.org/abs/2304.13705.

Jie Zheng, Andrea G. P. Schjetnan, Mar Yebra, Bernard A. Gomes, Clayton P. Mosher, Suneil K. Kalia, Taufik A. Valiante, Adam N. Mamelak, Gabriel Kreiman, and Ueli Rutishauser. Neurons Detect Cognitive Boundaries to Structure Episodic Memories in Humans. *Nature Neuroscience*, 25(3):358–368, 2022. doi: 10.1038/s41593-022-01020-w.

Yulin Zhou, Yimeng Wang, Nengyu Wang, Shaojia Xing, Shiyun Tu, Xiang Li, Jingkai Zhang, Ningbo Jiang, Yuankai Lin, Hua Yang, Xiangrui Zeng, and Zhouping Yin. Chronos: A Physics-Informed Full-History Framework for Non-Markovian Long-Horizon Manipulation, 2026. URL https://arxiv.org/abs/2606.30318.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation Engineering: A Top-Down Approach to AI Transparency, 2023. URL https://arxiv.org/abs/2310.01405.

A BASELINE AND PROTOCOL

The main reported protocol evaluates LIBERO-Long with seeds $\{7, 42, 123\}$ and 50 episodes per task and seed (1,500 episodes per configuration), with action horizon 10 and replan interval 5, on the frozen $\pi_{0.5}$ baseline implemented in PyTorch.

The baseline is the complete official action-horizon-10 series (1385/1500), evaluated with the same horizon and replan settings as all PACER rows; its archived files and extraction hashes were verified.

Source-code semantics are described in Appendix C. Seeds 42 and 123 are also reported on their own (Appendix G).

Table 4: Frozen $\pi_{0.5}$ baseline results, 50 episodes per task and seed. Task indices are zero-based throughout.

Seed	t0	t1	t2	t3	t4	t5	t6	t7	t8	t9	Total
7	46	50	50	49	49	49	45	49	31	47	465/500 (93.0%)
42	46	49	48	50	48	48	45	50	28	46	458/500 (91.6%)
123	45	50	48	48	50	48	47	50	32	44	462/500 (92.4%)
3-seed	137	149	146	147	147	145	137	149	91	137	1385/1500 (92.3%)

B ROUTER: ASSIGNMENT AND SCOPE

The router reads only the instruction, once at the start of each episode, and keeps that route throughout execution; it does not observe execution. Router M recognizes multi-object placements with a shared goal or multi-target placement instruction patterns and enables memory on $\{0, 1, 4, 6, 7, 8\}$. Router A uses only the multi-object set $\{0, 1, 7, 8\}$. The patterns encode memory dependencies rather than individual prompts: the classifier compiles quantifier, relation and action structure into dependency types without task identifiers or rollout state, and records a mechanism-level reason when it abstains. On LIBERO-Long it routes Tasks 0, 1, 7 and 8 as multi-object and Tasks 4 and 6 as multi-target, and abstains on Task 2 (appliance grounding: "turn on the stove and put the moka pot on it"), Tasks 3 and 9 (single terminal closure: "… and close it") and Task 5 (single local placement in a caddy compartment). The assignments used for the paper accounting are in Table 5. On LIBERO-Long, Router M's 26 successes beyond direct application are exactly the direct losses on the four tasks it leaves to the baseline (Tasks 2, 3, 5 and 9). These abstentions follow from instruction structure rather than from task results, and they fall where applying memory is worse than the baseline: additional runs reproduce the losses on Tasks 2 and 9 (Appendix G). The enabled sets were fixed from instruction structure before rollout evaluation, not from per-task success rates.

Table 5: Router assignments on LIBERO-Long. Router M activates the multi-object and multi-target rows (6/10 tasks); Router A activates only the multi-object rows (4/10).

Task	Instruction	Route
0	put both the alphabet soup and the tomato sauce in the basket	Multi-object
1	put both the cream cheese box and the butter in the basket	Multi-object
2	turn on the stove and put the moka pot on it	Abstain
3	put the black bowl in the bottom drawer of the cabinet and close it	Abstain
4	put the white mug on the left plate and put the yellow and white mug on the right plate	Multi-target
5	pick up the book and place it in the back compartment of the caddy	Abstain
6	put the white mug on the plate and put the chocolate pudding to the right of the plate	Multi-target
7	put both the alphabet soup and the cream cheese box in the basket	Multi-object
8	put both moka pots on the stove	Multi-object
9	put the yellow and white mug in the microwave and close it	Abstain

The rules pass a 102-case test suite over 52 prompts. In a dry audit of 1,061 instructions they activate on 43: 6 of the 10 LIBERO-Long instructions, 2 of 120 prompts from the other LIBERO

suites (a multi-target bowl stack), 5 of 7 physical-robot prompts, 1 of 423 CALVIN (Mees et al., 2022) annotations (a stack-blocks prompt), 21 of 491 RLBench (James et al., 2020) task templates and 8 of 10 curated RoboCasa (Nasiriany et al., 2024) and VLABench (Zhang et al., 2024) prompts. They therefore respond to instruction structure rather than to LIBERO-Long task identities. We also wrote five meaning-preserving paraphrases of each LIBERO-Long instruction, varying verbs, word order, politeness and quantifier wording. On these 50 paraphrases the router keeps all 20 abstentions on Tasks 2, 3, 5 and 9 and 27 of 30 activations on the enabled tasks; the three changes are conservative abstentions caused by a fronted destination (Tasks 1 and 7) and swapped clauses that place the relative reference first (Task 6). An actual fallback executes the baseline path; a false-positive memory activation can still degrade performance. The static rule would abstain on the single-object pen prompt, although direct physical deployment improves its observed count from 16/50 to 45/50 (Table 3). PACER's applicability therefore extends beyond the router's coverage: routing is a coverage heuristic, not a test of whether PACER can help.

For every routed table, enabled cells use the selected PACER configuration's count and all other cells use the baseline count. Routed totals are therefore fixed-set accounting rather than a separate online evaluation. This convention is kept separate from direct all-task application.

C Implementation and Configuration

Table 6 lists the main constants. Figure 2 illustrates selected-bank maintenance. The phase controller uses a minimum hold and also requests refresh within acting; refresh requests and phase transitions are separate events; and deep reads occur on both transition directions, whereas subsequent writes occur on entry to acting.

Table 6: Main configuration constants. Quantiles are thresholds, not retained fractions. The minimum hold is not consecutive-label debouncing. Snapshot path differences are in Table 7.

Parameter	Main simulation configuration	Role
Action horizon / replan interval	10 / 5	action chunking and replanning
Periodic refresh interval p	2 replans	full-prefix rebuild of the bank
Gripper-change / displacement thresholds τ_g, τ_x	> 0.001 / < 0.005	trigger the acting label
Acting minimum hold / refresh checks d	2 subsequent replans / every 2 replans	minimum hold and within-acting refresh
Long-set attention layer / retained fraction	4 / clamped to 20–80%	fixed language-relevant indices L
Short-set quantile α_t (acting / moving / initial)	0.7 / 0.6 / 0.7	reselected tracking indices S
Residual threshold τ_r / history K	≥ 0.1 / up to 3 preceding observations	changed tokens for partial update
Spatial smoothing	masked 3×3 average	smooth selection and residual maps
Snapshot capacity	1	event-memory size

Controller. Figure 3 shows the signals on one rollout. The controller reads raw simulator observations, before policy-state normalization: the first gripper joint position and the Cartesian end-effector position. The raw signal uses gripper change > 0.001 or end-effector displacement < 0.005 between replans. Initialization starts in moving. A trigger enters acting immediately and resets an acting-age counter to zero. On each later acting step, increment the counter. At a positive multiple of two, request refresh and exit if the raw signal is absent. After age two, an absent signal at an intervening step also exits and requests refresh. Moving without a trigger issues no controller request. The independent periodic channel refreshes at $t \bmod 2 = 0$. Both channels can coincide. This is not two-consecutive-label debouncing. In the selected readout, moving maps to S and acting to $L \cup S$. The acting label is a control-signal state, not a semantic detection of grasping; the controller reads only these raw proprioceptive signals, never task identity or outcomes.

