FV-PMP: ACCELERATING VISION-LANGUAGE-ACTION MODELS WITH FEEDBACK-VERIFIED PLAN-MEMORY PROPAGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models achieve strong performance in robot manipulation, but frequent policy refreshes impose substantial inference costs during closed-loop execution. Under receding-horizon control, an action-chunk VLA predicts multiple future actions, while only a limited portion of the predicted sequence is executed before the next policy refresh. Executing longer prefixes reduces inference frequency, but the predicted actions may become less compatible with the evolving state as the underlying observation becomes stale. This raises a question: can unexecuted predictions support subsequent control without being directly replayed? We propose **FV-PMP**, a Feedback-Verified Plan-Memory Propagation framework for efficient closed-loop VLA execution. Between visual policy refreshes, a lightweight propagator regenerates actions from current proprioceptive feedback and recent execution history. To support deeper propagation, FV-PMP converts the unexecuted action suffix into Remaining-Plan Memory, retaining compact motion and task-progress cues as conditioning information rather than executable commands. A Candidate-Conditioned Continuation Verifier estimates whether a propagated candidate remains suitable for continued propagation under the current execution context, while hard execution constraints further regulate propagation. Together, they determine whether execution continues through lightweight propagation or restores current visual conditioning through corrective or full replanning. Both replanning modes generate a fresh action chunk and renew the plan context. Experiments on SimplerEnv using the CogACT backbone show that FV-PMP reduces VLA invocation frequency to 32.38% of full-policy execution, achieving 3.27× average inference acceleration without sacrificing aggregate closed-loop task performance.

1 INTRODUCTION

Vision-Language-Action (VLA) models unify visual perception, language understanding, and action generation within a single policy, enabling increasingly generalizable robotic manipulation across diverse tasks and embodiments (Brohan et al., 2022; 2023; O'Neill et al., 2024; Bousmalis et al., 2023; Wu et al., 2024; Team et al., 2024; Kim et al., 2024; Intelligence et al., 2025; Black et al., 2024)

Recent VLAs further incorporate diffusion- or flow-based action generators to predict temporally extended action chunks, producing more coherent motion and richer temporal action representations (Zhao et al., 2023; Chi et al., 2025; Li et al., 2024a; Wen et al., 2024; Liu et al., 2025a; Kim et al., 2025). Despite these advances, closed-loop deployment remains computationally expensive. Modern VLAs typically combine a large vision-language backbone with an iterative action generation process, making frequent invocation of the full policy costly during control.

Existing work improves VLA inference efficiency from multiple directions, for example by pruning tokens, caching or reusing intermediate features, and introducing lightweight inference paths (Xu et al., 2026; Yang et al., 2026; Wang et al., 2025; Li et al., 2025). These techniques substantially reduce redundant computation. However, once the full visual policy is no longer invoked at every control step, a more direct closed-loop question arises: *how should subsequent actions be generated between two expensive visual policy refreshes, and how far can such propagation remain reliable?* If

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.

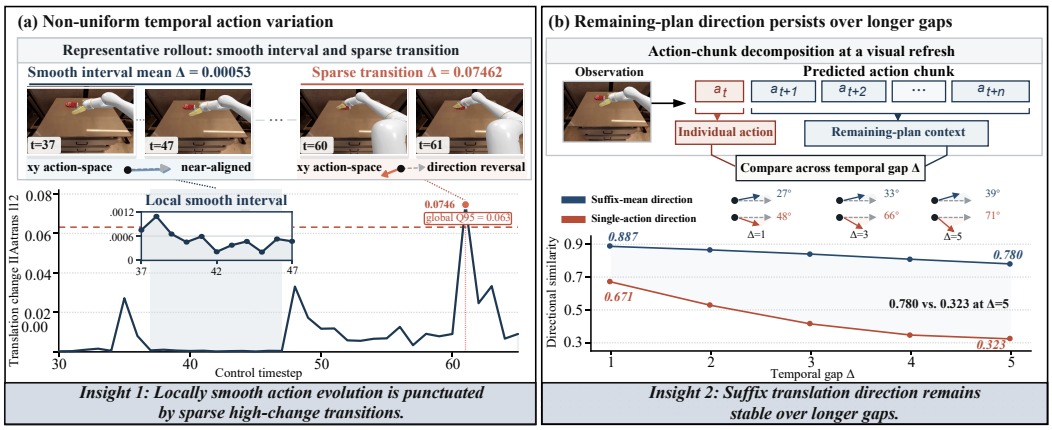


Figure 1: **Empirical motivation for FV-PMP.** (a) Action evolution is locally smooth but punctuated by sparse high-change transitions. (b) The normalized mean translation direction of the unexecuted action suffix remains more stable across temporal gaps than individual-action translation. The arrows visualize equivalent angular changes $\theta_\Delta = \arccos(s_\Delta)$ derived from the mean directional cosine similarity s_Δ; smaller angles indicate greater directional stability.

part of the subsequent control can be generated from available execution information, the frequency of expensive visual-policy calls can be further reduced. At the same time, propagating for too long without refreshed visual input may gradually make generated actions less compatible with the current execution state.

Our first observation is that closed-loop action evolution often exhibits strong local regularity, while the duration of such regularity is highly non-uniform. As shown in Figure 1(a), robotic manipulation trajectories contain many consecutive intervals with relatively small action changes, interspersed with only a few pronounced transitions involving larger motion or gripper adjustments. This temporal structure suggests that lightweight short-range action propagation can be supported over part of the control horizon using only recent execution history and current robot-state feedback, without invoking the expensive visual policy at every control step. However, because the duration of local regularity varies across states and tasks, a fixed propagation horizon may either terminate valid propagation too early or extend it into unreliable regions. We therefore combine lightweight action propagation with a learned candidate-conditioned continuation assessment, allowing the system to adaptively determine whether propagation should continue or visual conditioning should be restored.

While recent execution feedback provides a strong basis for short-range propagation, additional information about the intended continuation can further support deeper propagation as execution moves farther away from the most recent visual refresh. Our second observation is that different types of information contained in an action chunk do not share the same temporal horizon. As shown in Figure 1(b), the aggregate motion direction extracted from the remaining action suffix remains more stable over longer temporal gaps than individual-action translation. We complement this motion cue with relative plan-progression descriptors derived from the suffix. We summarize these residual cues as **Remaining-Plan Memory (RPM)** and use them as auxiliary conditioning for deeper propagation. RPM complements recent execution history and current robot-state feedback with compact information about motion tendency, upcoming gripper events, and relative task progression.

Motivated by these observations, we propose **FV-PMP**, a **F**eedback-**V**erified **P**lan-**M**emory **P**ropagation framework for efficient closed-loop VLA execution. After each visually conditioned policy refresh, FV-PMP executes the newly generated anchor action and constructs RPM from the remaining action suffix. Control then proceeds through staged lightweight propagation: short-range actions are generated from recent execution history and current robot-state feedback, while deeper propagation additionally incorporates RPM. A learned candidate-conditioned continuation score and hard execution constraints jointly determine whether propagation should continue. When propagation is no longer supported by the continuation criterion, FV-PMP restores current visual conditioning through corrective or full replanning. Both recovery modes generate a fresh action chunk, renew the plan context, and restart propagation. In this way, FV-PMP organizes local feedback-conditioned

propagation, plan-context-assisted deeper continuation, and selective visual recovery within a unified closed-loop execution framework.

