How Should Vision-Language-Action Models Use Proprioceptive State?

Anonymous authors
Paper under double-blind review

Abstract

Recent Vision–Language–Action (VLA) models almost universally take robot proprioceptive state as input, yet wire it in incompatible ways—serialized into text prompts, projected into the vision–language prefix, or fed directly to the action expert—and almost always as a single current frame. Three questions remain open: (1) whether, and on which tasks, current state actually improves closed-loop control; (2) how much state history helps, and whether its benefit reflects genuine temporal variation rather than added conditioning capacity; and (3) where state should enter the model—the vision–language backbone or the action-generation module. We answer these questions through controlled experiments on a flow-matching VLA, fixing the backbone, training data, action representation, and evaluation protocol throughout. We implement five representative interfaces—discrete state prompt, VLM prefix, action prefix, state expert, and feature modulation—under matched implementation details, and evaluate them on 45 atomic tasks spanning three task families plus 20 composite tasks; we then sweep the state-history length from 1 to 96 frames to examine how historical state information affects model performance. The experiments yield systematic answers to all three questions, distilled into testable design principles for state-aware VLAs.

1 Introduction

Trained on increasingly large-scale robot demonstrations, recent Vision-Language-Action (VLA) models can execute hundreds of manipulation tasks with a single set of weights and generalize across objects and scenes (Intelligence et al., 2025; Bjorck et al., 2025; Chen et al., 2026). Inspired by biological motor control—animals rely on proprioception to produce smooth, coordinated, and robust movement—it is natural to expect the robot's proprioceptive state (joint angles, end-effector pose, gripper aperture) to play a similar role in a VLA (Guo et al., 2026): smoother control, more stable motion phases, and more precise contact-rich manipulation. Yet a gap remains. Vision and language have been pretrained at scale inside the VLM. Proprioceptive state has no such pretraining. As a result, proprioception remains the least examined and least understood input in VLAs.

Precisely because no consensus exists, current VLAs use state in strikingly diverse ways. At the *representation* level, $\pi_{0.5}$ (Intelligence et al., 2025) discretizes the state into text tokens appended to the language prompt, whereas OpenVLA-OFT (Kim et al., 2025a) and GR00T N1 (Bjorck et al., 2025) project it continuously into embeddings. At the *injection* level, some designs feed the state into the VLM, where it joins images and language in multimodal context modeling, while others send it directly to the action expert to condition action generation. At the *temporal* level, the vast majority of methods use only the current single frame, and few examine the temporal evolution of state (Guo et al., 2026). Moreover, these design differences are entangled with the backbone, pretraining, data, and action representation, making them mutually incomparable. This makes a basic yet unanswered question especially important: with everything else held fixed, in *what form* should robot state be represented, *where* should it be injected, and *how much history* is needed for it to genuinely improve a VLA's decision making?

Contributions. We address these open questions through controlled experimental analysis. Concretely, we take a representative flow-matching VLA as our testbed and build a unified experimental framework that fixes the backbone network, training data, action representation, and evaluation protocol; evaluation follows a layered protocol built on RoboCasa365 (Nasiriany et al., 2026), whose

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

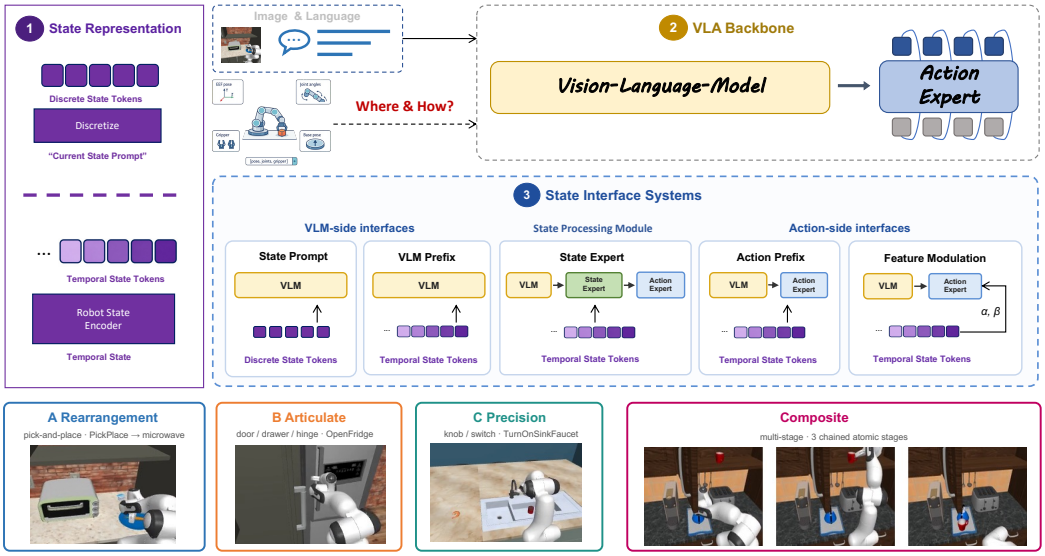

Figure 1. Overview of the state design space and evaluation suite. **(1) State representation:** robot state is represented either as a discrete current-state prompt or as continuous temporal tokens. **(2) Shared VLA scaffold:** multi-view observations and the task instruction enter the vision–language backbone, whose context conditions the action expert. **(3) State interfaces:** we compare a discrete state prompt, a continuous VLM prefix, a dedicated state expert, an action prefix, and feature modulation. The systems are evaluated on rearrangement (A), articulation (B), precision knob/switch control (C), and multi-stage composite tasks.

atomic tasks we pre-partition by control semantics into three families (pick-and-place, articulated-object interaction, and precise actuation), paired with 20 in-distribution composite tasks. Under this setup, the only thing that varies between systems is how the state is integrated, so any performance difference can be attributed to the state design itself. On this controlled basis, we make the following contributions along the three design axes.

First, for the *representation* question, we compare discrete and continuous state representations under matched conditions: the same state is either quantized and serialized into text tokens or projected into continuous embeddings, and we characterize whether, and on which tasks, each form of the current state helps.

Second, for the *temporal* question, we sweep the state history from 1 to 96 frames and, with a slot-matched repeat-current control, test whether history gains require genuine temporal content—how much history helps, and when more starts to hurt.

Finally, for the *interface* question, we distill five representative state interfaces from prior work (Figure 1)—a discrete state prompt (SP), a VLM prefix (VP), an action prefix (AP), a state expert (SE), and feature modulation (FM)—implement them in one scaffold, and compare injecting the same state into the VLM side versus the action side, showing that the preferred entry point is not fixed but switches with the temporal budget. Distilling the evidence along all three axes, we offer testable design guidelines and a reusable evaluation protocol for future state-aware VLAs.

2 Related Work

Proprioceptive interfaces in robot policies. Visuomotor policies often condition action decoders on low-dimensional robot state embeddings (Zhao et al., 2023; Chi et al., 2025). Recent VLAs instantiate three routes: discrete state in the language prompt, continuous state in the VLM sequence, and continuous state supplied to the action head (Kim et al., 2024; Black et al., 2024; Intelligence et al., 2025; Kim et al., 2025a; Bjorck et al., 2025), but they disagree on how state is encoded, where it enters, and what role it plays; Section 3.2 surveys these designs. Such systems establish that state and history conditioning is widely used, but their results are not directly comparable: each design ships inside a different system, so backbone, pretraining, data, action parameterization, and evaluation protocol all change with it. We isolate state design by moving matched numerical

content across interfaces within one policy scaffold. To our knowledge, no prior study moves the same state signal across interfaces, history depths, and injection routes with everything else fixed; that controlled comparison is the role of this paper.

Temporal state, history, and memory. Observation history can reduce partial observability and support multi-stage control; recurrent modules, memory tokens, and compressed latent states offer several ways to retain context (Bulatov et al., 2022; Dai et al., 2026; Shi et al., 2025; Li et al., 2026a). Yet history can also become a shortcut: behavioral cloning may exploit action-correlated past observations and exhibit causal confusion or copycat behavior (De Haan et al., 2019; Wen et al., 2020). Existing approaches often change the memory architecture together with history depth, and a depth sweep alone cannot separate temporal content from the extra conditioning slots that come with it. We instead vary raw state depth within a fixed interface and use a slot-matched repeat-current control to separate temporal variation from extra conditioning capacity.

Controlled analysis of VLA design. Recent systematic studies compare latent-action supervision, geometric features, and memory interfaces under shared VLA backbones (Lin et al., 2026; Yang et al., 2026; Dai et al., 2026). We extend this findings-oriented methodology to proprioception along three axes: whether state is present, how much history is retained, and where it is injected. We separate independently trained system utility from fixed-policy reliance under matched controls and anchor architectural comparisons in closed-loop performance: every claim about a design rests on independently trained systems compared under one protocol, rather than on perturbations applied to a single trained checkpoint.