Selection and residuals. L uses language-to-image attention at layer 4, head/query averaging, per-camera min–max normalization, and Otsu thresholding with the selected fraction clamped to 20–80%. Its indices are fixed after frame 0. S uses cosine similarity between image embeddings and the masked mean of valid language-prefix embeddings, with feature dimensions aligned before comparison, followed by per-camera normalization and masked 3×3 spatial averaging before quantile thresholding. The language pool follows the base policy's tokenization and includes tokenized state when it is supplied in the language prefix. The quantile threshold is 0.7 in acting and 0.6 in moving; these denote quantiles, not selected fractions. Frame 0 uses 0.7. Controller refreshes rese-

lect S; purely periodic refreshes update KV at existing indices. L and S are selected independently and may overlap; the bank keeps their union $L \cup S$ together with the language slots.

For change detection, the current image embedding at each position is compared with the corresponding embedding in each of the last at most three observations. Each cosine-distance map is min–max normalized over valid tokens per camera (denominator clamped to 10^{-6}), then spatially averaged using valid neighbors. The per-token maximum over those history maps is thresholded at 0.1. The history buffer is updated every replan. Language slots and their padding mask are retained in the bank; residual selection acts on image tokens. The residual update thus keeps the maintained bank approximately current: it is a continuity mechanism rather than a store of past observations.

Partial-forward computation and writeback. The selected current embeddings are passed through the backbone with a cloned bank as per-layer past KV. Their RoPE position IDs retain their original cumulative-valid-mask prefix positions. An additive zero attention mask permits every changed-token query to read every bank entry and every other changed token. At every transformer layer, the newly appended KV for patchable bank positions replaces its bank slot. Extra changed entries are concatenated into this step's readout and discarded afterward.

Approximation and masking. Unchanged bank entries stay stale, and both a retained old entry and a new entry can represent the same global position during a partial forward. The helper does not incorporate the bank padding mask into this partial-forward attention mask, even though the bank can retain padded language slots. Padding is retained for the subsequent action-head readout.

Snapshot paths. Both compared configurations share the snapshot content, capacity and source: valid base-camera plus language KV; capacity one, without snapshot decay or norm rescaling; and a copy of the unconditioned full-prefix cache, which deep-conditioned readout KV do not overwrite. At a refresh step the read precedes the write (Algorithm 2). In the shallow path, action-token positions shift from $\nu(R_t) + i$ to $\nu(R_t) + \nu(m) + i$, where $\nu(C)$ counts valid entries, and initialization writes a snapshot without applying it. In the deep path, current tokens keep their natural positions and prefix attention mask with attention to the snapshot added; snapshot keys are RoPE-shifted by the current valid-prefix length and values are unchanged, and after the raw snapshot block is removed, the current-token KV at the original readout indices is passed to the action head. The two configurations differ in timing as follows.

Table 7: Shallow and deep configuration differences. Outside frame 0, snapshots are applied at read events. A purely periodic refresh does not itself trigger snapshot application.

Field	Shallow	Deep
Read event	controller refresh or phase change	active-phase change
Read direction	either	either
Write condition after frame 0	read event with acting label	entry to acting
Minimum snapshot age	1 replan	0 replans
Non-refresh application	disabled	frame 0 only
Frame-0 snapshot written	yes	yes
Frame-0 snapshot applied	no	yes

Intuitively, shallow conditioning appends the snapshot to what the action head reads, whereas deep conditioning lets the backbone re-encode the current prefix while attending to the snapshot as past key-values. For shallow application, snapshot KV is concatenated after the current readout. The action head uses the resulting valid context length for action positions. For deep application, snapshot KV is copied before use, its keys are rotated by a uniform RoPE offset equal to the current valid prefix length, and values are unchanged. The current full prefix is re-encoded with its original positional IDs and prefix mask plus all-to-memory attention. The raw memory block is then removed; current-token KV is sliced to the same index order and pad mask as the original readout. The persistent working bank is not replaced with this conditioned result.

Frame 0 writes and then reads a snapshot from the same observation. Its contribution must be separated from retaining past observations.

Algorithm 2 Main deep configuration: one episode

Optionally route instruction (prompt-only, once per episode); if fallback, execute baseline and return.
Initialize moving state, empty working bank, embedding history, snapshot.
for replan $t = 0, 1, \ldots$ **do**
  Embed current observation/instruction with original prefix masks and positions.
  if $t = 0$ **then**
    Full-prefix forward; select L, S; build bank; choose full-prefix readout.
  else
    Update phase controller; record refresh request e_t and phase change δ_t.
    if e_t or $t \bmod 2 = 0$ **then**
      Full-prefix forward; reselect S only if e_t; rebuild bank; read full prefix.
    else
      Partial-forward changed tokens; patch bank; choose phase readout plus extras.
    end if
  end if
  if $t > 0$ and δ_t and a snapshot exists **then**
    Deep-condition current readout using the previous snapshot (either direction).
  end if
  if $t = 0$ or (δ_t and phase is acting) **then**
    Write snapshot from unconditioned full-prefix KV; replace previous snapshot.
  end if
  if $t = 0$ **then**
    Apply the frame-0 snapshot through deep conditioning.
  end if
  Denoise actions from resulting readout; execute chunk; update embedding history.
end for

D Continuity and Input-Source Ablations

L1 is a separate all-base continuity configuration: no selected-bank patching or explicit snapshot, periodic full refresh, and temporary changed tokens outside the bank; L1 and L2 are therefore not interchangeable names. Table 8 shows that maintaining current prefix state is the main source of the Task 8 improvement: L1 raises Task 8 from 91/150 to 144/150, and removing its stale base-camera readout still gives 145/150, so the gain does not depend on directly reading the old base-camera view. The quota-matched no-snapshot refresh schedules likewise keep Task 8 at 96–98/100 (Appendix O). Over the suite, however, L1 reaches only 1363/1500 directly. The residual-patch variant reaches the same Task 8 count but 1384/1500 direct; it is listed separately and does not redefine the no-patch L1 control. Selection alone gives 141/150 on Task 8 and 1369/1500 direct. The language-only control scores 0/150 on Task 8.

Table 8: Continuity and input-source controls. Task totals are out of 150; suite totals are out of 1,500. L1 is the no-residual-patch configuration; respatch and selection-only are distinct controls. Routed totals use the baseline fallback, with the enabled sets held fixed. The baseline and PACER rows are the same policies as in the main results. The language-only control was run on Task 8 over the three seeds.

Condition	Task 8	Task 6	Direct	Router A	Router M
$\pi_{0.5}$ (baseline)	91/150	137/150	1385	1385	1385
L1: periodic continuity (no residual patch)	144/150	118/150	1363	1442	1408
L1 without stale base-camera readout	145/150	124/150	1350	1444	1422
residual only	142/150	135/150	1375	1443	1436
with residual patch	144/150	132/150	1384	1447	1434
keep stale wrist, drop base	139/150	116/150	1337	1440	1411
selection-only	141/150	135/150	1369	1437	1426
language only (Task 8)	0/150	—	—	—	—

Cadence and sensitivity. For the main deep configuration, changing p2 to p1 changes periodic refresh and its associated full-prefix denoising. It does not change snapshot reads to every replan. With p1, direct success is 1386/1500, Router A 1389/1500 and Router M 1387/1500; Task 8 is 92/150 and Task 6 136/150. Table 9 reports a separate seed 7 sensitivity subset. Among the tested constants, the refresh period matters most: p1 lowers Task 8 from 149/150 to 92/150, and in Table 9 refresh $= 1$ and $= 4$ lose 24 and 34 successes, whereas threshold and minimum-hold perturbations lose at most 12. The refresh period is sensitive for a structural reason rather than as a tuned constant: a period of 1 rebuilds the full prefix at every replan, which is the original policy's own recomputation and leaves no maintained state to carry forward. A period of 2 was not selected by tuning; it is the smallest period that carries the maintained state across one additional replan.