Our main contributions are summarized as follows:

- We identify substantial local regularity in closed-loop VLA action evolution together with strongly non-uniform propagation horizons, motivating adaptive feedback-conditioned action propagation between expensive visual policy refreshes.
- We propose a staged lightweight propagation mechanism that generates short-range actions from recent execution history and current robot-state feedback, while augmenting deeper propagation with compact Remaining-Plan Memory extracted from the predicted action suffix.
- We introduce a feedback-verified execution strategy that combines a candidate-conditioned continuation verifier with hard execution constraints to adaptively alternate between lightweight propagation and visually conditioned corrective or full replanning.

2 Related Work

Vision-Language-Action Models. Vision-Language-Action (VLA) models extend pretrained vision-language representations to embodied control by jointly modeling visual perception, language understanding, and robot actions (Radford et al., 2021; Alayrac et al., 2022; Liu et al., 2023; Karamcheti et al., 2024; Driess et al., 2023; Li et al., 2024b). Early systems generate discrete action tokens autoregressively, while recent models increasingly adopt continuous action representations, parallel decoding, action chunking, and diffusion- or flow-based action generators (Brohan et al., 2022; 2023; Kim et al., 2024; Zhao et al., 2023; Chi et al., 2025; Kim et al., 2025; Liu et al., 2025a). Representative models include TinyVLA, Diffusion-VLA, CogACT, DexVLA, and π_0 (Wen et al.; 2024; Li et al., 2024a; Wen et al., 2025; Black et al., 2024). Action-chunking policies predict temporally extended action sequences that can be consumed according to the controller's execution strategy. FV-PMP builds on this predictive structure by summarizing the unexecuted suffix as compact context for subsequent feedback-conditioned action generation.

Acceleration for Vision-Language-Action Models. Existing work improves VLA inference efficiency through compact architectures, model compression, token pruning, feature reuse, and caching (Liu et al., 2024; Yue et al., 2024; Zhang et al., 2026; Xu et al., 2026; Yang et al., 2026; Wang et al., 2025; Liu et al., 2025b; Wei et al., 2026; Cheng et al., 2026; Ma et al., 2026). Parallel decoding, action reuse, and lightweight inference paths provide further reductions in computation (Kim et al., 2025; Song et al., 2025b;a; Tan et al., 2025; Li et al., 2025; Oi et al., 2026). FV-PMP addresses a complementary dimension of efficiency: closed-loop action generation between visually conditioned policy evaluations, combining feedback-conditioned lightweight propagation, plan-memory conditioning, and adaptive continuation verification.

3 Method

We consider a closed-loop execution setting in which a full VLA refresh produces an action chunk $A_r = (a_r^0, \ldots, a_r^{K-1})$ from visual observation o_r and language instruction ℓ. Under closed-loop execution, this predicted chunk contains temporal structure beyond the currently executed control step. FV-PMP exploits this structure as plan context for lightweight action generation conditioned on recent execution history and current feedback from proprioceptive and controller states. Generated action candidates are propagated between visual refreshes, while a learned candidate-conditioned continuation score and hard execution constraints govern whether propagation continues or current visual conditioning is restored through corrective or full replanning. As illustrated in Figure 2, FV-PMP couples feedback-conditioned action propagation, Remaining-Plan Memory, and adaptive visually conditioned recovery.

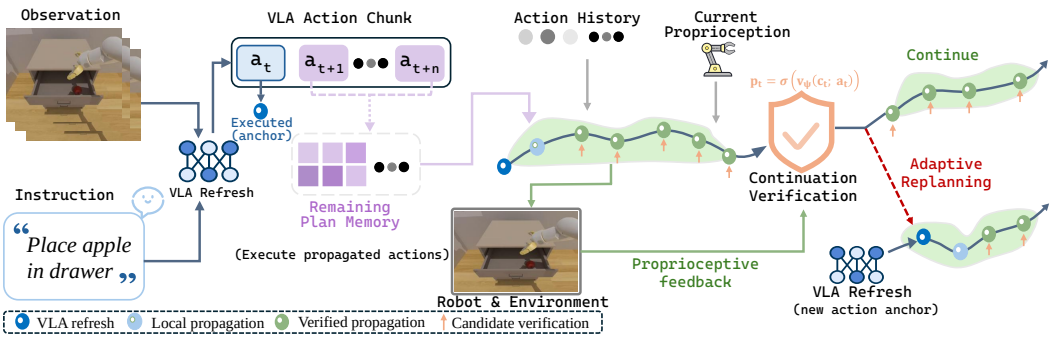


Figure 2: **Overview of FV-PMP.** A visual policy refresh produces a fresh action chunk and establishes an action anchor. FV-PMP subsequently propagates actions using recent execution history, proprioceptive feedback, and, at deeper propagation stages, Remaining-Plan Memory. Continuation verification determines whether propagation continues or corrective/full replanning is triggered.

3.1 Plan-Conditioned Action Propagation

Recent execution history and the remaining action prediction provide complementary information for subsequent control. Local execution context captures the robot's evolving state and immediate motion history, providing the primary basis for short-range propagation. At greater propagation depths, information retained in the unexecuted portion of the latest action chunk provides additional context about the intended continuation. FV-PMP therefore adopts a staged propagation strategy: short-range actions are generated from local execution context, while deeper propagation is additionally conditioned on compact remaining-plan information.

Local execution context. At control step t, the local context is $c_t = [a_{t-1}; a_{t-2}; s_t] \in \mathbb{R}^{22}$, where $a_{t-1}, a_{t-2} \in \mathbb{R}^7$ are the two most recently issued actions, each comprising three translational components, three rotational components, and a gripper command. The state vector $s_t \in \mathbb{R}^8$ contains end-effector pose and gripper-related feedback from proprioceptive and controller states. This context is updated at every control step, allowing the propagator to adapt to the evolving execution state without another visual policy refresh.

Remaining-Plan Memory. Local execution context captures recent control evolution but does not explicitly preserve information about the intended continuation represented in the latest VLA prediction. We therefore complement it with a compact summary of the remaining plan. Let r denote the most recent visual refresh and h the propagation depth since that refresh. The unexecuted suffix is $R_{r,h} = (a_r^h, \ldots, a_r^{K-1})$, which is summarized as

$$m_t = \phi(R_{r,h}) = \left[d_t^{\text{trans}}; d_t^{\text{rot}}; e_t; \delta_t; \rho_t\right] \in \mathbb{R}^9. \tag{1}$$

The deterministic mapping ϕ preserves two complementary aspects of the predicted continuation. The motion descriptors $d_t^{\text{trans}}, d_t^{\text{rot}} \in \mathbb{R}^3$ are the separately L_2-normalized mean translational and rotational components of the remaining suffix. The plan-progression descriptors encode the first predicted gripper transition within the suffix (e_t), its relative position (δ_t), and the remaining horizon fraction $\rho_t = (K-h)/K$. Together, these descriptors provide compact cues about motion tendency, upcoming gripper events, and relative plan progression. The exact encoding is detailed in the appendix.

Feedback-conditioned generation. FV-PMP uses a Local Propagator for the immediate continuation after a visual refresh and introduces remaining-plan context only at greater propagation depths:

$$\hat{a}_t = \begin{cases} f_{\text{local}}(c_t), & h \in \{1, 2\}, \\ f_{\text{plan}}\big([c_t; m_t]\big), & 3 \le h \le H_{\max}. \end{cases} \tag{2}$$

Both propagators are lightweight MLPs. This staged design uses current execution information as the primary signal for short-range propagation and introduces RPM as auxiliary conditioning when propagation extends farther from the latest visual refresh. The remaining suffix advances with the

propagation depth, and its summary m_t is recomputed accordingly, while the stored VLA chunk remains fixed until the next visual refresh.