3 Preliminaries

3.1 Vision–Language–Action Policies

This paper studies VLA policies that are jointly conditioned on vision, language, and robot proprioceptive state. At decision time t, the robot receives a visual observation $o_t \in \mathcal{O}$, a language instruction $\ell \in \mathcal{L}$, and an ordered state window over the most recent K steps,

$$S_t^{(K)} = (s_{t-K+1}, \ldots, s_t), \qquad s_t \in \mathbb{R}^{d_s}, \tag{1}$$

where s_t describes the robot's current kinematic configuration, such as end-effector pose, base pose, and gripper state; $K{=}1$ means that only the current step is used. Conditioned on these inputs, the policy π_θ generates a continuous action chunk of length H,

$$\hat{\mathbf{a}}_{t:t+H-1} = \pi_\theta\Big(o_t, \ell, S_t^{(K)}\Big) \in \mathbb{R}^{H \times d_a}, \tag{2}$$

where d_a is the per-step action dimension. The model is trained on a dataset of demonstration trajectories $\mathcal{D} = \{(o_t^{(i)}, \ell^{(i)}, S_t^{(i,K)}, \mathbf{a}_{t:t+H-1}^{(i)})\}$ with an action-generation objective $\mathcal{L}_{\text{act}}$ that matches the predicted action chunk to the expert action sequence. Crucially, state interfaces do not differ in this prediction target; they differ in how the state window $S_t^{(K)}$ is represented and in how it is injected into the vision–language backbone or the action-generation module.

3.2 Robot Proprioception and Its Use in VLAs

Robot proprioception—joint angles, end-effector pose, gripper state, and base or whole-body configuration—is the only policy input that lives in the same continuous physical space as the output actions, and current VLAs wire it in strikingly different ways: $\pi_{0.5}$ serializes it into discrete text tokens in the VLM prompt (Intelligence et al., 2025), OpenVLA-OFT projects it continuously into the language-model sequence (Kim et al., 2025a), and GR00T N1 feeds a state embedding directly to the action head (Bjorck et al., 2025). Around these designs, a rapidly growing line of work explores what else state can do: as alignment supervision or a contrastive regularizer for representation learning (Wen et al., 2025; Kim et al., 2025b; Zhang et al., 2026), as tokenized input that actively guides visual reasoning or routes a dedicated expert stream (Wang et al., 2026; Yu et al., 2026), as a normalized representation for cross-embodiment and whole-body control (Davies et al., 2025; Bai et al., 2026), as a continuously evolving signal for long-horizon manipulation and memory (Qi et al.,

2026; Wang et al., 2025; Shi et al., 2025), and as a cue for estimating execution risk and detecting false completion (Zhai et al., 2026; Li et al., 2026b). At the same time, several studies warn in the opposite direction: naively fused state can dominate vision, letting a policy "complete" a task from its internal progress while ignoring visual failure (Li et al., 2026b), can suppress visual learning around motion-phase transitions (Lu et al., 2026), and can serve as an action-correlation shortcut in behavioral cloning (De Haan et al., 2019; Wen et al., 2020). The community has thus explored many ways of using state, and the evidence pulls in both directions—but these findings come from different backbones, data, and protocols, which is precisely why this paper compares state interfaces, history depths, and injection routes under a single controlled scaffold.

4 METHODOLOGY

4.1 CONTROLLED POLICY SCAFFOLD

We instantiate all interfaces in $\pi_{0.5}$ (Intelligence et al., 2025), a VLM coupled to a flow-matching action expert (Lipman et al., 2023). We study state interfaces as *complete design choices* rather than isolated token-placement interventions: each interface retains the native hidden width and computation path of the module it conditions, as in practical VLAs. The backbone, observations, training data, action targets, and evaluation protocol are shared, while the state interface and history length K vary systematically. A difference in closed-loop success is therefore attributed to the state design as a whole—the interface together with its native width and computation path. Our goal is to characterize which interface design works under different control and temporal regimes, not to attribute performance differences to a single architectural primitive.

4.2 STATE REPRESENTATION

Each 16-dimensional robot-state frame contains the end-effector position and quaternion in the base frame, the mobile-base position and quaternion, and two gripper joint positions. We refer to this vector as the robot state; its content is identical across interfaces. Except for the discrete prompt, the interfaces receive the same numerical state window $S_t^{(K)}$ with the same temporal order. At the 20 Hz state rate, eight samples span 0.35 s from the oldest to the newest observation; we therefore refer to $K = 8$ as a *short state history*, not long-term episodic memory.

Each continuous frame is independently encoded into one token:

$$h_i = \phi_2(\mathrm{Swish}(\phi_1(s_i))), \qquad i = t - K + 1, \ldots, t. \tag{3}$$

The input is zero-padded from 16 to the framework's fixed 32 dimensions. The first projection lifts it to the host width d, and the second preserves that width: $d = 2048$ for the VLM prefix and $d = 1024$ for the action prefix, state expert, and feature modulation. Applying the same projector to each frame keeps the encoding procedure fixed as history grows, while each host module keeps the width and compute it has in its native design, so every interface is evaluated in the form in which it would be deployed in practice.

4.3 STATE INTERFACES

Figure 1 organizes five interfaces by how state reaches action generation. The comparison includes a native discrete interface and four continuous alternatives, not five interchangeable token placements. SP and VP inject state into the vision–language backbone, SE adds a dedicated state-processing stream, and AP and FM condition the action expert directly. Section 5.2.3 contrasts the closest VLM-side and action-side pair, VP and AP.

State Prompt (SP). The current state is quantized into 256 bins per dimension and serialized through the existing tokenizer, producing roughly 66 tokens appended to the language prompt. This route introduces no trainable parameters, but its tokens participate in VLM computation. It supports only the current frame.

VLM Prefix (VP) and Action Prefix (AP). Both routes introduce one continuous token per state frame but place it on opposite sides of the VLM–action boundary. VP appends state to the bidirectional image–language prefix, where it can alter the multimodal context before action generation. AP places state ahead of the noisy action tokens in the causal action stream, directly conditioning the action expert at every denoising step. Each keeps the native width of the module it conditions.

State Expert (SE) and Feature Modulation (FM). SE gives state a dedicated transformer stream that exchanges information with the action module. FM instead keeps state as separate conditioning memory: each action layer reads it through cross-attention and predicts a feature-wise scale γ and shift β,

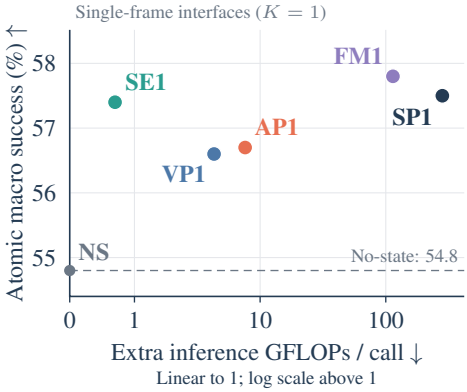

Figure 2. Single-frame macro success versus extra inference GFLOPs. Points match Table 1; the dashed line is no-state. Compute is theoretical, and macro seed spreads are unavailable.

$$\mathrm{Mod}(z; S) = (1 + \gamma(z, S)) \odot z + \beta(z, S), \tag{4}$$

where z denotes action features and $\odot$ is element-wise multiplication. These designs test a separate processing path and repeated feature conditioning, respectively, beyond prefix placement alone.

Under single-frame conditioning, SP, VP, AP, SE, and FM add 0, 4.26M, 1.08M, 199.30M, and 123.84M trainable parameters. Parameter count is not a proxy for arithmetic cost: token length, host width, and whether state computation repeats during denoising also matter. Table 1 therefore reports marginal training and inference GFLOPs alongside success. Appendix D gives additional implementation details; Appendix C derives these arithmetic counts, which are not measured latency.

5 Experiments

5.1 Experimental Protocol

We evaluate on RoboCasa365 (Nasiriany et al., 2026) with shared multi-view images, language instructions, and end-effector action targets. Visual observations come from left exterior, right exterior, and wrist cameras. All policies are initialized from the same pretrained checkpoint and fully fine-tuned. The **atomic suite** contains 45 tasks, pre-partitioned into rearrangement (A), articulation (B), and knob/switch control (C), with 15 tasks and a separately trained category policy per family. Each task has 50 evaluation rollouts. The partition is based on control semantics rather than observed performance: A emphasizes large-range positioning, B sustained contact with articulated objects, and C precise actuation in small workspaces. Navigation is outside the evaluation scope. The **composite suite** contains 20 task types with two or three subgoals, jointly trained per interface and evaluated with 25 rollouts per task. It holds out episodes, not task semantics. Success requires completing all task conditions within the horizon. The joint-angle variant changes the state coordinate system while keeping the output as end-effector deltas; it therefore checks whether the routing comparison depends on supplying state in coordinates already close to the action representation.