Table 9: One-at-a-time sensitivity analysis, seed 7, tasks 0/1/3/6/7/8 (300 episodes). The nominal row is the main configuration's seed 7 result (Table 19). Threshold and minimum-hold perturbations lose at most 12 of 300 successes; changing the refresh period loses 24–34, because a period of 1 is the original per-replan recomputation.

Constant change	Task 6	Task 8	Subtotal (/300)	Δ
nominal (main)	50/50	50/50	297/300	—
gripper thr. $\times 0.5$	44/50	49/50	290/300	-7
gripper thr. $\times 2$	45/50	49/50	289/300	-8
displacement thr. $\times 0.5$	44/50	48/50	290/300	-7
displacement thr. $\times 2$	41/50	49/50	285/300	-12
minimum hold $= 1$	47/50	47/50	289/300	-8
minimum hold $= 4$	45/50	47/50	289/300	-8
refresh $= 1$ (p1)	45/50	31/50	273/300	-24
refresh $= 4$	39/50	43/50	263/300	-34

E Signal and Readout Ablations

Table 10: Count-matched refresh schedules, three seeds; each cell is time-7 / phase-triggered with the same readout, and both keep p2. Time-7 matches the mean controller refresh interval measured on Task 8. Unmarked cells are direct successes out of 1,500; †cells are Tasks 6+8 out of 300. Snapshot columns use fixed-index snapshots, except the deployed row, which uses all-base deep conditioning; there, time-7 flips the moving/acting labels every seven replans, so readout switching, snapshot writes and deep reads follow the clock. Per-task counts are in Tables 11 and 15.

Readout	No snapshot	+shallow	+deep
Fixed all-base	1385 / **1393**	—	—
Residual-only	1387 / **1398**	1388 / **1412**	—
$L \cup S \to L \cup S$	1381 / **1406**	287 / **291**†	280 / **290**†
$S \to L \cup S$ (deployed)	—	—	1413 / **1433**

The time-7 interval matches the mean controller refresh interval measured on Task 8 with the deployed configuration (one request per 6.72 replans over 150 episodes, counting entries to and exits from acting and the checks within acting); an interval of 6 would coincide with every third p2 refresh. On the deployed configuration ($S \to L \cup S$ readout with all-base deep conditioning), the clock-driven counterpart keeps everything else fixed and only replaces the phase labels by labels that flip every seven replans. Over three seeds it reaches 1413/1500 direct and 1448/1500 under Router M, against 1433 and 1459 for PACER; per seed the direct totals are 472, 470 and 471 against 484, 473 and 476, and Tasks 8 and 6 reach 143 and 146 against 149 and 147. In every cell of Table 11, which compares time-7 and phase-triggered refresh without snapshot injection at the listed readouts, the phase-triggered count is no lower than the time-7 count, and direct success is higher by 8 to 25 episodes out of 1,500. The phase controller also requests refresh while remaining in acting, so the gain comes from phase-triggered refresh rather than from transition alignment alone.

Table 12 separates the L2 readout choices. The selected $S \to L \cup S$ row is 1407/1500 direct, with Task 8=145 and Task 6=133; it is the no-snapshot row in the main result table. The $L \cup$

Table 11: Time-7 versus phase-triggered results with corresponding no-injection readouts. Tasks are out of 150 and direct totals out of 1,500. Router M uses each condition's counts on the enabled tasks and the baseline's counts on the others (Appendix B). The stale-wrist row keeps the stale wrist-camera view and drops the base-camera view, as in Table 8.

	time-7 clock				phase-triggered			
Readout (no injection)	Task 8	Task 6	Direct	Router M	Task 8	Task 6	Direct	Router M
fixed all-base	141	127	1385	1430	141	134	1393	1435
residual-only	142	132	1387	1433	145	134	1398	1440
$L \cup S \to L \cup S$	143	129	1381	1433	145	133	1406	1434
keep stale wrist, drop base	131	113	1354	1407	145	126	1369	1422

$S \to \mathrm{res}$ alternative gives 150/150 on Task 6, limiting claims that explicit snapshot memory is necessary. No single readout leads on the full suite and on both Tasks 6 and 8, so readout choice is the method's main sensitivity: across the no-snapshot readouts in Table 12, Task 6 ranges from 131/150 to 150/150. These high counts all come from maintained inference state, consistent with the continuity controls in Appendix D.

Table 12: No-snapshot readout comparisons under phase-triggered refresh. Notation is moving readout $\to$ acting readout; res denotes language KV plus current residual-output KV. The res$\to$res and residual-only conditions expose these same token types, but form their residual KV against different banks: selected $L \cup S$ versus expanded all-base camera KV. Residual-only here also uses phase-triggered refresh, unlike the periodic-only row in Table 8.

Readout (phase-triggered, no injection)	Task 8	Task 6	Direct
fixed all-base	141/150	134/150	1393/1500
$S \to S$	144/150	135/150	1389/1500
$L \cup S \to S$	143/150	131/150	1399/1500
$L \cup S \to L \cup S$	145/150	133/150	1406/1500
$S \to L \cup S$ (main)	145/150	133/150	1407/1500
residual-only	145/150	134/150	1398/1500
residual-output readout grid			
$S \to$ res	145/150	144/150	1410/1500
$L \cup S \to$ res	142/150	150/150	1420/1500
res $\to S$	145/150	143/150	—
res $\to L \cup S$	145/150	143/150	1418/1500
res $\to$ res	146/150	145/150	1402/1500

On the full suite, the selected $S \to L \cup S$ readout gives 1407/1500 without a snapshot, 1404/1500 with fixed-index deep conditioning and 1433/1500 with all-base deep conditioning, while its Task 6+8 subtotal rises from 278/300 to 295/300 and 296/300; fixed-index and all-base deep snapshots thus differ by one success on Task 6+8 but by 29 on the full suite. Residual-only readout with fixed-index deep conditioning gives 1421/1500, and res$\to$res changes from 291/300 to 283/300 with all-base deep conditioning. Removing the temporary changed-token path E_t reduces Task 6+8 to 80/300 and 95/300 with fixed-index and all-base deep conditioning, so it cannot be dropped.

F Snapshot Configuration Ablations

This appendix covers snapshot content, write/read schedules, initialization and the time-7 comparison with snapshots. Tables 13 and 14 give full-suite totals alongside the Task 6+8 subtotals.

Shallow and deep configurations. Shallow conditioning appends snapshot KV to the action head's readout, whereas deep conditioning passes it to the backbone as past key-values (Appendix C). Table 17 (Appendix G) compares these configurations with their Task 6+8 subtotals; all use the selected $S \to L \cup S$ readout except the readout-direction control; among shallow, fixed deep and all-base deep, the main configuration has both the highest direct total (1433) and the high-

est Task 6+8 subtotal (296/300). Shallow and deep also differ in event source, non-refresh eligibility and minimum snapshot age (Table 7).