The propagators are trained on actions produced by the frozen Full policy. We optimize f_{local} using mean-squared error and f_{plan} using Smooth-L_1 loss, both in standardized action space. Architectural details and training configurations are provided in the appendix.

3.2 Candidate-Conditioned Continuation Verification

Lightweight propagation reduces the need for frequent visual policy refreshes, but its compatibility with the current execution state can degrade as execution moves farther from the most recent visual observation. Moreover, the effective propagation horizon varies across states and tasks, making a fixed stopping rule undesirable. FV-PMP therefore employs a lightweight *Candidate-Conditioned Continuation Verifier* to estimate whether propagation should continue with the proposed action under the current execution context. At inference time, the verifier operates only on local execution information and the proposed action, without invoking the VLA.

Candidate-conditioned continuation score. Given the local execution context c_t and a propagated candidate $\hat{a}_t$, the verifier predicts $p_t = \sigma(v_\psi([c_t; \hat{a}_t]))$, where v_ψ is a lightweight MLP with a 29-dimensional input and σ denotes the sigmoid function. The score $p_t \in [0, 1]$ represents the estimated suitability of continuing propagation with the proposed candidate under the current execution context. Conditioning explicitly on $\hat{a}_t$ allows the verifier to evaluate the candidate under consideration rather than relying only on elapsed propagation depth. The same verifier is applied to both local and plan-conditioned candidates. Remaining-Plan Memory is not provided to the verifier directly; instead, its effect is reflected through the candidate generated by the plan-conditioned propagator.

Teacher-agreement persistence supervision. We construct an offline supervision target from agreement with the frozen Full policy along data-collection trajectories. Let i index recorded verifier decisions, and let $\tilde{a}_i$ and a_i^{Full} denote a propagated candidate and the corresponding Full-policy action, respectively. Single-record agreement is defined as

$$q_i = \mathbf{1}\left[\left\|\tilde{a}_i^{1:6} - (a_i^{\text{Full}})^{1:6}\right\|_2 \le \epsilon\right] \mathbf{1}\left[g(\tilde{a}_i) = g(a_i^{\text{Full}})\right], \tag{3}$$

where the first six dimensions are the unnormalized CogACT translational and rotational action components before environment-level conversion, $g(a) = \mathbf{1}[a^7 \ge 0.5]$ denotes the binary gripper state, and $\epsilon = 0.1$ is fixed throughout all experiments.

Single-record agreement alone does not indicate whether teacher agreement persists across nearby recorded decisions. We therefore define the record-sequence persistence target

$$y_i = \prod_{j=0}^{\min(3, N-1-i)} q_{i+j}, \tag{4}$$

where N is the number of recorded verifier decisions in the trajectory. The target is positive only when agreement persists at the current record and the following up to three available records. Here, i follows verifier records in data-collection order rather than consecutive low-level control timesteps. This target provides offline supervision for learning the persistence of teacher agreement around the current recorded decision.

We train the verifier using task-balanced binary cross-entropy. A shared acceptance threshold τ is selected on a trajectory-grouped local-candidate validation split to maximize the number of accepted candidates while maintaining at least 90% accepted-sample precision for every task. The resulting threshold is $\tau = 0.69114$. Transfer to plan-conditioned candidates is evaluated separately in Appendix B.4. During deployment, the continuation score p_t is combined with hard execution constraints to determine whether propagation continues or visually conditioned replanning is triggered, as described next.

3.3 Adaptive Replanning

A rejected propagation candidate indicates that the candidate should no longer continue propagation under the current execution context. The controller must therefore restore current visual conditioning,

but doing so does not necessarily require the full action-generation computation. FV-PMP provides two recovery modes: corrective replanning performs a lower-cost visually conditioned replan, whereas full replanning uses the complete action-generation process.

Corrective and full replanning. Both replanning modes invoke the VLA using the current visual observation and task instruction, generate a fresh action chunk, and renew Remaining-Plan Memory (RPM). Full replanning follows the original DDIM-10 action-generation process, with all DiT blocks recomputed at every denoising iteration. Corrective replanning instead uses interval-5 intermediate-feature caching, following EfficientVLA (Yang et al., 2026). Specifically, attention and MLP residual features are recomputed at denoising indices 0 and 5 and reused at the intermediate iterations. The cache is reset for every VLA invocation and is not shared across successive replanning events. The two modes therefore use the same current visual observation and task instruction and produce the same form of fresh action chunk, but differ in the computation allocated to action generation.

Continuation-guided recovery selection. For a candidate subject to verification, let $b_t = \mathbf{1}[g(\hat{a}_t) \neq g(a_{t-1})]$ indicate whether the candidate proposes a gripper transition. The execution response is selected by the deterministic rule

$$u_t = \begin{cases} \text{Propagate}, & b_t = 0,\ p_t \geq \tau, \\ \text{Corrective}, & b_t = 0,\ \tau_{\text{low}} \leq p_t < \tau, \\ \text{Full}, & \text{otherwise}, \end{cases} \tag{5}$$

where $\tau_{\text{low}} = 0.5$. Candidates satisfying the acceptance criterion continue propagation, whereas rejected candidates with intermediate continuation scores trigger corrective replanning. Lower-scoring candidates invoke full replanning. For verified candidates, a proposed gripper transition directly triggers full replanning, providing a conservative safeguard for discrete controller-state changes. The learned continuation score and hard execution constraint jointly determine the execution response without introducing a separate learned router.

Staged closed-loop execution. Following each visually conditioned policy refresh, FV-PMP performs staged propagation, with verification and remaining-plan context introduced at different propagation depths. At $h = 1$, the Local Propagator generates the first propagated action from recent execution history and current robot-state feedback, and the candidate is executed directly. This design choice is evaluated in Appendix D.2. At $h = 2$, the same Local Propagator is retained, while the Candidate-Conditioned Continuation Verifier and hard execution constraint are activated. From $h \geq 3$ onward, Remaining-Plan Memory is additionally incorporated through the Plan-Conditioned Propagator, with verification applied to every candidate.

An accepted candidate is executed and updates recent execution history without invoking the VLA. A corrective or full replanning event instead uses the current visual observation, generates a fresh action chunk, establishes a new action anchor, renews Remaining-Plan Memory, and restarts propagation depth. We impose a maximum propagation depth $H_{\max} = 15$; after the action at $h = 15$, the next control decision is forced to perform full replanning. Execution is initialized with three Full-policy bootstrap steps, with further implementation details provided in the appendix.

4 Experiments

4.1 Experimental Settings

Benchmark. We evaluate FV-PMP on SIMPLER (Li et al., 2024c) using the Google Robot setting, including Visual Matching (VM) and Variant Aggregations (VA); the latter introduces variations in lighting, backgrounds, textures, and object appearance. Each setting contains four suites: *Pick Coke Can*, *Move Near*, *Open/Close Drawer*, and *Open Top Drawer and Place Apple*.

Baselines. We use CogACT (Li et al., 2024a) as the VLA backbone. We compare FV-PMP with the reported results of FastV, VLA-Cache, EfficientVLA, and SP-VLA on SIMPLER (Yang et al., 2026; Li et al., 2025). We additionally evaluate Direct Chunk Replay and Fixed Propagation under our protocol.