The principal comparisons report three-training-seed mean and standard deviation; the repeat-current control uses two seeds. The $K = 1$–96 atomic sweep is exploratory and uses one checkpoint per configuration. Evaluation seeds and action targets are matched across interfaces, and all policies share one optimization recipe, data pipeline, and learning-rate schedule. Appendix E gives the remaining training, evaluation, and uncertainty conventions; Appendices A–B report fixed-checkpoint probes and a paired case study.

We distinguish three levels of evidence. Independently trained policies measure whether an interface or history window improves closed-loop control. Fixed-checkpoint state interventions measure what an already trained policy uses. Task- or episode-bootstrap intervals describe variation over evaluation units; they do not replace the training-seed spread. Unless otherwise specified, differences in success are percentage points. We interpret small gaps as point-estimate rankings, not as significance claims inferred from overlapping or non-overlapping standard deviations.

Table 1. Single-frame interfaces: success on 45 atomic tasks and theoretical compute added over no-state. A/B/C report three-seed mean ± standard deviation; Macro is the unweighted family average without a seed spread. Parentheses show changes from NS in percentage points (green/red: gain/drop). Bold/underline mark the best/second-best success rates. Compute is per training sample or ten-step inference call, respectively.

	Atomic success rate (%) ↑				Extra GFLOPs ↓	
Interface	A Rearrangement	B Articulation	C Knob / switch	**Macro**	Train / sample	Infer / call
NS No state	$62.3^{\pm 0.8}$	$61.9^{\pm 1.0}$	$40.2^{\pm 0.8}$	54.8	0.0	0.0
SP1 State prompt	$\mathbf{67.9}^{\pm 1.1}$ (+5.6)	$65.0^{\pm 0.8}$ (+3.1)	$39.6^{\pm 0.9}$ (−0.6)	<u>57.5</u> (+2.7)	1114.0	282.0
VP1 VLM prefix	$62.6^{\pm 0.7}$ (+0.3)	$\mathbf{69.4}^{\pm 0.9}$ (+7.5)	$37.7^{\pm 0.8}$ (−2.5)	56.6 (+1.8)	16.9	4.3
AP1 Action prefix	$62.8^{\pm 1.2}$ (+0.5)	$66.7^{\pm 1.0}$ (+4.8)	<u>40.5</u>$^{\pm 1.0}$ (+0.3)	56.7 (+1.9)	3.5	7.6
SE1 State expert	$60.9^{\pm 0.9}$ (−1.4)	$67.8^{\pm 0.8}$ (+5.9)	$\mathbf{43.6}^{\pm 1.0}$ (+3.4)	57.4 (+2.6)	2.6	0.7
FM1 Feature mod.	<u>65.1</u>$^{\pm 1.0}$ (+2.8)	<u>68.9</u>$^{\pm 0.9}$ (+7.0)	$39.4^{\pm 1.1}$ (−0.8)	**57.8** (+3.0)	45.4	114.0

5.2 Results and Analysis

5.2.1 RQ1: Does Current State Improve Control?

Current state helps overall, but the best interface is task-dependent. All five single-frame interfaces exceed the no-state macro of 54.8% by 1.8–3.0 percentage points (Table 1), yet no interface leads every family. SP leads rearrangement at 67.9%, VP leads articulation at 69.4%, and SE leads knob/switch control at 43.6%, a family in which SP, VP, and FM even fall below no-state.

The contrast between SP and SE shows why the macro alone is insufficient. Their macro scores differ by only 0.1 point, yet SP gains 5.6 points over no-state on A and loses 0.6 on C, whereas SE loses 1.4 on A and gains 3.4 on C. Similarly, VP's 7.5-point gain on B coexists with a 2.5-point drop on C. The value of an interface thus depends on what the task requires from state, so we report control regimes separately rather than rely on a single macro ranking.

Better control performance does not necessarily come with lower inference cost. Figure 2 shows that success and computation are largely decoupled. The two highest macro means, FM (57.8%) and SP (57.5%), add 114 and 282 inference GFLOPs per call, whereas SE reaches a similar 57.4% with only 0.7. Parameter count does not track this cost either: SP adds no parameters but the most computation, because its prompt tokens pass through the VLM, while SE adds 199.30M parameters and the least. The highest success estimate, fewest parameters, and least extra computation therefore select different interfaces; a compute-constrained choice must weigh all three, with actual latency and memory measured on the deployment platform.

5.2.2 RQ2: How Much History Helps?

Short state history is beneficial, while longer history can degrade performance. The atomic depth sweep shows a bounded benefit from short history (Figure 3); longer raw sequences do not consistently help. Averaged over families, the change from the single-frame model peaks in the short-window range and turns negative for windows of 48 frames or more. Families A and B are comparatively tolerant of increased context, whereas C, which demands precise local control, degrades more strongly as the raw sequence grows. One possible account is that older state values become less aligned with the immediate action needed for precise control. This is an interpretation of the trend, not an identified mechanism: the sweep does not separately manipulate staleness, temporal resolution, and token count. We therefore use $K = 8$ as a practical short-history setting, not a universal optimum. On composite tasks, this window improves AP, SE, and FM, but not the EEF-state VP configuration (Table 2). A depth curve compares history lengths within an interface, whereas Table 2 tests selected endpoints with repeated training; keeping these roles separate avoids presenting every fluctuation in the exploratory sweep as a reproducible optimum. The operational

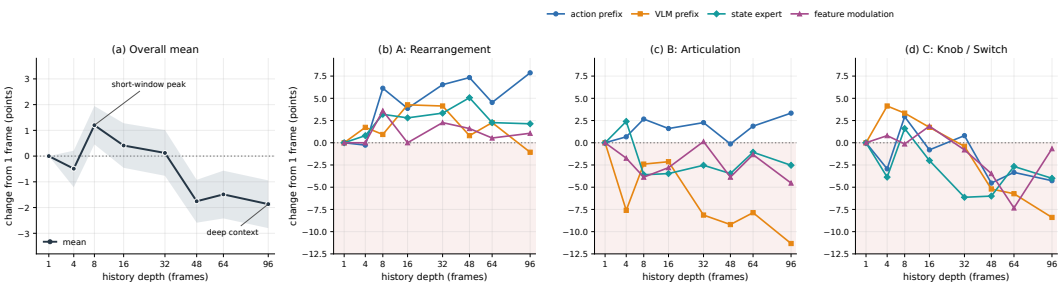

Figure 3. Exploratory single-checkpoint history-depth sweep on atomic tasks. (a) Overall change from the matched single-frame model; (b–d) task-family changes. Short histories can help, while long histories particularly harm family C. The sweep does not quantify training-seed uncertainty.

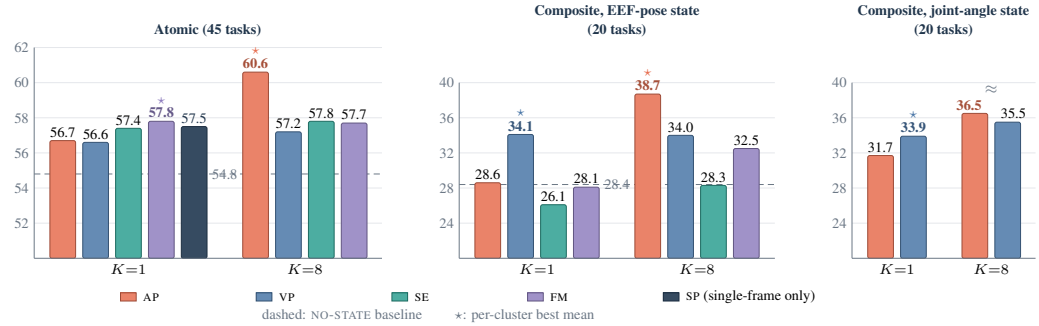

Figure 4. Interface–history summary (macro SR, %). Dashed lines are no-state baselines. Atomic $K = 1$ and composite values are the three-seed means in Tables 1–2. Atomic $K = 8$ endpoints are derived from the $K = 1$ means plus single-checkpoint depth-sweep increments, not measured three-seed results.

lesson is to validate a compact window for each interface rather than assume that a longer input sequence preserves or improves control.

The gain comes from temporal information, not merely additional input capacity. A longer window also adds conditioning slots, so we train a slot-matched control that repeats the current state in every slot. It keeps the interface, slot count, and slot positions of AP_8 and changes only whether the slots carry distinct recent states. Replacing repeated current state by ordered history raises AP success from $29.2^{\pm 2.3}$ to $39.1^{\pm 0.1}$ across two seeds, a 9.9-point difference. These are separately trained, slot-matched systems, so the comparison addresses the utility of temporal content during learning, not merely a trained model's response to deleting information at test time. The two-seed control and three-seed endpoint comparison are distinct runs; their AP8 means should not be pooled. The gain therefore requires variation across recent states rather than additional conditioning slots; we interpret AP_8 as using recent kinematic evolution, without attributing it to any specific temporal statistic. The history gain also holds with joint-angle state under the same end-effector action space: 4.8 points for AP and 1.6 for VP, compared with 10.1 and -0.1 for EEF-pose state. Changing the state coordinates thus preserves the directional routing crossover but changes its magnitude, so coordinates and history should be checked together at design time.