Snapshot content. Table 13 varies the reported snapshot selector. The all-base/language configuration scores 296/300 on Task 6+8, compared with 295 for fixed indices, 289 for visual-only, and 258 for the $L \cup S$-indexed control, so what is stored matters locally. These differences motivate content-controlled experiments; they do not show that the benefit comes specifically from temporally distant scene information. In the frozen content study (Appendix P), replacing snapshot content with current-frame content leaves Task 8 at 98/100 and Task 6 at 96/100, versus 98/100 for both with historical content: the benefit comes mainly from maintained state and the conditioning path rather than from old views in the snapshot.

Table 13: Snapshot-selector comparisons with the readout and deep conditioning fixed, so only the selector varies; Task 6+8 out of 300. The $L \cup S$-indexed condition has no full-suite total.

Snapshot	Tasks 6+8	Direct
fixed-index	295/300	1404/1500
all-base, PACER (main)	296/300	1433/1500
visual-only all-base	289/300	1421/1500
$L \cup S$-indexed	258/300	—

Write/read schedules and initialization. The default deep configuration reads on active-state changes in both directions and writes on entry to acting; it is not a to-acting-only read schedule. ACT instead writes on completion, and STA reverses the write/read direction. With the all-base deep selector, the default configuration reaches 1433/1500 direct, above ACT (1419) and STA (1415), the best of the three. With the same $L \cup S \rightarrow L \cup S$ readout, phase-triggered refresh exceeds the time-7 schedule without snapshots (278 versus 272), with shallow conditioning (291 versus 287) and with deep conditioning (290 versus 280; Table 15). On the main $S \rightarrow L \cup S$ readout, deep conditioning exceeds shallow (295 versus 289).

All-base frame-0-only and no-frame-0 deep controls score 290/300 and 291/300 on Task 6+8 (1417/1500 and 1425/1500 direct), versus 296/300 and 1433/1500 for the complete configuration, which is highest on both scales.

Table 14: Schedule variants. The first four numeric columns give Task 6+8 successes out of 300; the last gives full-suite direct successes out of 1,500 for all-base deep conditioning. ACT and STA change write/read directions; the default deep read can occur on either active-phase transition.

Write timing	fixed shallow	fixed deep	all-base shallow	all-base deep	all-base deep Direct
default events (main)	289	295	291	296	**1433**
completion write (ACT)	294	293	291	289	1419
reversed write/read (STA)	291	290	294	293	1415

Table 15: Time-7 versus phase-triggered schedules with snapshot conditioning, Task 6+8 out of 300. Rows pair time-7 and phase-triggered refresh under the same readout; the last row is the main $S \to L \cup S$ readout.

Schedule	Readout	none	fixed shallow	fixed deep	all-base deep
fixed time-7	fixed all-base	268	262	—	—
phase-triggered	fixed all-base	275	271	272	278
fixed time-7	residual-only	274	281	—	—
phase-triggered	residual-only	279	288	288	285
fixed time-7	$L \cup S \to L \cup S$	272	287	280	—
phase-triggered	$L \cup S \to L \cup S$	278	291	290	292
phase-triggered (main)	$S \to L \cup S$	278	289	295	**296**

Table 16: Readout and snapshot controls. (a) Shallow and deep use all-base snapshots; their schedules and initialization also differ. (b) Initialization controls use the main readout and all-base deep conditioning; frame-0 only limits application to once per episode, normally at initialization. Selector and write/read controls are in Tables 13 and 14.

(a) Task 6: readout and snapshot conditioning

Moving → acting readout	No snapshot	Shallow	Deep
$S \to L \cup S$ (main L2)	133/150	145/150	147/150
$L \cup S \to$ res	150/150	145/150	143/150
res → res	145/150	144/150	140/150

(b) Main readout, all-base deep: initialization and later use

Configuration	Task 8	Task 6	Direct
Complete	149/150	147/150	1433/1500
Frame-0 only	147/150	143/150	1417/1500
No frame-0	146/150	145/150	1425/1500

G PER-TASK AND PER-SEED RESULTS

Direct application and routed aggregation answer different questions. The main deep configuration's direct total exceeds the baseline on each seed (484, 473 and 476 versus 465, 458 and 462 out of 500); routed totals also reuse baseline cells on tasks outside the enabled set. Table 17 lists the direct counts of the snapshot configurations and recomputes baseline fallback. Tables 18 and 19 expand the working-memory, shallow and deep rows per task and seed. Without seed 7, direct PACER and Router M reach 949 and 967 against the baseline's 920 out of 1,000; without Task 8 they reach 1284 and 1310 against 1294 out of 1,350. All delta and routed columns use the baseline in Table 4.

Regressions on Tasks 2 and 9. We ran Tasks 2 and 9 again with the same checkpoint and protocol, two seeds (5051 and 5052) with 50 initial states each, together with the baseline. The regressions reproduce: the baseline reaches 97/100 on Task 2 and 91/100 on Task 9, whereas five memory variants reach 80–88/100 and 76–82/100. The variants are the deep configuration with historical or current-frame snapshot content (Appendix P) and three injection schedules with a fixed initial anchor (Appendix Q). Both tasks lie outside the router's enabled set (Appendix B).

Table 17: Snapshot configurations: Tasks 6+8 out of 300, direct and recomputed routed totals out of 1,500. All seven direct totals (1404–1433) exceed the baseline's 1385, so the direct gain does not hinge on a single configuration. Injection depth is not the only varying parameter between shallow and deep (Table 7). The frame-0-only and completion-state rows use the fixed-index selector; the all-base frame-0-only control is in Table 16(b). Routed columns use fixed-set baseline fallback.

Condition	Tasks 6+8	Direct	Router A	Router M
L2 + shallow	291/300	1413	1448	1452
L2 + fixed-index deep	295/300	1404	1447	1452
frame-0-only fixed-index deep	295/300	1426	1448	1456
readout-direction control ($L \cup S \to S$, all-base deep)	295/300	1424	1447	1457
completion-write shallow (fixed-index)	294/300	1415	1449	1454
completion-write deep (fixed-index)	293/300	1418	1445	1451
PACER, deep (main)	**296/300**	**1433**	**1451**	**1459**

Table 18: Working memory (L2: selected bank, readout $S \to L \cup S$, no snapshot) and L2 + shallow (all-base snapshot appended on the same readout), per task and seed. Without any snapshot, working memory raises Task 8 from 91 to 145 successes ($+19$, $+19$ and $+16$ per seed); shallow conditioning raises Task 6 from 137 to 145 with Task 8 at 146. Per-task cells are out of 50 per seed or 150 over three seeds; suite totals are out of 500 or 1,500. Routed columns substitute the baseline in Table 4 outside the enabled sets.

Configuration	Seed	t0	t1	t2	t3	t4	t5	t6	t7	t8	t9	Direct	Router A	Router M
L2 (no snapshot)	7	48	50	43	49	47	44	46	50	50	46	473	487	486
	42	47	50	42	50	49	47	42	50	47	42	466	479	477
	123	47	50	44	47	47	49	45	50	48	41	468	480	475
	3-seed	142	150	129	146	143	140	133	150	145	129	1407	1446	1438
L2 + shallow	7	49	50	40	50	48	49	48	49	50	44	477	487	489
	42	48	49	42	50	47	44	48	50	48	43	469	480	482
	123	48	50	41	49	48	48	49	50	48	36	467	481	481
	3-seed	145	149	123	149	143	141	145	149	146	123	1413	1448	1452

Table 19: PACER, deep (main), per task and seed (50 episodes per cell). Task 8 reaches 50, 49 and 50 on the three seeds, versus 31, 28 and 32 for the baseline, so its improvement appears on every seed. Direct counts sum all tasks; Router A/M substitute baseline results outside their enabled sets.