Table 1: Performance and efficiency comparison with representative VLA acceleration methods on SIMPLER. FV-PMP results are averaged over three seeds; all other entries are reported results.

SIMPLER	Method	Success Rate (%, ↑) / Speedup (↑)				Average
		PickCan	MoveNear	Drawer	DrawerApple	
Visual Matching	CogACT	91.30 / 1.00	**85.00** / 1.00	71.80 / 1.00	50.90 / 1.00	74.80 / 1.00
	FastV	92.60 / 1.21	81.40 / 1.21	69.80 / 1.21	52.40 / 1.21	74.10 / 1.21
	VLA-Cache	92.00 / 1.38	83.30 / 1.38	70.50 / 1.38	51.60 / 1.38	74.40 / 1.38
	EfficientVLA	93.30 / 1.93	81.30 / 1.93	68.20 / 1.93	**53.80** / 1.93	74.20 / 1.93
	SP-VLA	90.00 / 2.62	82.08 / 2.52	75.35 / 1.80	52.78 / 1.67	75.05 / 2.15
	FV-PMP	**93.67** / **4.12**	82.64 / **4.18**	**76.08** / **2.45**	52.47 / **3.23**	**76.21** / **3.49**
Variant Aggregations	CogACT	89.60 / 1.00	**80.80** / 1.00	28.30 / 1.00	46.60 / 1.00	61.30 / 1.00
	FastV	91.40 / 1.19	78.60 / 1.19	27.60 / 1.19	**50.60** / 1.19	62.10 / 1.19
	VLA-Cache	91.70 / 1.37	79.30 / 1.37	32.50 / 1.37	45.80 / 1.37	62.30 / 1.37
	EfficientVLA	**93.20** / 1.91	75.80 / 1.91	26.90 / 1.91	49.20 / 1.91	61.20 / 1.91
	SP-VLA	86.18 / 2.48	77.33 / 2.63	**55.29** / 1.81	41.80 / 1.44	**65.16** / 2.09
	FV-PMP	92.12 / **4.00**	76.33 / **3.39**	40.12 / **2.14**	41.80 / **2.66**	62.59 / **3.05**

Implementation and evaluation. The pretrained CogACT backbone remains frozen throughout all experiments. CogACT predicts action chunks of length $K = 16$, with each action represented by seven dimensions; FV-PMP uses $H_{\max} = 15$. Thresholds and execution rules are shared across tasks. We report task success, VLA Rate, policy frequency, latency, and speedup relative to full CogACT. Task success is macro-averaged equally across the four suites. VLA Rate is computed as the fraction of executed control steps that invoke the visually conditioned VLA, where both corrective and full replanning count as one VLA invocation. Within each suite, policy frequency is the reciprocal of the unrounded mean per-control-step latency, and speedup is the Full-CogACT-to-FV-PMP latency ratio. We macro-average the four suite-wise speedups. The main benchmark and latency results are averaged over three evaluation seeds, with latency measured on a single NVIDIA RTX 4090. Controlled analyses and ablations use the same paired VA evaluation split on a single NVIDIA RTX 5090.

4.2 Main Results

Main results. As shown in Table 1, FV-PMP achieves the largest policy-inference speedup among the compared methods, reaching $3.49\times$ on VM and $3.05\times$ on VA. It achieves competitive macro success of 76.21% on VM and 62.59% on VA, while reducing the VLA invocation rate to 30.97% and 33.79%, respectively. These results demonstrate that FV-PMP avoids roughly two-thirds of expensive VLA calls while preserving aggregate closed-loop task performance.

4.3 Analysis

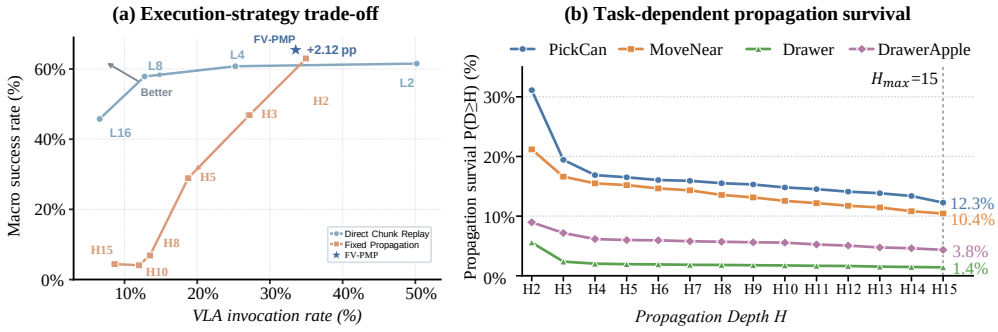


Figure 3: **Execution-strategy trade-off and propagation survival.** (a) Macro success versus VLA Rate for Direct Chunk Replay, Fixed Propagation, and FV-PMP. (b) Propagation survival $P(D \geq H)$ across task suites.

Alternative execution strategies. Figure 3(a) shows that both longer chunk replay and fixed propagation reduce VLA usage at the cost of task success, indicating that neither stale action reuse nor a uniform propagation horizon is sufficient for reliable closed-loop execution. FV-PMP instead adapts propagation to the current execution context, achieving 64.86% macro success at a 33.75% VLA Rate and outperforming the closest fixed setting ($H = 2$) by 2.12 percentage points. Figure 3(b) further shows substantially deeper propagation in PickCan and MoveNear than in drawer-related tasks, supporting adaptive rather than fixed propagation.

Table 2: **Component ablation of FV-PMP.**

Variant	Success ↑	VLA Rate ↓	Steps/Refresh	Exec. Depth
w/o Remaining-Plan Memory	64.20%	37.66%	2.655	2.859
w/o Verifier	17.50%	**9.81%**	10.198	7.526
w/o Staged Propagation	55.41%	56.70%	1.764	7.185
w/o Corrective Replanning	62.16%	33.23%	3.009	4.593
Full FV-PMP	**64.86%**	33.75%	2.963	**4.515**

Component ablations. Table 2 shows that the main components play distinct roles. Removing Remaining-Plan Memory primarily shortens propagation and increases VLA Rate, with only a small change in success. Removing the Verifier causes success to collapse despite substantially fewer VLA calls. Removing staged propagation also hurts both success and refresh efficiency. Corrective replanning changes VLA Rate only slightly but improves success by 2.70 percentage points, confirming its role as a selective recovery mechanism rather than a primary source of acceleration.

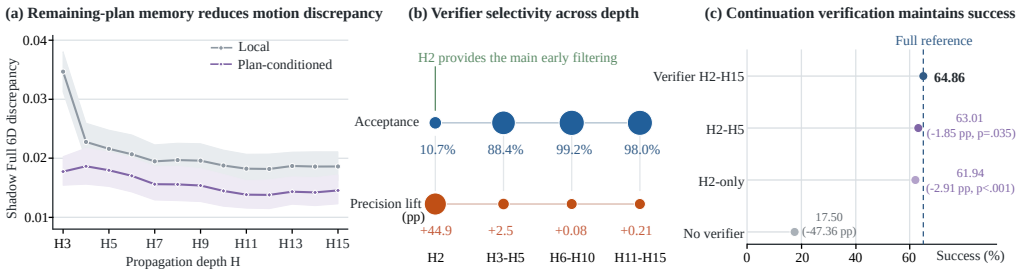


Figure 4: **Depth-wise propagation and verification analysis.** (a) Plan conditioning reduces deep-propagation motion discrepancy. (b) Verifier selectivity is strongest at early depths. (c) Truncating verification progressively reduces macro success.