History matters most at stage transitions. A paired case study on PREPARETOAST localizes where the composite gain arises (Figure 5; Appendix B). Over 50 paired episode seeds, AP_1 and AP_8 progress similarly while placing the two items (S1: 90% versus 96%; S2: 64% versus 68%) but separate when control must switch back to an earlier target and close the cabinet: S3 rises from 30% to 56%, and the same 26-point gap persists to final success (paired episode-bootstrap 95% interval $[+10, +42]$). The fixed AP_8 checkpoint shows the same timing. Replacing ordered history by repeated current state, with images, flow noise, and slot count held fixed, changes the action most at milestone boundaries, 5.2 times the within-stage change. The change grows from 0.20 within a stage to 0.36 just before a switch, peaks at 1.03 at the boundary, and remains elevated at 0.75 afterward. The policy thus relies on recent state most where the task changes stage. This co-

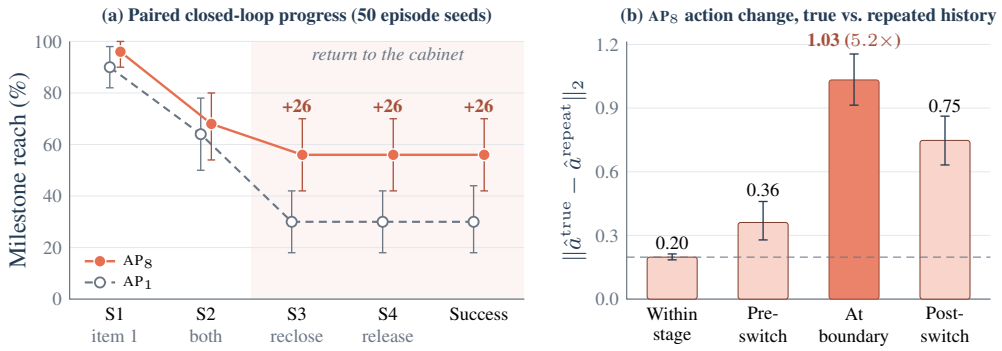

Figure 5. Paired PREPARETOAST case study. (a) AP_1 and AP_8 reach the two item placements (S1–S2) at similar rates but separate once control must return to reclose the cabinet (S3, shaded); AP_8 then stays 26 points ahead through final success. (b) Within the fixed AP_8 checkpoint, replacing ordered history with repeated current state changes the action most at native milestone boundaries; the dashed line marks the within-stage level. Error bars are episode-bootstrap 95% intervals.

occurrence is descriptive evidence of temporal use on one task; it is not a causal mediation analysis of the closed-loop success gain.

5.2.3 RQ3: WHERE SHOULD STATE ENTER?

Current state favors VLM-side injection, while short state history favors action-side injection. On EEF-state composites, VP leads AP at $K = 1$ (34.1% versus 28.6%), but AP leads at $K = 8$ (38.7% versus 34.0%). Joint-angle state shows the same directional crossover, with a smaller $K = 8$ gap (36.5% versus 35.5%). The joint-angle reversal is smaller, so the common direction, not its magnitude, is the finding. We do not infer a significance level from these mean differences alone. The question answered is where state should enter when each route is instantiated in its native form, as it would be deployed. Appendix F explains the derivation of the atomic endpoints in Figure 4.

In Figure 4, the atomic single-frame leaders lie within 0.4 points of one another, and the atomic $K = 8$ endpoints are a derived summary rather than multi-seed measurements. We therefore base the routing conclusion on the directly measured composite endpoints.

The scope of the recommendation is consequently conditional: when only the current frame is available, VP is the stronger composite prefix baseline in these experiments; when a compact ordered history is available, AP is the stronger prefix baseline. That statement does not rank all five interfaces on every task, nor does it predict what would happen if state tokens were moved between modules without retraining. The comparison evaluates complete trained systems.

Table 2. Composite SR (%). Main rows: three-seed mean ± SD. Last row: repeat-current-8 versus AP8 (two seeds), not $K = 1$ versus $K = 8$. ΔSR is in percentage points.

Interface	K=1	K=8	ΔSR
EEF-pose state			
AP	$28.6^{\pm 0.6}$	$38.7^{\pm 0.5}$	+10.1
VP	$34.1^{\pm 0.5}$	$34.0^{\pm 0.4}$	−0.1
SE	$26.1^{\pm 0.5}$	$28.3^{\pm 0.5}$	+2.2
FM	$28.1^{\pm 0.5}$	$32.5^{\pm 0.5}$	+4.4
Joint-angle state			
AP	$31.7^{\pm 0.5}$	$36.5^{\pm 0.5}$	+4.8
VP	$33.9^{\pm 0.5}$	$35.5^{\pm 0.5}$	+1.6
Slot-matched control (EEF; two seeds)			
Repeat-current-8 → AP8	$29.2^{\pm 2.3}$	$39.1^{\pm 0.1}$	+9.9

Fixed-checkpoint probes. Matched true-state and state-off forward passes over 180 cached contexts are consistent with this data-flow reading (Appendix A). VLM-side state changes the multimodal prefix before action generation: across the last six VLM layers, VP_1 redistributes language-to-image attention by 17.3%, whereas AP_8 leaves the prefix unchanged because its state enters afterward. On the action side, ordered history strengthens the state-conditioned correction along the flow trajectory: from AP_1 to AP_8, final alignment with the expert residual rises from 0.079 to 0.270 (task-bootstrap 95% interval of the increase $[+0.143, +0.239]$). These probes describe what trained checkpoints use; they do not replace the closed-loop comparisons.

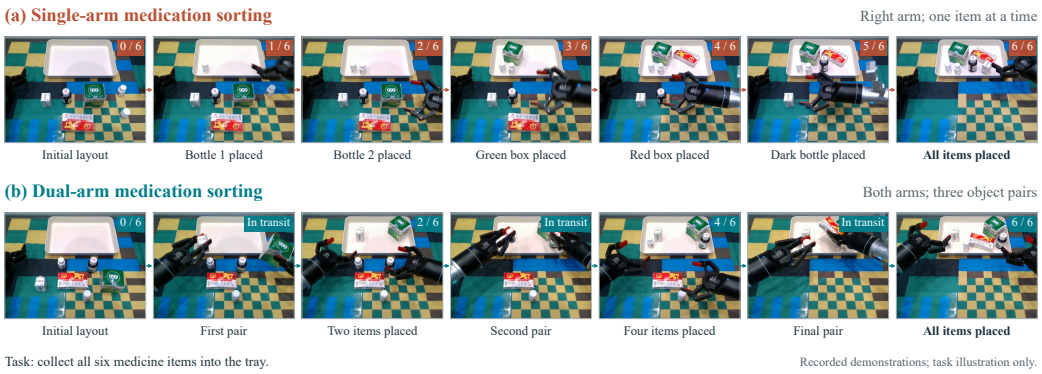

Figure 6. Real-robot medication-sorting tasks. **(a)** Single-arm: the right arm places six medicine items into the tray one at a time. **(b)** Dual-arm: both arms transfer the items as three pairs. Counters give the number of items in the tray. Frames come from recorded demonstrations and illustrate the tasks; they are not policy rollouts.

5.3 Real-Robot Validation

Physical execution tests a boundary that the simulated comparison cannot: state comes from onboard sensing, with the noise, latency, and contact variability that simulator state may not capture. We therefore ask whether the configuration favored on simulated composites, AP_8, still improves on a no-state policy when trained and deployed on a real robot. We use two multi-stage medication-sorting tasks (Figure 6). In the single-arm task, the right arm places six medicine items into a tray one at a time. In the dual-arm task, both arms place the same six items as three pairs. For each task, we collect 150 demonstrations with randomized item placement and fine-tune NO-STATE and AP_8 from the same pretrained $\pi_{0.5}$ checkpoint under one fixed recipe. Each of the four policies is evaluated in 10 trials, with item placement re-randomized for every trial. The success criterion is strict: a trial succeeds only if all six items reach the tray in one uninterrupted attempt. Re-grasping after a missed grasp, any human correction, or disturbing an item still on the table counts as failure. Items that tip over after they are already inside the tray do not count as failure.