Seed	t0	t1	t2	t3	t4	t5	t6	t7	t8	t9	Direct	Router A	Router M
7	48	50	47	49	49	49	50	50	50	42	484	487	492
42	47	50	46	49	47	46	47	50	49	42	473	481	482
123	48	50	45	48	49	47	50	50	50	39	476	483	485
3-seed	143	150	138	146	145	142	147	150	149	123	1433	1451	1459

H STATISTICAL CALCULATIONS

Table 20 reports two-sided Fisher tests (Fisher, 1922) on aggregate success/failure counts for Router M versus the baseline, pooling the three seeds; per-seed counts are in Appendix G. The paired timing and snapshot-content studies are analyzed in Appendices O and P.

Task 6's comparison is 137/150 versus 147/150 ($p \approx 0.018$). For the frozen refresh-timing, snapshot-content and injection-timing studies (Appendices O, P and Q), intervals are individual 95% percentile intervals, and sign-flip tests assume exchangeability under the null.

Table 20: Two-sided Fisher tests, Router M versus baseline, three seeds pooled. Task 8 improves on each seed.

Comparison	Baseline	PACER	p
Suite, three-seed	1385/1500	1459/1500	9.7853×10^{-10}
Task 8, three-seed	91/150	149/150	1.0212×10^{-19}
Task 6, three-seed	137/150	147/150	1.8036×10^{-2}

I OPERATION COUNTS AND PROFILING

I.1 COST DECOMPOSITION

Maintaining state trades computation on current tokens against the cost of carrying and conditioning on retained KV. Let $T_t(n, r; k)$ denote encoding n current tokens against k cached entries and denoising from r readout entries. The baseline re-encodes all P prefix tokens and reads its full context of C entries at every replan, without cached backbone context. PACER instead combines full refreshes with partial forwards. For H replans, a bookkeeping approximation is

$$\bar{T} \approx \frac{1}{H} \sum_{t=0}^{H-1} \left[T_t(n_t, |R_t^{\text{act}}|; k_t) + c_t + \eta_t T_t^{\text{deep}} \right], \tag{9}$$

where $n_t = P$ on full refreshes and $|U_t|$ on partial forwards, k_t is the retained context read by the backbone, R_t^{act} is the final action context (including shallow snapshot entries when applied), c_t includes copying, selection and snapshot handling, and η_t indicates an additional deep forward. If ρ is the fraction of full-refresh replans, the first term averages to $\rho \bar{T}_{\text{full}} + (1-\rho)\bar{T}_{\text{partial}}$, where the two means are taken over their respective step types. A partial forward re-encodes $|U_t|$ changed tokens in place of the full prefix; selecting S_t or $L \cup S_t$ trades readout coverage against the context processed by the action head. The remaining terms account for maintaining that context and amortizing additional deep forwards over the episode. Partial encoding and selected readout offset these costs, so in the measured settings median inference latency is nearly unchanged (+0.6%) and mean latency rises by 4.3% (Section 4.7).

I.2 OPERATION COUNTS AND TIMING

In the snapshot-application logs, Task 8 deep p1 has 1,553 applications over 13,801 replans, so p1 does not inject at every replan.

The reported timing study used the same machine and bf16, excluding episode 0, on Tasks 6/7/8. Baseline and deep task-mean inference latencies round to 241 and 251 ms; averaged over Tasks 6/7/8, L2 readout only, shallow and deep PACER add 5.7, 4.1 and 10.4 ms (2.4%, 1.7% and 4.3%) over the 240.7 ms baseline, without any training. A second same-machine profile of the baseline and deployed PACER on one RTX 4090 (Tasks 6/7/8, ten episodes each after a warm-up episode; about 2,000 and 1,800 inferences) gives median per-inference latencies of 253.0 and 254.5 ms (+0.6%), mean latencies of 254.5 and 265.4 ms (+4.3%), and peak allocated GPU memory of 7.15 and 7.19 GiB (+0.6%). Table 21 retains per-task timing; Table 21 gives readout sizes.

Table 22 varies the action horizon on Task 8 (seed 7). Baseline success falls from 31 to 17 as the horizon grows, whereas the three memory configurations stay between 41 and 50.

Table 21: Same-machine inference profile (bf16; episode 0 excluded), Tasks 6/7/8. (a) Mean inference latency per task. (b) Mean live image-token KV and latency over the three tasks for readout configurations; selected readouts reduce live image-token KV from 968 to 800–826 while mean latency stays within 245–247 ms.

(a) Mean latency (ms)

Condition	Task 6	Task 7	Task 8
$\pi_{0.5}$ (baseline)	239.4	240.5	242.3
L2 readout only	249.1	246.8	243.4
L2 + shallow	248.3	243.4	242.7
L2 + fixed-index deep	258.6	254.1	250.2
frame-0-only fixed-index deep	249.6	249.5	240.2
PACER, deep (main)	253.5	252.1	247.9

(b) Live image-token KV by readout

Readout	KV	ms
$\pi_{0.5}$ (full prefix)	968	241
full base cache	863	245
$S \to L \cup S$ (L2, main)	818	246
$S \to S$ (narrow)	818	245
$L \cup S \to S$	823	246
$L \cup S \to L \cup S$ (wide)	826	245
residual only	800	247

Table 22: Task 8 action-horizon diagnostic, seed 7, 50 episodes per cell; the main protocol uses horizon 10.

Action horizon (Task 8, seed 7)	10	20	30	40	50
$\pi_{0.5}$ (baseline)	31	24	19	19	17
L2 readout only	50	44	45	45	43
L2 + shallow	50	46	45	44	44
PACER, deep (main)	50	44	45	42	41

J Qualitative Evidence Scope

Diagnostic annotations describe grasp/contact stalls, incomplete repeated-object placement, and relative-placement errors. The sampled-frame analysis in Appendix P.1 concerns the snapshot-content study, and Figure 4 shows physical examples.

Six Task 8 diagnostic rollouts each evaluate the baseline, working memory and full PACER at the same seed and initial state. Launch records, result files and 18 complete action traces agree on condition identity and termination, and initial end-effector and gripper values match within each triplet. Visual inspection sampled 24 frames from each of the 12 baseline/PACER videos; working-memory outcomes were checked from results and traces.

In five cases (seed 7, initial states 2, 6, 21, 41 and 49), the baseline times out at 520 actions after prolonged stalls around the first or second moka pot. Both working memory and full PACER succeed in each case. Baseline/PACER sampled frames show PACER progressing earlier through object manipulation. In the sixth case (seed 42, initial state 24) the baseline succeeds in 406 actions while both memory conditions time out. A sustained gripper-closing command can span attempted contact, holding and transport; its duration is not a measured grasp duration or retry count.

K Extended Related Work

Memory for robot policies. Beyond the methods discussed in Section 2, CAMP trains a memory module that keeps a compressed representation of the robot's own past actions for partially observable, contact-rich manipulation (Wang et al., 2026), and MAP-VLA builds a memory library from historical demonstrations and turns retrieved units into prompts for a pretrained VLA (Li et al., 2025b). RoboMME provides a standardized benchmark of history-dependent manipulation tasks, such as counting repeated actions or handling temporarily occluded objects, for evaluating memory in generalist policies (Dai et al., 2026). These approaches learn or retrieve memory content, whereas PACER maintains the frozen policy's own inference state and adds no trained memory component.