Depth-wise propagation and verification. Figure 4(a) shows that Plan-Conditioned Propagation consistently reduces the six-dimensional CogACT-action discrepancy relative to same-state Shadow Full VLA actions across $h = 3$–15, providing action-level evidence that Remaining-Plan Memory supplies useful context for deeper propagation. Figures 4(b)–(c) further explain the role of multi-stage continuation verification.In the held-out candidate-source transfer audit, the verifier accepts 10.68% of the recorded $H2$ local candidates and rejects 95.8% of the candidates labeled inconsistent. This audit statistic is distinct from the deployment VLA rate, whose denominator is the total number of executed control steps. The deeper audit cohorts are progressively filtered survivor records, so their higher acceptance rates should not be interpreted as unconditional runtime acceptance rates.Nevertheless, truncating verification reduces macro success from 64.86% to 63.01% for $H2$–$H5$ and to 61.94% for $H2$ only. Thus, early verification performs the strongest filtering, while continued verification remains beneficial for reliable deep propagation.

Remaining-plan context. Table 3 shows that progress context provides the stronger individual contribution, while combining motion and progress information yields the lowest VLA Rate and largest executed depth. Compared with History Only, full RPM reduces VLA Rate from 37.66% to 33.75% and increases mean executed depth from 2.859 to 4.515 with comparable success, indicating that its primary role is to provide additional context for deeper propagation.

Runtime behavior. Figure 5 shows that propagation accounts for roughly two-thirds of executed control steps, with plan-conditioned stages extending execution beyond the initial local steps, partic-

Table 3: **Analysis of Remaining-Plan Memory.** H, M, and P denote execution history, remaining-motion context, and remaining-progress context, respectively.

Variant	Propagation Context			Success ↑	VLA Rate ↓	Steps/Refresh ↑	Exec. Depth ↑
	H	M	P				
History Only	✓	–	–	64.20%	37.66%	2.655	2.859
+ Motion	✓	✓	–	62.61%	38.72%	2.583	3.089
+ Progress	✓	–	✓	64.68%	35.10%	2.849	4.160
Full RPM	✓	✓	✓	**64.86%**	**33.75%**	**2.963**	**4.515**

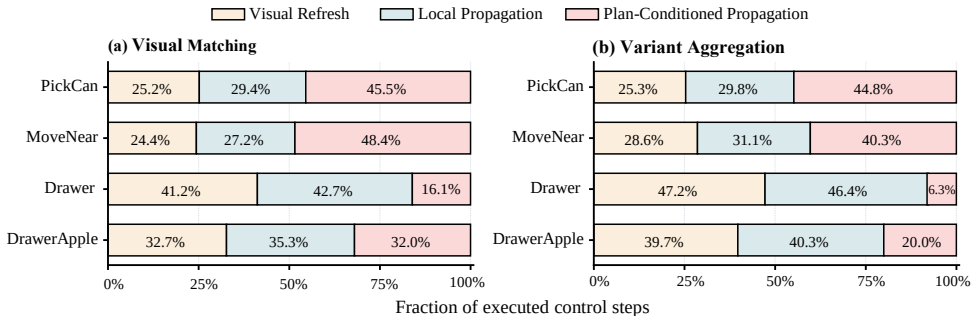


Figure 5: **Runtime decision composition across task suites.** Visual Refresh includes corrective and full replanning; Local Propagation covers $h = 1$–2, and Plan-Conditioned Propagation covers $h = 3$–15.

ularly in PickCan and MoveNear. In contrast, the drawer-related tasks trigger visual refresh more frequently. This task-dependent composition agrees with the survival patterns in Figure 3(b), further showing that FV-PMP adapts its effective propagation horizon to the execution context.

Table 4: **Policy inference frequency and latency on SIMPLER.** Full CogACT and FV-PMP are measured on a single NVIDIA RTX 4090 under the same three-seed evaluation protocol. The Average column separately reports the arithmetic mean of suite-level frequency and latency. The speedups in Table 1 are macro-averages of the four suite-wise speedup ratios.

SIMPLER	Method	Frequency (Hz,↑) / Latency (s,↓)				
		PickCan	MoveNear	Drawer	DrawerApple	Average
VM	Full CogACT	6.88/0.145	7.05/0.142	7.04/0.142	7.00/0.143	6.99/0.143
	FV-PMP	28.32 / 0.035	29.42 / 0.034	17.23 / 0.058	22.58 / 0.044	**24.39 / 0.043**
VA	Full CogACT	6.82/0.147	7.01/0.143	6.45/0.155	6.19/0.162	6.62/0.151
	FV-PMP	27.32 / 0.037	23.79 / 0.042	13.78 / 0.073	16.47 / 0.061	**20.34 / 0.053**

Frequency and latency. FV-PMP increases the macro-averaged policy frequency from 6.99 to 24.39 Hz on VM and from 6.62 to 20.34 Hz on VA. The corresponding macro-averages of the four suite-wise speedups are 3.49× and 3.05×, respectively. These gains confirm that reduced VLA invocation translates into substantial wall-clock acceleration.

5 Conclusion

We propose FV-PMP, a closed-loop framework for efficient VLA inference through feedback-conditioned action propagation. FV-PMP combines lightweight local propagation, Remaining-Plan Memory for deeper continuation, and candidate-conditioned continuation verification with selective replanning. On SIMPLER using the CogACT backbone, FV-PMP reduces VLA invocation rates to 30.97% on VM and 33.79% on VA, achieving 3.49× and 3.05× average inference speedups while maintaining competitive task success. These results demonstrate that the unexecuted action suffix can be summarized as reusable plan context, rather than directly replayed, for efficient closed-loop VLA execution.

AI USE STATEMENT

Generative AI tools were used during manuscript preparation to improve textual clarity and organization, refine scientific figures, and provide critical feedback on the presentation of the methodology and experimental analysis. The authors manually examined and revised all resulting suggestions against the original experiments, data, and cited sources. All scientific decisions and conclusions remain those of the authors, who assume full responsibility for the submitted work.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

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.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauzá, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving generalist agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Jintao Cheng, Haozhe Wang, Weibin Li, Gang Wang, Yipu Zhang, Xiaoyu Tang, Jin Wu, Xieyuanli Chen, Yunhui Liu, and Wei Zhang. Vla-iap: Training-free visual token pruning via interaction alignment for vision-language-action models. *arXiv preprint arXiv:2603.22991*, 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.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

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.

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. *arXiv preprint arXiv:2402.07865*, 2024.

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*, 2025.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024a.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. In *International Conference on Learning Representations*, volume 2024, pp. 26703–26721, 2024b.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024c.

Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shu-Tao Xia, Zhi Wang, and Wenwu Zhu. Sp-vla: A joint model scheduling and token pruning approach for vla model acceleration. In *The Fourteenth International Conference on Learning Representations*, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Pengju An, Xiaoqi Li, Kaichen Zhou, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation. *Advances in Neural Information Processing Systems*, 37:40085–40110, 2024.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025a.

Ziyan Liu, Yeqiu Chen, Hongyi Cai, Tao Lin, Shuo Yang, Zheng Liu, and Bo Zhao. Bridging the semantic-action gap in visual token pruning for efficient vla inference. *arXiv preprint arXiv:2511.16449*, 2025b.

Shilin Ma, Chubin Zhang, Changyuan Wang, Yuji Wang, Yue Wu, Zixuan Wang, Jingqi Tian, Zheng Zhu, and Yansong Tang. Safe-pruner: Semantic attention–guided future-aware token pruning for efficient vision-language-action manipulation. In *European Conference on Computer Vision*, pp. 163–180. Springer, 2026.