AP_8 succeeds in 10/10 single-arm trials versus 8/10 for NO-STATE, and in 7/10 dual-arm trials versus 4/10. Both gaps point the same way as the simulated composite result, and the larger gap is on the task that coordinates two arms. The evidence is nonetheless small. With ten trials per policy, the 95% Wilson intervals overlap substantially, and neither per-task difference is significant (two-sided Fisher exact $p = 0.47$ and 0.37). The check deploys the configuration favored in simulation, state with an eight-frame history, against no state. It confirms that this design remains beneficial with onboard state; the interface ranking and the repeat-current control come from simulation. Appendix G gives the full training and evaluation protocol.

6 Conclusion and Outlook

Within the evaluated VLA scaffold, current state helps overall but the best interface depends on task structure. A short state history improves composite control beyond repeated-state conditioning, while long raw history can hurt. The VLM-to-action routing crossover suggests choosing the state interface jointly with the temporal budget, rather than treating either as a fixed architectural default. The practical implication is to evaluate the interface against the target control regime and compare a short history with a slot-matched current-state baseline before extending the context further. On two real-robot sorting tasks, AP_8 outperformed the no-state policy in the same direction, although ten trials per policy do not establish a significant difference. Robot proprioception should also cover tactile and force signals and the robot's physical structure, which we have not yet explored.

These findings are stated at the level of deployable interface designs. A natural next layer is component-level analysis, asking which state dimensions and which statistics of recent state variation each design exploits. Such questions refine rather than qualify the present conclusions; Appendix F.2 states the scope of each claim.

AI USE STATEMENT

The authors used generative AI tools to polish the language, draft portions of the manuscript, explore the literature in a preliminary way, assist with code, organize and present experimental data and figures, and give feedback on the methodology and manuscript drafts. This assistance included routine statistical calculations for the real-robot trials, such as the confidence intervals and exact tests. The authors reviewed and verified every AI-assisted output, including literature suggestions and all numerical and visual material. The authors take full responsibility for the study's ideas, methodology, experiments, analysis, claims, and final text.

REFERENCES

Shuanghao Bai, Meng Li, Xinyuan Lv, Jiawei Wang, Xinhua Wang, Fei Liao, Chengkai Hou, Langzhe Gu, Wanqi Zhou, Kun Wu, Ziluo Ding, Zhiyuan Xu, Lei Sun, Shanghang Zhang, Zhengping Che, Jian Tang, and Badong Chen. HEX: humanoid-aligned experts for cross-embodiment whole-body manipulation. *CoRR*, abs/2604.07993, 2026. doi: 10.48550/ARXIV.2604.07993. URL https://doi.org/10.48550/arXiv.2604.07993.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

Ziyang Chen, Shaoguang Wang, Weiyu Guo, Qianyi Cai, He Zhang, Pengteng Li, Yiren Zhao, and Yandong Guo. Phaser: Phase-aware and semantic experience replay for vision-language-action models, 2026.

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. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

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. *arXiv preprint arXiv:2603.04639*, 2026.

Travis Davies, Yiqi Huang, Yunxin Liu, Xiang Chen, Huxian Liu, and Luhui Hu. Tenma: Robust cross-embodiment robot manipulation with diffusion transformer. *arXiv preprint arXiv:2509.11865*, 2025.

Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in neural information processing systems*, 32, 2019.

Weiyu Guo, He Zhang, Pengteng Li, Tiefu Cai, Ziyang Chen, Yandong Guo, Xiao He, Yongkui Yang, Ying Sun, and Hui Xiong. A brain-inspired embodied intelligence for fluid and fast reflexive robotics control. *arXiv preprint arXiv:2601.14628*, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025a.

Taeyoung Kim, Jimin Lee, Myungkyu Koo, Dongyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. Contrastive representation regularization for vision-language-action models. *CoRR*, abs/2510.01711, 2025b. doi: 10.48550/ARXIV.2510.01711. URL https://doi.org/10.48550/arXiv.2510.01711.

Pengteng Li, Weiyu Guo, He Zhang, Tiefu Cai, Xiao He, Yandong Guo, and Hui Xiong. Spatial memory for out-of-vision manipulation in vision-language-action. *arXiv preprint arXiv:2605.22283*, 2026a.

Zhuohao Li, Yinghao Li, Jian-Jian Jiang, Lang Zhou, Tianyu Zhang, and Wei-Shi Zheng. Revip: Reducing false completion in vision-language-action models with vision-proprioception rebalance. *CoRR*, abs/2601.16667, 2026b. doi: 10.48550/ARXIV.2601.16667. URL https://doi.org/10.48550/arXiv.2601.16667.

Yihan Lin, Haoyang Li, Yang Li, Haitao Shen, Yihan Zhao, Chao Shao, and Jing Zhang. From pixels to tokens: A systematic study of latent action supervision for vision-language-action models. *arXiv preprint arXiv:2605.04678*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023.

Jingxian Lu, Wenke Xia, Yuxuan Wu, Zhiwu Lu, and Di Hu. When would vision-proprioception policies fail in robotic manipulation? *CoRR*, abs/2602.12032, 2026. doi: 10.48550/ARXIV.2602.12032. URL https://doi.org/10.48550/arXiv.2602.12032.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. *arXiv preprint arXiv:2603.04356*, 2026.

Xiuxiu Qi, Yu Yang, Jiannong Cao, Luyao Bai, Chongshan Fan, Chengtai Cao, and Hongpeng Wang. Continuous vision-language-action co-learning with semantic-physical alignment for behavioral cloning. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 24900–24908. AAAI Press, 2026. doi: 10.1609/AAAI.V40I29.39677. URL https://doi.org/10.1609/aaai.v40i29.39677.

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. *arXiv preprint arXiv:2508.19236*, 2025.

Fangyuan Wang, Peng Zhou, Jiaming Qi, Shipeng Lyu, David Navarro-Alarcon, and Guodong Guo. Think proprioceptively: Embodied visual reasoning for VLA manipulation. *CoRR*, abs/2602.06575, 2026. doi: 10.48550/ARXIV.2602.06575. URL https://doi.org/10.48550/arXiv.2602.06575.

Xiaofan Wang, Xingyu Gao, Jianlong Fu, Zuolei Li, Dean Fortier, Galen Mullins, Andrey Kolobov, and Baining Guo. Lola: Long horizon latent action learning for general robot manipulation. *CoRR*, abs/2512.20166, 2025. doi: 10.48550/ARXIV.2512.20166. URL https://doi.org/10.48550/arXiv.2512.20166.

Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories. *Advances in Neural Information Processing Systems*, 33:2564–2575, 2020.

Yuqing Wen, Kefan Gu, Haoxuan Liu, Yucheng Zhao, Tiancai Wang, Haoqiang Fan, and Xiaoyan Sun. Rosa: Harnessing robot states for vision-language and action alignment. *arXiv preprint arXiv:2506.13679*, 2025.

Yurou Yang, Muyuan Lin, Roberto Martin-Martin, Martin Labrie, Shreekant Gayaka, Cheng-Hao Kuo, and Luca Carlone. Understanding the impact of geometric foundation models on vision-language-action models. *arXiv preprint arXiv:2605.24642*, 2026.

Bin Yu, Shijie Lian, Xiaopeng Lin, Yuliang Wei, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Xinming Wang, Bailing Wang, Cong Huang, and Kai Chen. Twinbrainvla: Unleashing the potential of generalist vlms for embodied tasks via asymmetric mixture-of-transformers. *CoRR*, abs/2601.14133, 2026. doi: 10.48550/ARXIV.2601.14133. URL https://doi.org/10.48550/arXiv.2601.14133.

Xuanran Zhai, Binkai Ou, Yemin Wang, Hui Yi Leong, Qiaojun Yu, Ce Hao, and Yaohua Liu. Cofreevla: Collision-free dual-arm manipulation via vision-language-action model and risk estimation. *arXiv preprint arXiv:2601.21712*, 2026.

Chubin Zhang, Jianan Wang, Zifeng Gao, Yue Su, Tianru Dai, Cai Zhou, Jiwen Lu, and Yansong Tang. CLAP: contrastive latent action pretraining for learning vision-language-action models from human videos. *CoRR*, abs/2601.04061, 2026. doi: 10.48550/ARXIV.2601.04061. URL https://doi.org/10.48550/arXiv.2601.04061.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://doi.org/10.15607/RSS.2023.XIX.016.

APPENDIX

Appendices A–C contain fixed-checkpoint probes, a paired case study, and the compute derivation. Appendices D–G provide interface implementations, the full experimental protocol, supplementary comparisons and scope, and real-robot experimental details.

A. PROBE EXPERIMENTS

The main experiments reveal a routing crossover: a current state frame is most effective through a VLM-side interface, whereas an ordered short history is used most reliably through the action prefix. We connect these system-level results to three matched fixed-checkpoint probes. The probes trace where state first changes the computation and how that change reaches the generated action; they complement, rather than replace, closed-loop evaluation.