KV caching and token efficiency. In language models, StreamingLLM keeps initial attention-sink tokens together with a recent window for streaming generation (Xiao et al., 2024), H_2O evicts cache entries while retaining heavy-hitter tokens (Zhang et al., 2023a), SnapKV compresses prompt KV using attention patterns observed at the end of the prompt (Li et al., 2024b), and Prompt Cache precomputes and reuses the attention states of recurring prompt segments across requests (Gim et al., 2024). For visual tokens, Token Merging combines similar tokens inside vision transformers (Bolya et al., 2023), FastV prunes image tokens after the early layers of vision-language models (Chen et al., 2024), and interaction-aligned pruning selects visual tokens for embodied manipulation without training (Cheng et al., 2026); DeeR-VLA instead adapts computation through dynamic early exit (Yue et al., 2024). These methods aim at efficiency for a given input. PACER's selected readout also reduces live tokens, but its purpose is to carry state across replans.

Inference-time interventions and test-time adaptation. Inference-Time Intervention shifts selected activations along directions associated with truthfulness (Li et al., 2024a), contrastive activation addition adds steering vectors derived from paired examples (Rimsky et al., 2024), and representation engineering reads and controls high-level representations (Zou et al., 2023); for VLAs, TACO stabilizes execution at test time by selecting among sampled action noises (Yang et al., 2025a). Test-time adaptation methods such as Tent and SAR update normalization parameters by entropy minimization (Wang et al., 2021; Niu et al., 2023), and their failure modes have been studied systematically (Zhao et al., 2023a). Like activation interventions, PACER changes internal state without updating weights, but its intervention reuses the policy's own cached KV rather than directions estimated from contrast data, and no parameter is adapted. Its router is related to selective prediction, in which a model may abstain (Geifman & El-Yaniv, 2017); the router abstains on instruction structure rather than on confidence and then runs the baseline.

Training-based VLA improvement and task structure. Optimized fine-tuning (Kim et al., 2025) and reinforcement-learning post-training (Li et al., 2025a; Zang et al., 2025; Chen et al., 2025; 2026) improve VLA policies by updating their weights and are complementary to a frozen-policy intervention. Segmenting behavior into reusable skills (Konidaris et al., 2012) or waypoints (Shi et al.,

2023), and evidence that event boundaries act as access points for memory retrieval (Michelmann et al., 2023), motivate PACER's use of phase transitions as moments to refresh or re-expose maintained state.

L External Comparison Scope

Appendix K surveys further related work. TempoFit (Sun et al., 2026), Long-VLA (Fan et al., 2025), and HAMLET (Koo et al., 2026) provide related mechanisms with different training and evaluation choices. As of September 26, 2026, the public TempoFit repository states that it does not yet contain a runnable implementation, and its experimental documentation does not specify the exact memory-layer indices or full numerical configuration, including FGTB decay strength;[1] UAOR (Yang et al., 2026b) provides a project page but no released code. We therefore compare each method's reported gain over its own baseline (Table 23). Measured this way, routed PACER gives the largest gain and removes the largest share of baseline failures, while direct PACER's gain is below TempoFit's on $\pi_{0.5}$. Base checkpoints, evaluation protocols and trial counts differ, and only PACER's routed row uses baseline fallback.

Table 23: Reported LIBERO-Long gains of training-free methods over their own baselines. Failure reduction is the share of baseline failures removed, $(s - b)/(100 - b)$. TempoFit and UAOR values are from their papers (base policies include RLinf (Zang et al., 2025) and OpenVLA-OFT (Kim et al., 2025)); neither has released code.

Method	Base policy	Baseline	With method	Gain (pp)	Failure reduction
PACER + Router M	$\pi_{0.5}$	92.3	97.3	+5.0	65%
TempoFit	$\pi_{0.5}$ (RLinf)	92.6	96.6	+4.0	54%
PACER, direct	$\pi_{0.5}$	92.3	95.5	+3.2	42%
TempoFit	QwenGR00T	90.8	94.4	+3.6	39%
UAOR	OpenVLA-OFT	94.2	96.2	+2.0	34%
UAOR	π_0	80.5	82.5	+2.0	10%

In mechanism, Long-VLA and HAMLET train phase or history components with the policy, whereas PACER and TempoFit keep policy weights frozen. TempoFit writes and retrieves layer-wise KV at every step, while PACER maintains the current prefix state and keeps one snapshot written at phase events (Section 2).

M Real-Robot Setup

The paper's physical study uses a Piper 6-DoF arm and parallel-jaw gripper, with 7-D state/actions (six joint angles plus gripper), action horizon 10, and a fixed head/base camera plus wrist camera. The converted $\pi_{0.5}$ base was fine-tuned on an in-house dataset of seven pick/place, stacking and drawer tasks, with 236 teleoperated demonstrations (approximately 134k frames), and then frozen. Normalized state and delta joint predictions are converted using the Piper training statistics. PACER itself adds no training.

All five tasks use the main simulation configuration (the $S \to L \cup S$ readout with deep snapshot conditioning), applied directly without routing. No instruction filtering is needed: pen transfer would fall back to the baseline under the simulation router rules, yet direct deployment raises it from 16/50 to 45/50. The phase signal is gripper-only, so the simulation end-effector displacement trigger is unused. Table 24 lists the fine-tuning tasks: all five evaluation tasks are among them and use the same prompts as in fine-tuning. The videos of two yellow blocks show where the gain comes from. Baseline failures are color-selection errors: after the first block, the policy does not keep track of which block it has chosen and picks up blocks of other colors. PACER episodes select the two yellow blocks.

Across the five tasks, success rises from 150/250 to 215/250 and improves on every task, including pen transfer (16/50 to 45/50) and two yellow blocks (24/50 to 44/50). The five-task results in Ta-

[1] `https://github.com/LucioSunj/TempoFit`; configuration details: `https://github.com/LucioSunj/TempoFit/blob/master/docs/experiments.md`.

Table 24: Piper fine-tuning tasks (236 demonstrations, 134,476 frames) and their use in the physical evaluation.

Fine-tuning task	Training prompt	Demos	Evaluation use
Second drawer	Put the blocks into the second drawer and close it.	32	Evaluated, same prompt
Two yellow blocks	Put two yellow blocks into the drawer.	40	Evaluated, same prompt
Pen into cup	Pick the pen and put it in the cup.	30	Evaluated, same prompt
Stack three bowls	Stack the three bowls on top of each other.	30	Evaluated, same prompt
First and third bowls	Place the first and the third bowls in the middle.	30	Evaluated, same prompt
Cups on plate	Put the two cups onto the plate.	41	Fine-tuning only
Carrot into bowl	Pick the carrot and put it in the bowl.	33	Fine-tuning only

ble 3 contain 50 episodes per condition and task, including the initial ten episodes under unchanged checkpoint, camera and task settings. For *"Put the blocks into the second drawer and close it,"* all blocks must come to rest inside that drawer; closing it is not required. For *"Put two yellow blocks into the drawer,"* both yellow blocks must rest inside it, and a wrong-colored placement or an unretrieved second target counts as failure. For *"Pick the pen and put it in the cup,"* the pen must be released and come to rest inside the cup. For *"Stack the three bowls on top of each other,"* all three bowls must end nested in one stack. For *"Place the first and the third bowls in the middle,"* the two bowls identified by their positions must both reach the middle location; a wrong bowl or an unmoved target counts as failure. Retraction is not an extra scoring condition. A separate third-person camera supplies the displayed frames, not policy input.

N Reproducibility Notes

The simulation evaluation is reported with Python 3.11, PyTorch 2.7.1/CUDA 12.6, and single-GPU inference on an RTX 4090 (24 GB). Horizon, replan and controller settings are given in Appendix C. Appendix A specifies the protocol and the verified baseline files, and Appendix H gives the statistical methods. An anonymized code supplement accompanies the submission.

O Frozen Refresh-Timing Evaluation

Table 25: Quota-matched timing studies on Task 8, 100 episodes per condition (two seeds $\times$ 50 matched initial states). Differences are phase minus the listed schedule in percentage points with individual 95% initial-state-cluster bootstrap intervals; all prespecified Holm-adjusted $p = 1.0$. Timing schedules share a per-window quota; native p2 is shown for reference. The injection-timing study is described in Appendix Q.