Ryuji Oi, Hikari Otsuka, Kosuke Matsushima, Yuki Ichikawa, Masato Motomura, Tatsuya Kaneko, and Daichi Fujiki. Training-free acceleration for vision-language-action models with action caching and refinement. *arXiv preprint arXiv:2607.06370*, 2026.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Yuxin Huang, Han Zhao, Donglin Wang, and Haoang Li. Ceed-vla: Consistency vision-language-action model with early-exit decoding. *arXiv preprint arXiv:2506.13725*, 2025a.

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Han Zhao, Wei Zhao, Zhide Zhong, Zongyuan Ge, Zhijun Li, Donglin Wang, Lujia Wang, et al. Pd-vla: Accelerating vision-language-action model integrated with action chunking via parallel decoding. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13162–13169. IEEE, 2025b.

Xudong Tan, Yaoxin Yang, Peng Ye, Jialin Zheng, Bizhe Bai, Xinyi Wang, Jia Hao, and Tao Chen. Think twice, act once: Token-aware compression and action reuse for efficient inference in vision-language-action models. *arXiv preprint arXiv:2505.21200*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning. *arXiv preprint arXiv:2509.05614*, 2025.

Yujie Wei, Jiahan Fan, Jiyu Guo, Ruichen Zhen, Rui Shao, Xiu Su, Zeke Xie, and Shuo Yang. Learning to accelerate vision-language-action models through adaptive visual token caching. *arXiv preprint arXiv:2602.00686*, 2026.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation, 2024. *URL https://arxiv. org/abs/2409.12514*, 2.

Junjie Wen, Minjie Zhu, Yichen Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Chengmeng Li, Xiaoyu Liu, Yaxin Peng, Chaomin Shen, et al. Diffusion-vla: Generalizable and interpretable robot foundation model via self-generated reasoning. *arXiv preprint arXiv:2412.03293*, 2024.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, volume 2024, pp. 10641–10662, 2024.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. *Advances in Neural Information Processing Systems*, 38:164448–164473, 2026.

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. *Advances in Neural Information Processing Systems*, 38:40891–40914, 2026.

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. *Advances in Neural Information Processing Systems*, 37:56619–56643, 2024.

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18764–18772, 2026.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

A Implementation Details

A.1 Remaining-Plan Memory encoding

Let $A_r = (a_r^0, \ldots, a_r^{K-1})$ be the action chunk produced by the most recent visual refresh, and let h denote the number of propagated actions elapsed since that refresh. FV-PMP summarizes the unexecuted suffix $R_{r,h} = (a_r^h, \ldots, a_r^{K-1})$ using a deterministic nine-dimensional mapping. For the motion components, we first compute the suffix mean

$$\bar{d}_h = \frac{1}{K-h} \sum_{j=h}^{K-1} a_r^{j,1:6}, \tag{6}$$

and separately normalize its translational and rotational parts,

$$d_h^{\mathrm{trans}} = \frac{\bar{d}_h^{1:3}}{\max(\|\bar{d}_h^{1:3}\|_2, 10^{-6})}, \tag{7}$$

$$d_h^{\mathrm{rot}} = \frac{\bar{d}_h^{4:6}}{\max(\|\bar{d}_h^{4:6}\|_2, 10^{-6})}. \tag{8}$$

The remaining three entries encode plan progression. After binarizing the gripper component at 0.5, $e_h \in \{-1, 0, 1\}$ records the direction of the first gripper-state transition within the suffix, $\delta_h \in [0, 1]$ records its relative position, and $\rho_h = (K-h)/K$ is the remaining-horizon fraction. If no transition exists, $(e_h, \delta_h) = (0, 1)$. The complete memory is

$$m_t = [d_h^{\mathrm{trans}}; d_h^{\mathrm{rot}}; e_h; \delta_h; \rho_h] \in \mathbb{R}^9. \tag{9}$$

The stored chunk is not modified by propagation. Instead, the suffix index advances with h and the summary is recomputed before each plan-conditioned candidate. Thus, the memory supplies plan-relative context but never an action that is replayed directly.

A.2 Lightweight networks

Table 5 specifies all trainable components added by FV-PMP. The VLA backbone remains frozen. The local propagator consumes the 22-dimensional execution context $c_t = [a_{t-1}; a_{t-2}; s_t]$. The plan-conditioned propagator concatenates the same context with m_t, yielding a 31-dimensional input. The verifier receives $[c_t; \hat{a}_t] \in \mathbb{R}^{29}$. Inputs and action targets are standardized using statistics computed only on the corresponding training split and are de-standardized before control.

Table 5: Architectures of the lightweight FV-PMP components. Each arrow denotes a fully connected layer.

Component	Input	Architecture	Output
Local propagator	22	2×128 ReLU MLP	7-D action
Plan propagator	31	2×64 ReLU MLP	7-D action
Verifier	29	2×128 GELU MLP	probability

The verifier applies dropout with probability 0.1 after its first GELU activation during training. At inference time all three networks operate deterministically and require no image features or VLA forward pass.

A.3 Closed-loop execution rules

The first three policy decisions in each episode use Full CogACT to initialize the two-action history and establish a visually conditioned chunk. After a refresh, $H1$ is generated by the local propagator and executed directly. At $H2$, the local propagator remains active and its candidate is verified. From $H3$ through $H15$, the plan-conditioned propagator is used and every candidate is verified. A gripper-state transition proposed by a verified candidate forces Full replanning. After executing $H15$, the next decision is also forced to use Full replanning, bounding the interval without a new visual condition.

The shared verifier acceptance threshold is $\tau = 0.6911365$, and the lower recovery threshold is $\tau_{\text{low}} = 0.5$ at every verified depth from $H2$ through $H15$. A non-transition candidate with $p_t \geq \tau$ is executed, a non-transition candidate with $\tau_{\text{low}} \leq p_t < \tau$ triggers corrective refresh, and a lower score or gripper transition triggers Full refresh. The final deployed policy explicitly overwrites the higher recovery-floor default inherited from its parent implementation; consequently, it uses 0.5, not $(1 + \tau)/2$, for deep propagation as well. Corrective refresh uses the same current image and instruction as Full refresh but enables interval-5 feature caching during the ten denoising steps: residual features are recomputed at indices 0 and 5 and reused at intermediate indices. The cache is cleared at the beginning of every VLA query and is never shared across control decisions.

B Training and Data Construction

B.1 Propagation targets and splits

Both propagators regress the action issued by the frozen Full CogACT policy at the corresponding recorded state. The local model is trained from the 22-dimensional execution context. The plan-conditioned model uses the same context together with the nine-dimensional summary of the chunk suffix available at that state. We forbid task identity, direct chunk-action references, and candidate-minus-chunk differences from the plan propagator's inputs. Consequently, Remaining-Plan Memory can condition regeneration but cannot serve as an action-replay shortcut.

All train/validation partitions are grouped by physical rollout (or job identifier for plan-memory collection), preventing states from the same trajectory from appearing in both partitions. Twenty percent of groups are held out for validation. The plan-conditioned model additionally uses task-balanced sample weights and was checked with leave-one-task-out diagnostics before the frozen checkpoint was selected.

B.2 Optimization

Table 6 reports the optimization settings used by the released training scripts. Training is full-batch because these MLPs and their offline datasets fit in memory. The checkpoint with the lowest grouped validation loss is retained.