A.1 MATCHED STATE INTERVENTIONS

For each of the 45 atomic tasks, we cache four expert-action contexts and evaluate the same image, instruction, action target, and flow-noise sample under *true-state* and *state-off* forward passes, giving 180 matched contexts per interface. Continuous state is disabled by zeroing its projected embedding while preserving token positions and masks. For the discrete state prompt, we use both a zero-state serialization and a masked state span.

For VLM-prefix probes, language-to-image attention redistribution is the total-variation distance between normalized image-patch distributions under true state and state-off. Image-token representation change is the relative ℓ_2 distance between their hidden states. For action-flow probes, let

$$c_t = \hat{a}_t^{\text{true}} - \hat{a}_t^{\text{off}}, \qquad r_t = a^\star - \hat{a}_t^{\text{off}}, \tag{5}$$

where c_t is the state-conditioned correction at Euler step t and r_t is the residual from state-off to the expert action. We report correction direction $\cos(c_t, r_t)$ and normalized magnitude $\|c_t\|_2/\|r_t\|_2$.

A.2 CURRENT STATE HAS AN EARLY VLM-SIDE PATH

Across the final six VLM layers, the 45-task mean language-to-image attention redistribution is 17.3% for VP_1 and 22.0% for VP_8; the corresponding image-token relative ℓ_2 changes are 19.6% and 26.2%. Both responses grow toward later layers. Disabling AP_8 leaves these VLM-prefix quantities unchanged because its state tokens enter only after the prefix has been formed. Thus, VLM-side state has an early contextualization route, while action-prefix state must influence generation directly inside the action expert.

A.3 STATE-CONDITIONED CORRECTION ACCUMULATES DURING GENERATION

Every interface produces a correction that grows as noise is transformed into an action, but route and history depth determine how strongly the correction aligns with the expert residual. With one continuous state frame, the final alignment is 0.245 for VP_1 and 0.079 for AP_1; normalized magnitudes are 0.297 and 0.174. Ordered short history changes the action-side response: moving from AP_1 to AP_8 raises final alignment from 0.079 to 0.270 and normalized magnitude from 0.174 to 0.382. Across the 45 paired tasks, the eight-minus-one increases are $+0.191$ for alignment and $+0.208$ for magnitude, with task-bootstrap 95% intervals $[+0.143, +0.239]$ and $[+0.171, +0.244]$.

A.4 REPRESENTATIVE ACTION-CONDITIONED SPATIAL ATTENTION

The VLM-prefix state visibly redistributes the image regions read by action queries. The one-frame action prefix changes little spatially, while ordered short history increases its downstream response without rewriting the VLM prefix. This example is illustrative; the quantitative layer-wise and flow-trajectory results above aggregate all 45 tasks. Together, the probes support a data-flow interpretation of the main result: current VLM-side state can alter the multimodal context before action generation, whereas temporal action-side state acts directly through the action expert.

B. Case Study

The aggregate experiments establish that compact state histories can improve composite control, particularly through the action-prefix route. We use PrepareToast to identify where this advantage appears within an episode and when a fixed policy responds to genuine temporal variation.

B.1 Task and Paired Protocol

PrepareToast requires the policy to place two task-relevant items and then return to the cabinet to close it. We compare independently trained one-frame and eight-frame action-prefix policies, AP_1 and AP_8, using 50 paired episode seeds. Pairing fixes the scene, object instances, object placements, and simulator randomization. Four monotonic milestones are derived from native task predicates and summarized in Table 3.

Within the fixed AP_8 checkpoint, we also compute a non-executed counterfactual action after replacing the ordered eight-frame history by eight copies of the current state. Images, language, parameters, state-slot count, and initial flow noise are held fixed. The resulting distance

$$D_q = \left\| \hat{\mathbf{a}}_q^{\text{true}} - \hat{\mathbf{a}}_q^{\text{repeat}} \right\|_2 \tag{6}$$

measures fixed-policy sensitivity to temporal variation rather than to the number of state slots.

B.2 Where the Advantage Appears

The two policies progress similarly through the early placement stages: AP_1 reaches S1 and S2 in 90% and 64% of episodes, while AP_8 reaches them in 96% and 68%. The separation begins after both items have been placed. AP_1 reaches the cabinet-reclosed milestone S3 in 30% of episodes, whereas AP_8 reaches it in 56%. The same $+26$-point difference persists through S4 and final success, with a paired episode-bootstrap interval of $[+10, +42]$ points. Conditional on reaching S2, completion of S3 rises from 46.9% to 82.4%.

B.3 The Policy Reads Genuine Temporal Variation

The mean true-history versus repeat-current action distance is 0.198 within a stage, 0.361 before a boundary, 1.033 at a boundary, and 0.748 afterward. Boundary sensitivity is therefore 5.2 times the within-stage value. Because every paired forward pass shares the same image, instruction, state-slot count, and flow noise, this contrast cannot be attributed to more tokens or a different visual trajectory. The two analyses jointly localize the benefit to the late placement-to-closure transition and show that the trained policy is most sensitive to ordered state variation around progress boundaries. Their co-occurrence is descriptive evidence of temporal use, not a causal mediation claim.

B.4 Case-Study Takeaway

The paired analysis separates two questions that an aggregate success rate alone cannot resolve. The independently trained AP_1–AP_8 comparison measures system-level temporal utility: the performance separation appears only after the two placement subgoals have been completed and control must return to the cabinet. The fixed-checkpoint true-history–repeat-current comparison measures temporal reliance: the eight-frame policy changes its action most strongly near the native progress boundaries while images, parameters, flow noise, and state-slot count remain matched. The two observations therefore agree at the behavioral level: ordered recent evolution is read most strongly

Table 3. Native PrepareToast milestones. Each predicate is monotonic and defined independently of either policy's outcome.

Stage	Native predicate
S1	The first task-relevant item has been placed.
S2	Both task-relevant items have been placed.
S3	The cabinet has been reclosed.
S4	The gripper has been released after completion.

where the active subgoal changes, and the trained history policy is more likely to complete the corresponding late transition. We retain the distinction between association and mediation; the case study localizes a consistent use pattern but does not claim that boundary sensitivity is the sole cause of the success gain.

Several design choices make this localization auditable. The four milestones are monotonic predicates supplied by the task definition, rather than stages inferred from either policy's actions or final outcome. Paired episode seeds hold the scene and simulator randomization fixed for the system-level comparison, so each reach-rate difference refers to the same set of task instances. The fixed-policy intervention then removes a different alternative explanation: repeating the current state preserves the number and positions of all state tokens, so the measured action change is tied to their temporal content rather than additional conditioning capacity. These two controls operate at complementary levels. The first asks whether a policy trained with recent history progresses farther through the same episodes; the second asks whether that trained policy changes its prediction when only the order-dependent state content is removed.

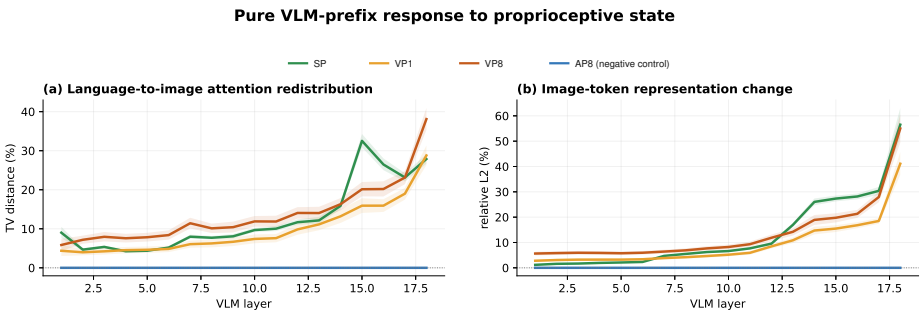

Figure 7. VLM-prefix response to proprioceptive state. Layer-wise language-to-image attention redistribution and image-token representation change are measured between matched true-state and state-off passes. VLM-side state progressively alters the shared multimodal prefix, whereas action-prefix state leaves that prefix unchanged and acts downstream through action generation.