Study	Schedule	Success	Diff. (pp)	95% CI
Refresh timing (no snapshot)	Phase	97/100		
	Fixed	96/100	$+1$	$[-3, +5]$
	Random	98/100	-1	$[-5, +3]$
	Native p2 (reference)	98/100		
Injection timing (fixed frame-0 anchor)	Phase	97/100		
	Fixed	97/100	0	$[-4, +4]$
	Random	98/100	-1	$[-5, +2]$

This study fixed its sample size, two primary contrasts and analysis before outcomes were examined. It contains exactly 400 episodes: four conditions, seeds 2027/2028 and the same 50 initial-state IDs per seed. The protocol, checkpoint, initial-state file and 509 source files were hashed before the formal runs. Four development episodes (seed 7, initial state 0) are excluded, and all eight 50-episode shards completed without outcome-based extensions.

The three quota-controlled schedules use the same selected-bank implementation, fixed $L \cup S \rightarrow L \cup S$ readout, no snapshots and PACER's partial-forward mask. Action horizon is 10, replan interval 5, precision bfloat16, denoising steps 10 (Lipman et al., 2023) and a maximum episode length of 520

actions after 10 settling steps. The phase controller runs under each schedule, but its refresh request is replaced by the quota rule and the independent p2 channel is disabled. Each scheduled refresh performs a full forward, rebuild and short-set reselection. Random scheduling uses an independent hash-keyed RNG and does not consume the action RNG. The native p2 reference retains PACER's combined controller/periodic schedule with the same readout and no snapshots. All three quota schedules keep the maintained prefix state and differ only in when refreshes occur, so the study tests whether refresh timing matters.

Table 26: Actual operations and quota verification for the refresh-timing study in Table 25, from per-replan journals; all full forwards are counted, including initialization. Rebuild counts equal full-forward counts, and all snapshot write/application counts are zero. Each complete window of each quota schedule contains exactly one full forward, rebuild and reselection; additional full forwards occur only in truncated windows. Per-seed successes of the three quota schedules lie between 48/50 and 50/50, and native p2 reaches 49/50 on both seeds. Full forwards make up 50.36%, 50.34%, 50.63% and 57.52% of replans under phase, fixed, random and native p2. Purely periodic native-p2 refreshes do not reselect S.

Schedule	Replans	Full fwd.	Partial fwd.	S reselect.	Complete windows	Truncated replans	Extra full fwd.	Refresh at phase change
Phase	8,019	4,038	3,981	4,038	3,932	55	6	795/814
Fixed	8,025	4,040	3,985	4,040	3,940	45	0	413/821
Random	8,067	4,084	3,983	4,084	3,958	51	26	435/818
Native p2	7,982	4,591	3,391	1,324	—	—	—	—

Each quota schedule also has 100 initialization forwards. The schedules follow the same complete-window rule, and their realized totals differ only with trajectory length and truncation (Table 26). The phase rule does align refreshes with phase changes (795/814, versus about half under the other schedules), yet success barely changes: in this configuration, maintaining state matters more than refresh alignment.

The analysis unit is an initial-state cluster containing both seeds. For each contrast, the two within-state paired success differences are averaged, then averaged across 50 clusters. We use 20,000 paired cluster-bootstrap replicates for a 95% percentile interval (Efron, 1979; Efron & Tibshirani, 1993) and 20,000 two-sided cluster sign flips (Fisher, 1935), with random seed 3401 and a plus-one Monte Carlo correction. Both contrasts use the same resampling arrays, and Holm correction (Holm, 1979) covers phase–fixed and phase–random. Their effects are $+1$ and -1 percentage point, intervals $[-3, 5]$ and $[-5, 3]$, and raw and adjusted $p = 1.0$. Only five clusters have nonzero differences in each contrast. All three schedules reach 96–98/100, far above the main-table baseline's 91/150.

P Frozen Snapshot-Content Evaluation

Like the refresh-timing study, this study fixed its sample size and two task-specific H−C contrasts before formal outcomes. It contains 400 episodes: historical (H) and current-frame (C) conditioning on Tasks 8 and 6, with seeds 3031/3032 and initial-state IDs 0–49 per task and seed. The protocol, source, checkpoint and initial states were frozen before evaluation; four development diagnostics are excluded, and all eight 50-episode groups completed without outcome-based extensions. Both conditions apply PACER directly, without routing, and use the same frozen $\pi_{0.5}$ checkpoint, observation pipeline and action decoding.

Both conditions retain the main $S \to L \cup S$ readout, periodic refresh every two replans, the phase controller, frame-0 application, all-base/language deep snapshots and PACER's partial-forward mask. Action horizon is 10, replan interval 5, precision bfloat16, denoising steps 10 and an episode limit of 520 actions after 10 settling steps. Each actual deep application includes one extra unconditioned current-prefix preparation forward in both conditions. C selects the historical snapshot's indices and valid-token mask from this current KV, while H discards that candidate and reads the stored historical payload. Neither condition writes the replacement or deep-conditioned output into persistent memory. Snapshot storage and read eligibility follow the same rules, although their real-

ized timing may diverge with the trajectories. At frame 0, historical and current content coincide; fixed-input checks also found identical actions for the H wrapper and native PACER.

Table 27: Seed-level results, actual operations and injected-content age (in replans) for the snapshot-content study in Table 2. All four success totals lie between 96/100 and 98/100: replacing historical with current-frame content leaves success nearly unchanged. Every row includes both fixed seeds (100 episodes per condition); development diagnostics are excluded. Native full forwards exclude the extra current-prefix preparation and deep-conditioning forwards, which are shown separately. Ages describe the injected content, not the historical snapshot that C replaces.

Task	Content	Seed3031	Seed3032	Total	Replans	Native full	Prepare / deep	Age mean / max
8	Historical H	49/50	49/50	98/100	8,014	4,636	948 / 948	8.03 / 26
8	Current C	49/50	49/50	98/100	8,003	4,649	924 / 924	0 / 0
6	Historical H	48/50	50/50	98/100	4,571	2,870	810 / 810	6.23 / 29
6	Current C	49/50	47/50	96/100	4,713	2,936	798 / 798	0 / 0

Within each task, the two seed-specific paired success differences for an initial state are averaged into one cluster; the effect is the mean over 50 clusters. We use 20,000 paired cluster-bootstrap replicates for individual 95% percentile intervals and 20,000 two-sided cluster sign flips with a plus-one Monte Carlo correction, using analysis seed 3402. Holm adjustment covers the two task-specific contrasts. Task 8 has two H-only and two C-only successes, yielding 0 percentage points with interval $[-4, +4]$ and raw $p = 1.0$. Task 6 has four H-only and two C-only successes, yielding $+2$ points with interval $[-2, +6]$ and raw $p = 0.6279$. Both adjusted p-values are 1.0; interval interpretation follows Appendix H.

Matching one preparation forward per application does not make the total episode compute equal: application counts and trajectory lengths differ (Table 27). The historical payload has a mean age of 8.03 (Task 8) and 6.23 (Task 6) replans, whereas C injects current content with age zero; despite this difference, success is nearly identical. Both working banks keep the same maintained state and both conditions retain deep conditioning, so the comparison isolates snapshot content. The extra preparation forwards also prevent treating these runs as a fresh measurement of native PACER latency.

P.1 Failure Observations and Annotation Scope

The qualitative sample contains all ten failed rollouts and their ten successful counterparts at the same task, seed and initial-state ID. These are the ten outcome-discordant H/C pairs; the remaining 190 of 200 pairs succeed in both conditions and are outside the visual sample. Each of the 20 videos contributes 16 uniformly sampled frames, and each failed video contributes a further 16 frames around selected interaction stages. The authors inspected these frames together with the corresponding action logs and videos to categorize the failures.