Table 6: Training configuration for FV-PMP's lightweight components.

Component	Objective	LR	WD	Epochs
Local	standardized MSE	10^{-3}	10^{-2}	200
Plan-conditioned	balanced Smooth-L_1	10^{-3}	10^{-4}	900
Verifier	balanced BCE	10^{-3}	10^{-3}	500

All components use AdamW. Models are selected by validation MSE (local), validation motion ℓ_2 (plan-conditioned), or validation BCE (verifier). The plan propagator uses early stopping with patience 100. Training seeds are 7 for the local propagator, 20260811 for plan-memory data partitioning, and 43 for the verifier. The final plan checkpoint is fit on all available groups after the grouped and leave-one-task-out feasibility checks.

B.3 Continuation supervision and calibration

The verifier training set contains 41,196 recorded $H2$ local-candidate decisions from 708 trajectory groups, with Full CogACT supplying teacher actions. Agreement requires an unnormalized six-dimensional CogACT-action discrepancy of at most 0.1 and an identical binary gripper command. This fixed mixed-unit metric does not separately weight translation and rotation. The persistence target covers the current and following up to three records in trajectory order; records may cross policy refreshes and need not be adjacent control steps. It is an offline proxy, not a counterfactual execution guarantee.

Task-balanced BCE uses 567 training groups (32,983 records) and 141 validation groups (8,213 records). The grouped validation split is used for checkpoint selection and threshold calibration; generalization to deployed candidate cohorts is evaluated on a separate 200-episode collection in

Appendix B.4. Maximizing coverage subject to at least 90% accepted precision per task gives $\tau = 0.6911365$. This constraint calibrates local candidates, while transfer to deployed plan-conditioned candidates is evaluated below.

B.4 CANDIDATE-SOURCE TRANSFER AT DEPLOYMENT

The verifier observes a 29-dimensional candidate–context pair, not generator identity or depth. We evaluate its transfer from training-time local candidates to deployed plan-conditioned candidates using a separate 200-episode VA collection, unused for frozen-verifier training or calibration, with 7,829 $H2$ local and 6,819 $H3$–$H15$ plan-conditioned candidates.

Table 7: Verifier behavior on deployed candidate cohorts. Precision lift is accepted precision minus the positive rate within each cohort. The pooled row aggregates different depth cohorts and is not used to infer within-cohort discrimination.

Cohort	Candidate source	N	Accept.	Precision	Lift	AUROC	AUPRC
$H2$	Local	7,829	10.68%	70.93%	+44.86	0.782	0.600
$H3$–$H5$	Plan-conditioned	2,018	88.40%	78.20%	+2.48	0.592	0.803
$H6$–$H10$	Plan-conditioned	2,588	99.19%	81.53%	+0.08	0.552	0.846
$H11$–$H15$	Plan-conditioned	2,213	98.01%	80.82%	+0.21	0.592	0.869
$H2$–$H15$	All	14,648	50.22%	79.31%	+28.37	0.841	0.812

At $H2$, the verifier accepts 10.68% of candidates and rejects 5,545 of 5,788 inconsistent candidates (95.80%). Because deeper cohorts consist of progressively selected survivors, their high acceptance and small precision lift should be interpreted together with the closed-loop schedule ablation below. Although deep rejection is rare, it remains consequential, particularly for DrawerApple.

Source-matched deep-stage alternative. Requiring a common visual anchor and consecutive timesteps and depths yields 4,184 strict windows over $H3$–$H12$, split by trajectory group into 2,867/698/619 train/validation/test records. On the strict-label test split, AUROC improves from 0.690 to 0.810. At the validation operating point matched to deployed-verifier coverage ($\tau = 0.7996$), accepted precision improves from 89.64% to 94.14%, FPR falls from 19.26% to 11.85%, and acceptance changes from 40.55% to 44.10%. Adding generator identity and depth does not improve AUROC (0.794).

The closed-loop diagnostic is hybrid: the strict gate replaces only $H3$–$H12$, while $H2$ and $H13$–$H15$ retain the frozen verifier. In an episode-matched 80-episode pilot, Success is 71.25% for the deployed configuration and 67.50% for the hybrid variant. Thus, at this pilot scale, the offline gain yields no observed closed-loop gain. We retain the deployed verifier; the 1,992-episode schedule ablation below further shows that verification through $H15$ outperforms stopping at $H5$ or $H2$.

C EVALUATION PROTOCOL AND METRICS

Controlled ablations use the complete 1,992-episode SIMPLER Variant Aggregation evaluation with episode seed base 44000. Methods compared within an ablation use identical episode identities and initializations, enabling paired outcome analysis. Macro success is the unweighted mean over the four task-suite success rates. We refer to this macro-averaged metric as Success throughout the following ablations. VLA Rate is the fraction of executed control decisions that invoke either corrective or Full CogACT. Steps/Refresh is the reciprocal cycle-level view of that frequency, while reported propagation depth is action-weighted over actually executed propagated actions and excludes VLA actions.

Main FV-PMP results use three evaluation seeds in both Visual Matching and Variant Aggregations. Diagnostic ablations use the paired seed-44000 protocol. Wall-clock latency is reported only for methods measured under the same device and timing protocol.

D Additional Experimental Results

D.1 How long must verification continue?

Table 8 isolates the depth range over which the verifier is active. H2-only retains the first post-refresh verification but bypasses probability-based rejection at $H3$–$H15$. H2–H5 retains early and intermediate verification but disables it at $H6$–$H15$. Plan-conditioned propagation, the gripper hard guard, the $H15$ refresh ceiling, and adaptive recovery remain unchanged in both variants.

Table 8: Verifier-depth ablation on the paired VA evaluation. Full step-wise verification provides the best success–invocation trade-off.

Schedule	Success ↑	VLA Rate ↓	Steps/Refresh ↑	Exec. Depth ↑
H2 only	61.94%	**32.04%**	**3.121**	**4.923**
H2–H5	63.01%	33.43%	2.991	4.629
H2–H15 (full)	**64.86%**	33.75%	2.963	4.515

Restricting verification to H2–H5 loses 1.85 Success points while reducing VLA Rate by only 0.31 points. Under paired evaluation, Full succeeds and H2–H5 fails on 23 episodes, whereas the reverse occurs on 10 episodes (McNemar exact $p = 0.035$). The paired Success difference is -1.846 points with a 95% confidence interval of $[-3.144, -0.584]$. As shown in Table 9, most of the residual benefit arises on DrawerApple, the longest and most interaction-sensitive task.

Table 9: Task-wise success (%) for the verifier-depth ablation.

Suite	H2 only	H2–H5	H2–H15
PickCan	92.61%	**92.97%**	92.85%
MoveNear	76.33%	**77.33%**	77.17%
Drawer	39.68%	40.48%	**40.74%**
DrawerApple	39.15%	41.27%	**48.68%**

D.2 Verification from the First Propagation Step

FV-PMP executes the first propagated candidate after each visual refresh without applying the continuation verifier. To evaluate this design choice, we compare the deployed policy, which verifies candidates from $H2$ through $H15$, against a variant that applies the same frozen verifier from $H1$ onward. The propagators, verifier checkpoint and thresholds, Remaining-Plan Memory, recovery router, gripper guard, propagation ceiling, and bootstrap rules are otherwise unchanged. Both variants are evaluated on the same 1,992 VA episodes with identical initializations.

Table 10: Verify-from-H1 ablation on the paired VA evaluation.