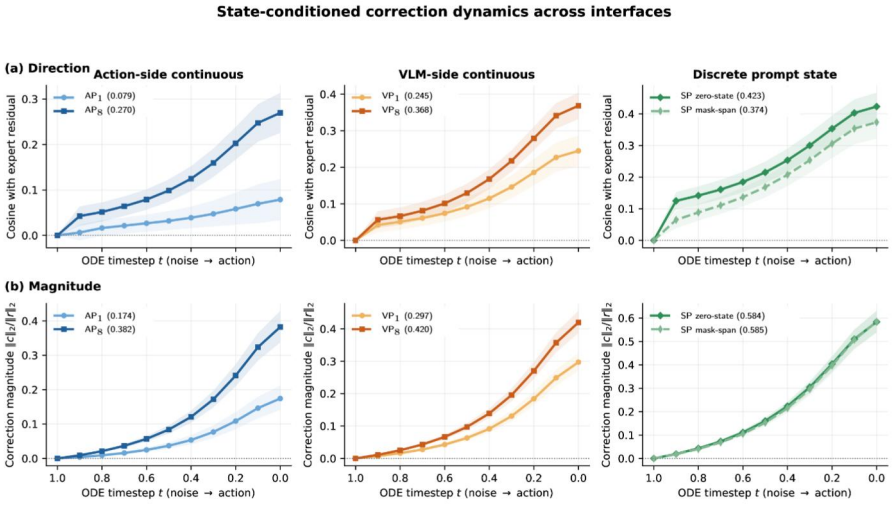

Figure 8. State-conditioned correction over the ten-step flow trajectory. The top row reports alignment between the true-minus-off correction and the state-off-to-expert residual. The bottom row reports correction magnitude normalized by that residual. Columns compare action-side continuous state, VLM-side continuous state, and discrete prompt state. Curves average 45 tasks; bands are task-bootstrap 95% intervals.

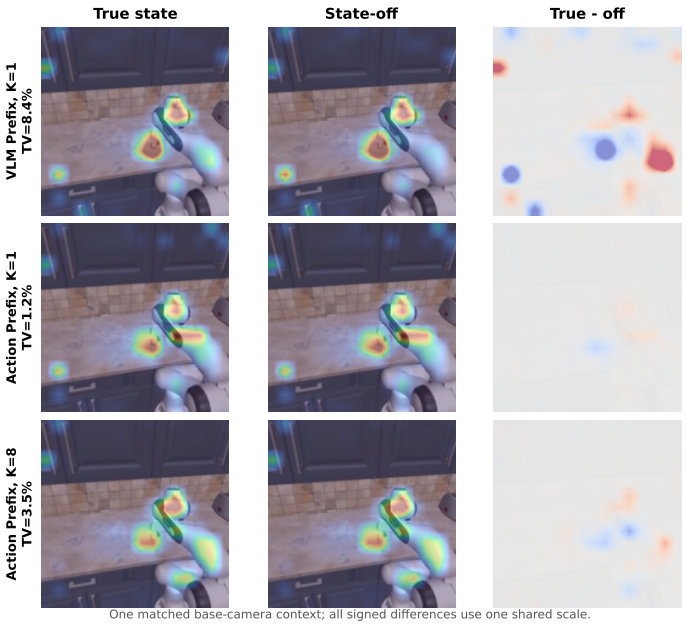

Figure 9. Representative action-conditioned spatial attention. Matched true-state and state-off passes show how a one-frame VLM prefix and one- and eight-frame action prefixes redistribute action-conditioned visual attention. The visualization is descriptive: it reveals where state changes the spatial readout, while the flow probe in Figure 8 measures how that change reaches action generation.

C. Detailed Compute-Cost Derivation

This section expands the cost analysis summarized in the main paper. The purpose is to compare the *semantic marginal arithmetic* introduced by each state interface relative to a token-free no-state policy. Consequently, the calculation charges State Prompt for its 66 serialized state tokens even when an implementation pads both state and no-state prompts to the same static tensor shape. The estimates are hardware independent and should not be interpreted as wall-clock latency or peak memory.

C.1 Shared Backbone and Counting Convention

The analytic model follows the evaluated $\pi_{0.5}$ scaffold. It contains $L = 18$ Transformer layers, three image streams with 256 tokens each, 16 instruction tokens, and $A = 50$ action tokens. Thus, before state is added, the conditioning prefix has

$$P = 3 \times 256 + 16 = 784 \tag{7}$$

active tokens. The VLM width is $d_v = 2048$ with an FFN width of 16384; the action expert width is $d_a = 1024$ with an FFN width of 4096. The grouped attention inner width is $H = 8 \times 256 = 2048$, and policy inference uses $N = 10$ Euler denoising steps.

We count one multiply-add as two floating-point operations. The marginal forward cost of processing one additional token through all Transformer layers is approximated by

$$F_{\text{tok}}(d, m) = 2L[d(H + 2{\times}256 + H) + 3dm]\,, \tag{8}$$

where d and m are the host module's hidden and FFN widths. A shared two-layer state projector, $32 \to d \to d$, contributes

$$F_{\text{enc}}(d, K) = 2K(32d + d^2) \tag{9}$$

for K state frames. Because the same projector is reused frame by frame, its parameter count is independent of history depth.

C.2 Attention Expansion by Injection Route

Adding K bidirectional state tokens to the VLM prefix enlarges prefix and prefix-to-action attention by

$$\Delta F_{\mathrm{VP,attn}} = 4HL\left(2PK + K^2 + AK\right). \tag{10}$$

Placing the same tokens in the causal action suffix instead gives

$$\Delta F_{\mathrm{AP,attn}} = 4HL\left(PK + \frac{K(K+1)}{2} + AK\right). \tag{11}$$

When a state token has already been encoded and cached in the prefix, each Euler step pays an additional suffix-to-prefix attention cost

$$\Delta F_{\mathrm{cache}} = 4HLAK. \tag{12}$$

These expressions are added to the corresponding state-encoder and per-token Transformer costs. State Prompt uses the VLM-prefix expression with 66 serialized tokens and no additional learned parameters. Feature Modulation uses no ordinary sequence tokens; its cost instead comprises one state-encoder pass plus per-layer cross-attention and scale/shift projection.

C.3 Training and Ten-Step Inference

The reported training column approximates executed arithmetic as

$$\Delta F_{\mathrm{train}} \simeq 4\,\Delta F_{\mathrm{forward}}, \tag{13}$$

because the Transformer blocks are rematerialized during backpropagation. This factor accounts for the forward evaluation and the recomputation needed by the backward pass; it is an arithmetic convention, not a measured timing multiplier. During inference, VLM-prefix state is encoded once and reused through the ten denoising steps, whereas action-prefix processing and feature-modulation operations inside the action expert are repeated at each step. This difference explains why Action Prefix has low marginal training cost but a larger ten-step inference increment than VLM Prefix.

C.4 Interpretation

The derivation separates three notions that are easy to conflate. Active token count determines sequence-level arithmetic, extra parameters describe learned capacity, and measured throughput additionally depends on padding, compilation, rematerialization, and hardware scheduling. State Prompt, for example, adds no parameters but is analytically expensive because 66 semantic tokens traverse the VLM. Its near-baseline measured throughput remains compatible with this estimate because the implementation uses a fixed padded prompt shape. Conversely, State Expert and Feature Modulation add substantial learned capacity without adding many ordinary sequence tokens.

Temporal depth must likewise be considered jointly with injection route. VLM-prefix training cost grows quickly because every additional state token participates in the large VLM prefix, whereas action-prefix inference repeats state-token processing across all denoising steps. The analysis therefore does not identify a universally cheapest interface; it makes explicit which resource each route consumes and provides the counting basis for the main paper's performance–efficiency comparison.

D Interface and State-Encoding Details

D.1 State Encoding

Each raw state frame $s_q \in \mathbb{R}^{16}$ contains the end-effector position and quaternion in the base frame, the mobile-base position and quaternion in the world frame, and two gripper joint positions. We refer to this vector as the robot state; it combines arm, base, and gripper kinematics and is identical across interfaces.

Except for the discrete state prompt, which uses only the current frame, all continuous interfaces receive state sequences with identical numerical content, temporal order, and history length. Each frame is passed independently through a two-layer projector to form one continuous state token,

$$h_i = \phi_2(\mathrm{Swish}(\phi_1(s_i))), \tag{14}$$

where ϕ_1 lifts the state input, zero-padded from 16 to the framework's fixed 32 dimensions, to width d, and ϕ_2 preserves that width, with d=2048 for the VLM prefix and d=1024 for the action prefix, the state expert, and feature modulation, matching the hidden width of the host module in each case. Sharing the per-frame projector guarantees a consistent encoding process across history depths, but it does not equalize the total parameter count or compute across interfaces; these differences are reported in Table 1 and illustrated in Figure 2.

D.2 Interface Implementations

State Prompt (SP). This interface reuses the native text entry of $\pi_{0.5}$: each dimension of the current state is quantized into 256 bins and serialized through the existing tokenizer, producing roughly 66 prompt tokens that join the language instruction in the VLM input. Because the bins map onto the pretrained vocabulary, this route adds no trainable parameters and is the only interface whose state tokens pass through the same embedding space as language; by construction it supports only the current frame.

VLM Prefix (VP). Per-frame state tokens (d=2048) are inserted into the bidirectional VLM prefix, after the image and language tokens. State first participates in multimodal context modeling—every image and language token can attend to it—and then influences action generation indirectly through the conditioning prefix.