All ten failures reach the 520-action limit, and failures occur in both conditions (4 H, 6 C). On Task 8, all four are prolonged or repeated grasp attempts (2 H, 2 C). On Task 6, one first mug transfer is incomplete (H), four rollouts remain incomplete after the second-object manipulation (1 H, 3 C), and one mug topples during that manipulation (C). None is a wrong-target error.

A timing analysis includes all 200 Task 8 H/C rollouts, counting executed policy actions and assigning the 520-action limit to timeouts. Mean lengths are 398.65 for H and 398.25 for C; H minus C is $+0.40$ actions with a 95% initial-state cluster-bootstrap interval of $[-6.77, +7.91]$. The 50 initial-state clusters retain both seeds, using 20,000 resamples and analysis seed 3501. Consistent with the success results, historical content shows no detected difference in execution length. Video playback duration is not used as execution time, and action counts do not measure physical grasp/contact events.

Q Fixed-Initial-Anchor Injection Timing

This separate Task 8 study asks whether injection timing matters when the snapshot is a fixed initial anchor; success changes little across schedules (97/97/98, Table 28). Its configuration and analysis were frozen before 300 formal rollouts. It uses seeds 4041/4042 and initial states 0–49 for

each of phase, fixed and random injection schedules. Development diagnostics are excluded. All conditions retain PACER's periodic refresh, phase controller, phase-dependent readout and partial-forward mask. A single all-base/language snapshot is stored at frame 0 and never overwritten. All conditions inject it initially; subsequent PACER injection channels are disabled in favor of the common window rule.

Starting at replan 1, each complete seven-replan window allows one injection. Phase injects at the first committed phase boundary, falling back to the last slot if none occurs; fixed uses the last slot; random draws an independent uniform slot. Complete-window quotas match, while early termination and partial windows can produce different total operation counts. Journals verify actual injections and an unchanged snapshot hash from the first to last use in every episode. Table 28 reports both success and realized operations; boundary overlap excludes initialization and uses each condition's own trajectory.

Table 28: Fixed-initial-anchor injection timing, 300 formal episodes. Mean injections include frame 0; boundary overlap is the fraction of non-initial injections coinciding with a phase boundary.

Schedule	Seed4041	Seed4042	Total	Mean replans	Mean injections	Boundary overlap
Phase	49/50	48/50	97/100	79.02	11.76	51.9%
Fixed	48/50	49/50	97/100	79.26	11.75	11.1%
Random	48/50	50/50	98/100	79.35	12.10	10.6%

Phase minus fixed is 0 percentage points (95% interval $[-4, +4]$); phase minus random is -1 ($[-5, +2]$). Both raw and Holm-adjusted two-sided p values are 1.0. Each of the 50 initial-state clusters retains the two seeds; 20,000 percentile-bootstrap resamples and cluster sign flips use analysis seed 3403, with a plus-one Monte Carlo correction and Holm correction over the two contrasts. Interval interpretation follows Appendix H. The phase rule raises boundary overlap from about 11% to 52% while success stays at 97/97/98, so injection timing is not critical in this setting and no timing advantage is detected. The study uses a fixed initial anchor with the original working-memory rules retained.

R Mid-Episode Memory Intervention

This study asks whether PACER's benefit in the second half of an episode depends on information stored before that point. For each initial state, deployed PACER runs from the start until a fixed cut point S: on Task 8, the first replan boundary at which exactly one moka pot rests in the stove region and the end effector is more than 10 cm away from it; on Task 6, the first replan boundary at which the white mug rests on the plate, the end effector is more than 10 cm away and the pudding has not moved. At S we save the simulator state, the random-number state and PACER's memory, and continue the same episode from the identical state under four conditions with the same remaining step budget: *keep* (PACER with its own memory), *clear* (PACER with its memory reset at S, so initialization happens at S), *swap* (PACER with the memory another initial state had at its own cut point) and *baseline* (the frozen policy). All four conditions restore the same saved state.

The design, cut-point rules, contrasts and analysis were frozen, with source hashes, before the formal runs. The study uses seeds 6061/6062 and initial states 0–49 on Tasks 8 and 6, with the main protocol (action horizon 10, replan interval 5, bfloat16, ten denoising steps, 520 actions after 10 settling steps); two development initial states per task are excluded. All 200 initial states reach S, at a median of 210 actions on Task 8 and 140 on Task 6. Contrasts are keep minus each other condition, paired by initial state, with 95% initial-state cluster-bootstrap intervals (20,000 resamples, both seeds of a state resampled together).

On Task 8, PACER still adds 17 successes over the baseline after the first pot is placed. Clearing its memory at S or replacing it with another episode's memory leaves success at 93/100 and 97/100, within three successes of keep, so the second-half benefit comes from state that PACER rebuilds after S rather than from information stored before it. No condition ever lifts the pot that is already on the stove again, and successful episodes need a median of 185–188 actions after S in every condition. On Task 6, the remaining pudding placement is completed at the same rate by all four conditions (93–

Table 29: Mid-episode memory intervention: success after the cut point S, out of 100 (two seeds $\times$ 50 initial states; per seed in parentheses). Differences are keep minus the listed condition in percentage points with 95% initial-state cluster-bootstrap intervals.

	Keep	Clear	Swap	Baseline
Task 8	96 (49, 47)	93 (47, 46)	97 (50, 47)	79 (41, 38)
keep minus		$+3$ $[-2, +9]$	-1 $[-5, +2]$	$+17$ $[+9, +26]$
Task 6	93 (45, 48)	95 (48, 47)	93 (46, 47)	94 (46, 48)
keep minus		-2 $[-8, +3]$	0 $[-4, +4]$	-1 $[-7, +5]$

95/100), consistent with the Task 6 gain arising at the first placement through snapshot conditioning (Section 4.3).

S TRANSFER TO OPENVLA-OFT ON LIBERO-MEM

We adapted PACER to OpenVLA-OFT (Kim et al., 2025), whose image, language, proprioceptive and action positions share one bidirectional transformer, and evaluated it on two LIBERO-Mem tasks (LIBERO-Mem, 2025), bowl grasping (task 0) and cheese placement (task 8), whose sequential subgoals require remembering earlier steps. The frozen task checkpoints, inputs and action head are unchanged. We test two adaptations: an event-memory version with a capacity-one base-camera and language snapshot, read at phase changes and written on entry to acting, and a full adaptation that adds the working bank, L/S selection, changed-token patching, periodic p2 refresh and the phase controller. All conditions use seed 7, bfloat16, action chunks of 8, 10 settling steps, at most 600 control steps and the same 50 initial states per task; success requires completing all ordered subgoals.

Table 30: OpenVLA-OFT on two LIBERO-Mem tasks, 50 matched initial states per condition and task.

Task	OpenVLA-OFT (baseline)	Event memory	Full adaptation
Bowl grasping (task 0)	8/50	23/50	16/50
Cheese placement (task 8)	10/50	10/50	15/50

The full adaptation improves both tasks, from 8/50 to 16/50 on bowl grasping and from 10/50 to 15/50 on cheese placement, and the event-memory version raises bowl grasping to 23/50 while matching the baseline on cheese placement. The prefix-state memory therefore also improves a policy with a different architecture and action head, while median inference latency changes little (113.58, 112.98 and 127.25 ms for the baseline, event memory and full adaptation). Because of compute and time limits, OpenVLA-OFT checkpoints for the remaining LIBERO-Mem tasks are still being trained, and their evaluation is in progress.