Verification schedule	Success ↑	VLA Rate ↓	Steps/Refresh ↑	Latency ↓
$H2$–$H15$ **(FV-PMP)**	**64.86%**	**33.75%**	**2.963**	**19.26 ms**
$H1$–$H15$	60.33%	61.08%	1.637	34.93 ms

Applying the deployed verifier at $H1$ decreases Success by 4.53 percentage points while increasing VLA Rate by 27.34 points. Under paired episode analysis, FV-PMP succeeds alone on 223 episodes, whereas the Verify-from-H1 variant succeeds alone on 139 episodes (exact McNemar test, $p = 1.18 \times 10^{-5}$). Latency also increases from 19.26 to 34.93 ms per control step. These results indicate that applying the deployed verifier immediately after a fresh visual anchor induces unnecessarily frequent visual refreshes and degrades the success–invocation trade-off. This experiment evaluates extending the current frozen verifier to $H1$; it does not rule out a separately trained verifier specialized for the near-anchor candidate distribution.

D.3 Recovery-policy ablation

We next hold propagation and verification fixed and change only the response to an ordinary verifier rejection. Full-only always recomputes the complete denoising process. Corrective-only uses interval-

5 corrective refresh for ordinary rejection events while retaining mandatory Full refreshes for hard constraints and the depth ceiling. Adaptive is the complete FV-PMP router.

Table 11: Recovery-policy comparison on paired VA episodes. Uniformly using either recovery mode is inferior to adaptive selection.

Recovery policy	Success ↑	VLA Rate ↓
Full-only	62.16%	**33.23%**
Corrective-only	63.61%	35.72%
Adaptive FV-PMP	**64.86%**	33.75%

Adaptive recovery improves Success by 2.70 points over Full-only at a 0.52-point increase in VLA Rate. It also dominates Corrective-only, improving Success by 1.25 points while reducing VLA Rate by 1.97 points. Corrective refresh is therefore not a universally better or cheaper substitute for Full replanning; the two responses cover complementary recovery regimes.

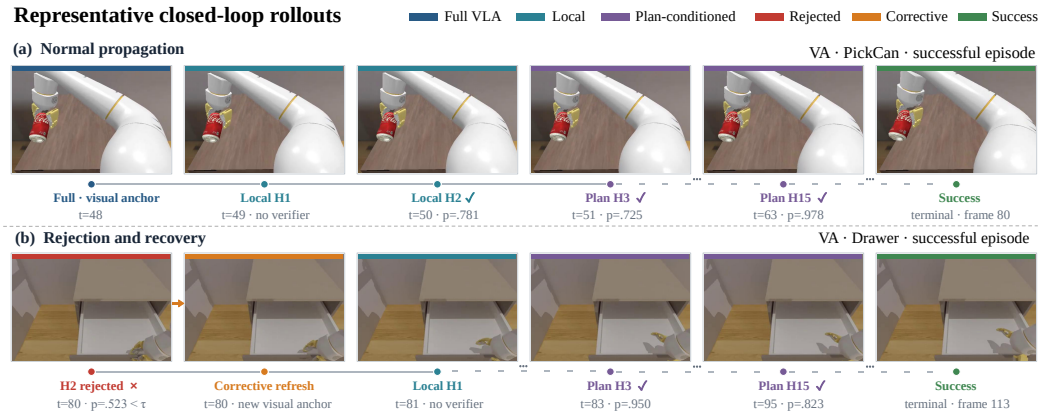


Figure 6: **Representative closed-loop rollouts on SIMPLER-VA.** (a) Normal propagation proceeds from a fresh visual anchor through Local propagation and Plan-Conditioned propagation to $H15$. (b) A rejected $H2$ candidate triggers corrective replanning and establishes a new visual anchor, after which propagation resumes and reaches $H15$ before task success. Dashed segments and ellipses denote omitted intermediate propagation steps. $H1$ is executed as a scheduled near-anchor continuation without verification; check marks denote accepted verified candidates.

D.4 Gripper hard-guard ablation

The gripper hard guard prevents verified propagation across a predicted binary gripper-state transition. To isolate its contribution, we disable only this guard while retaining the verifier, both propagators, Remaining-Plan Memory, the adaptive recovery router, the $H15$ ceiling, and all bootstrap rules. Table 12 reports the resulting paired comparison.

Table 12: Gripper hard-guard ablation on the paired 1,992-episode VA evaluation. The guard provides a modest success improvement without increasing the frequency of visually conditioned VLA queries.

Variant	Success ↑	VLA Rate ↓	Steps/Refresh ↑	Exec. Depth ↑
Full FV-PMP	**64.858%**	**33.746%**	**2.963**	**4.515**
w/o gripper guard	64.209%	34.379%	2.909	4.452

Removing the guard reduces Success by 0.649 percentage points. Under paired episode analysis, Full FV-PMP succeeds alone on 27 episodes, whereas the no-guard variant succeeds alone on 12 episodes (McNemar exact $p = 0.0237$). VLA Rate also increases by 0.633 points without the guard, indicating that unconstrained propagation across predicted gripper transitions can induce additional downstream recovery rather than eliminate visual refreshes.

The Full-policy trace contains 2,437 binding guard events that change the route that would otherwise have been selected, including 2,252 events on PickCan. This event concentration should not be

interpreted as an equal concentration of task-level benefit: the largest success-rate change occurs on MoveNear, which decreases from 77.17% to 74.83% when the guard is removed. The hard guard is therefore a lightweight conservative safeguard, not the primary source of FV-PMP's overall improvement.

D.5 Representative Closed-Loop Rollouts

Figure 6 illustrates two successful closed-loop executions. The normal case shows sustained propagation to $H15$ without an intervening visual refresh. In the recovery case, the verifier rejects the $H2$ candidate and triggers corrective replanning, after which propagation resumes from a new visual anchor and again reaches $H15$.

E Inference Algorithm

For clarity, the complete closed-loop procedure is summarized below. The notation $\textsc{Refresh}(q)$ denotes a visually conditioned CogACT query in mode $q \in \{\mathrm{Corrective}, \mathrm{Full}\}$.

1. Initialize the episode with three Full queries. Store the latest chunk A_r, initialize the two-action history, and set $h \leftarrow 1$.
2. At each control decision, form $c_t = [a_{t-1}; a_{t-2}; s_t]$.
3. If $h = 1$, generate $\hat{a}_t = f_{\mathrm{local}}(c_t)$ and execute it directly.
4. If $h = 2$, generate $\hat{a}_t = f_{\mathrm{local}}(c_t)$. Otherwise, compute $m_t = \phi(R_{r,h})$ and generate $\hat{a}_t = f_{\mathrm{plan}}([c_t; m_t])$.
5. For $h \geq 2$, compute $p_t = v_\psi([c_t; \hat{a}_t])$. Execute the candidate only if $p_t \geq \tau$ and it does not change the binary gripper state.
6. If the candidate is accepted, update the execution history and set $h \leftarrow h + 1$. After executing $H15$, force a Full query at the next control decision.
7. If the candidate is rejected without a gripper transition, choose Corrective refresh when $\tau_{\mathrm{low}} \leq p_t < \tau$ and Full refresh when $p_t < \tau_{\mathrm{low}}$, using $\tau_{\mathrm{low}} = 0.5$ at every verified depth. A gripper transition always triggers Full refresh. Execute the refreshed anchor action, replace A_r, and reset $h \leftarrow 1$.
8. Repeat until task success or the environment horizon is reached.