Action Prefix (AP). State tokens (d=1024) are placed in the causal action suffix, ahead of the noisy action tokens, so that they participate directly in the action expert's velocity-field prediction at every denoising step without first being compressed into the VLM representation. This is the most direct route from state to action.

State Expert (SE). A dedicated state-processing stream is added alongside the VLM and the action expert, giving state its own sequence-modeling path that exchanges information with the action module during generation. This is the largest-capacity design: state is neither compressed into the VLM nor folded into the action suffix, but processed by its own transformer stack.

Feature Modulation (FM). State is kept as a separate conditioning memory rather than as ordinary sequence tokens. Each layer of the action expert reads it through cross-attention and predicts a per-feature scale γ and shift β that continually modulate the action features:

$$\mathrm{Mod}(z; S) = (1 + \gamma(z, S)) \odot z + \beta(z, S). \tag{15}$$

Under single-frame conditioning, SP, VP, AP, SE, and FM add 0, 4.26M, 1.08M, 199.30M, and 123.84M trainable parameters, respectively. SE and FM thus change both topology and capacity, so the five interfaces jointly answer which *complete system* is effective under a given task structure; the closest route pair (VP versus AP) is examined in Section 5.2.3.

E Detailed Experimental Protocol

Benchmark. We evaluate the effect of proprioceptive state interfaces and state history on closed-loop robot control on RoboCasa365 (Nasiriany et al., 2026), which builds on the RoboCasa simula-

tion platform and contains diverse kitchen scenes, robot initial configurations, object instances, and manipulation tasks. All methods use the same visual inputs, language instructions, action space, and evaluation program; visual observations consist of three views from a left exterior camera, a right exterior camera, and a wrist camera.

Atomic benchmark. Atomic tasks assess single-stage control. We pre-partition the RoboCasa365 atomic tasks by their dominant manipulation semantics—not post hoc by model performance—into three families: A (rearrangement and pick-and-place, probing large-range positioning of the end effector and mobile base), B (articulated-object interaction, probing sustained contact and motion-phase modeling), and C (knob, switch, and appliance control, where small workspaces demand high local precision). Navigation is out of scope. Each family contains 15 representative tasks (45 in total) and trains a separate category expert, so cross-family results reflect interface behavior under different control demands rather than one unified multi-task policy. Every task is evaluated with 50 closed-loop rollouts under a fixed, identical task list and episode-seed schedule.

Composite benchmark. Composite evaluation uses the `lifelong_learning_phase2` setting: 20 task types, each chaining two or three atomic subgoals within one episode. Each interface trains one policy jointly on all 20 types; evaluation holds out only new episodes, so Phase 2 measures in-distribution composite control rather than generalization to unseen semantics. In addition to the default end-effector-pose state representation, we repeat the key composite-task comparisons using joint-angle state under the identical training and evaluation protocol. The action representation remains end-effector deltas in both settings, isolating the effect of the state coordinate system. Every composite task is evaluated with 25 closed-loop rollouts under the same fixed seed schedule.

Experimental notes. The primary metric is closed-loop task success rate (SR). A rollout counts as successful only if the environment-defined success condition is satisfied within the horizon; for Phase 2, success requires completing all constituent subgoals in an episode. All models are initialized from the same pretrained $\pi_{0.5}$ checkpoint and fully fine-tuned on the corresponding task data under a shared optimization recipe, keeping the data pipeline, action representation, learning-rate schedule, and training budget matched across interfaces; SE and FM train with a slightly lower nominal sample exposure due to hardware allocation, so we do not use them for capacity-matched claims. Random seeds are fixed before training and evaluation, and the same seed schedule is reused for all models within a benchmark to control scene initialization, object instances, object placements, and other environment randomness. Slot-matched temporal comparisons also fix images, language, state-slot count, expert actions, and initial flow noise.

Multi-seed protocol. To ensure that the principal comparisons are not artifacts of a single optimization run, the five single-frame interfaces in Table 1 and the composite comparisons in Table 2 are trained from three independent random seeds. Their reported success rates are seed-averaged means, with a seed-level spread across runs. The slot-matched repeat-current control is reported separately over two training seeds. Paired task-bootstrap intervals separately resample aligned tasks for a fixed checkpoint, capturing evaluation and task-sampling uncertainty. The full atomic history-depth sweep in Figure 3 is an exploratory single-checkpoint analysis; its intervals do not include training-seed variance. The four pre-designated primary comparisons (AP and VP at $K{=}1$ and $K{=}8$) are corrected for multiplicity across the route-switch tests; the remaining interface-by-depth cells are exploratory and uncorrected.

F Supplementary Comparisons and Scope

F.1 Combined Interface–History Summary

Figure 4 reorganizes the main tables and the depth sweep into one visual comparison; it is not an additional experiment. Atomic $K = 8$ macros in this summary are constructed by adding the single-checkpoint $K = 1 \to 8$ sweep increment to the three-seed $K = 1$ macro. For example, the AP endpoint is $56.7 + 3.9 = 60.6$. They are therefore neither independently measured three-seed endpoints nor a source of training-seed uncertainty. The main routing comparison uses the directly reported composite means in Table 2.

F.2 Scope of the Claims

State composition. All interfaces receive the same 16-dimensional robot state, which combines end-effector, mobile-base, and gripper kinematics; the base pose is expressed in the world frame. Our claims concern how this complete state vector is best integrated. Attributing family-level gains to individual state dimensions is a component-level question outside the present estimand.

Temporal variation. The trained repeat-current control holds the number and positions of state slots fixed, so the $K = 8$ gain requires variation across recent states rather than additional conditioning capacity. We interpret it as use of recent kinematic evolution over a 0.35 s window and do not attribute it to a specific temporal statistic; identifying which statistic the policy exploits would refine, not change, this claim. Fixed-checkpoint probes measure reliance on the supplied history, not the causal mechanism behind an independently trained policy's success gain.

Interface designs and external validity. The five interfaces are evaluated as complete designs with the native widths and parameter counts they have in practical VLAs; SE and FM also train with slightly lower nominal exposure. The routing results therefore answer which deployable design to use, not how an artificial width-matched token placement would behave. All simulated comparisons use one $\pi_{0.5}$ scaffold and RoboCasa365. The real-robot check deploys NO-STATE and AP_8 on two tasks with ten trials per policy (Appendix G); it confirms the favored configuration with onboard state rather than re-deriving the full interface ordering.

G Real-Robot Experimental Details

Tasks. Both tasks ask the robot to collect six medicine items (bottles and boxes) from the table into a tray (Figure 6). In the single-arm task, the right arm transfers the items one at a time. In the dual-arm task, the two arms transfer them as three pairs. Each trial therefore chains six placement subgoals, the multi-stage regime in which short history helped in the simulated composite suite.

Demonstrations. For each task we record 150 demonstrations. Item positions are randomized across demonstrations, and the frames in Figure 6 are taken from this demonstration set.

Policies and training. For each task we train two policies, NO-STATE and AP_8, which gives four models in total. The AP_8 interface follows Section 4.3. All four models start from the public $\pi_{0.5}$ base checkpoint and share one recipe: AdamW, batch size 32, gradient-norm clipping at 1.0, and an exponential moving average of the weights with decay 0.999. The learning rate warms up linearly over 500 steps to 2.5×10^{-5} and then follows a cosine decay to 2.5×10^{-6} at step 30,000. We extend training from the original 30K steps to 60K. For the extra steps, the learning rate remains at its 2.5×10^{-6} floor. Each model is evaluated at its final checkpoint (step 59,999). Training shards the model across two GPUs with FSDP.

Evaluation protocol. Each policy is evaluated in 10 trials, with item placement re-randomized for every trial. A trial is successful only if all six items are placed in the tray in one uninterrupted attempt. The following count as failures: re-grasping after a missed grasp, any human correction or reset, and disturbing an item that is still on the table. An item that tips over after it has already been placed in the tray does not count as failure.

Results and uncertainty. On the single-arm task, NO-STATE succeeds in 8/10 trials (80%; 95% Wilson interval 49–94%) and AP_8 in 10/10 (100%; 72–100%). On the dual-arm task, NO-STATE succeeds in 4/10 (40%; 17–69%) and AP_8 in 7/10 (70%; 40–89%). Two-sided Fisher exact tests give $p = 0.47$ and $p = 0.37$, respectively. Each cell comes from one trained model, so these intervals reflect trial-level uncertainty only and exclude training-seed variance.

Relation to the simulated study. The hardware tasks are not RoboCasa365 tasks. Their success criterion, which permits no retries, is stricter than the simulator's horizon-limited success condition, so absolute success rates are not comparable across the two settings. The hardware pair changes state presence and history length together. It tests NO-STATE against AP_8, not $K = 1$ against $K = 8$, and

it includes neither a repeat-current control nor another interface. The result supports transferring the recommended configuration to onboard state. It does not reproduce the simulated interface ordering or routing crossover on hardware.