# SPEC$^2$: EFFICIENT VLA CONTROL THROUGH ACTION DECODING AND PROPAGATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Vision-language-action (VLA) models have demonstrated strong capabilities in language-conditioned robotic manipulation, but their high inference cost limits the frequency and responsiveness of closed-loop control. The computational burden stems from two components: sequential action generation within each VLA inference and repeated multimodal evaluation over successive observations. To address both sources of overhead, we present **Spec**$^2$, a unified framework that reduces action-decoding overhead and the frequency of redundant VLA inference while keeping the base VLA frozen. Verified Action Block Decoding first checks the previous VLA action as a complete candidate and, after the first mismatch, uses a separately trained noncausal proposal module to propose the unresolved suffix in parallel. The frozen VLA verifies all candidates with the same exact acceptance rule, reducing serial decoding overhead without relaxing token acceptance. Visuomotor-Gated Action Propagation selectively propagates VLA actions across adjacent control steps according to visual consistency and motion mode consistency, without requiring an additional learned action model. Because propagation alters the physical trajectory, we introduce closed-loop recoverability to characterize whether subsequent feedback can preserve the task outcome. Extensive experiments show that Spec$^2$ achieves a $\mathbf{2.7}\times$ mean inference speedup on LIBERO, with an average success rate 1.6 percentage points below OpenVLA. Paired closed-loop evaluations observe successful recovery in more than **92%** of evaluated futures. Action Propagation is also instantiated with the continuous-action CogACT policy, yielding about $\mathbf{1.7}\times$ acceleration on SimplerEnv. Moreover, in real-robot evaluations, Spec$^2$ achieves a $\mathbf{2.6}\times$ inference speedup and improves aggregate success from **73.3%** to **81.3%**. The code is available at `https://anonymous.4open.science/r/vla-review-artifact-5418`.

## 1 INTRODUCTION

Vision-language-action (VLA) models provide a powerful interface from natural language instructions and visual observations to robot control, but their multimodal scale makes repeated closed-loop inference computationally expensive (Brohan et al., 2023; Zitkovich et al., 2023; Black et al., 2025a; Kim et al., 2025b). The computational burden stems from two components: sequential action generation within each VLA inference and repeated multimodal evaluation over successive observations. Recent acceleration methods improve efficiency through compact action representations, parallel decoding, visual compression, caching, and selective policy invocation (Pertsch et al., 2025; Kim et al., 2025a; Xu et al., 2025; Yang et al., 2025b; Li et al., 2026). Motivated by these complementary sources of redundancy, we present Spec$^2$, a base-VLA-frozen framework that reduces action-decoding overhead and redundant VLA inference. VABD trains an auxiliary proposal module for candidate construction, whereas Action Propagation requires no learned action model or policy fine-tuning.

Within each VLA inference, exact speculative decoding allows a lightweight proposal model to suggest action tokens that the VLA verifies in parallel, reducing serial generation while retaining exact VLA verification (Leviathan et al., 2023; Chen et al., 2023). For policies with autoregressive action decoding, such as OpenVLA, however, each action contains only seven tokens assigned to fixed action dimensions. Constructing another short candidate sequence one token at a time can



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.

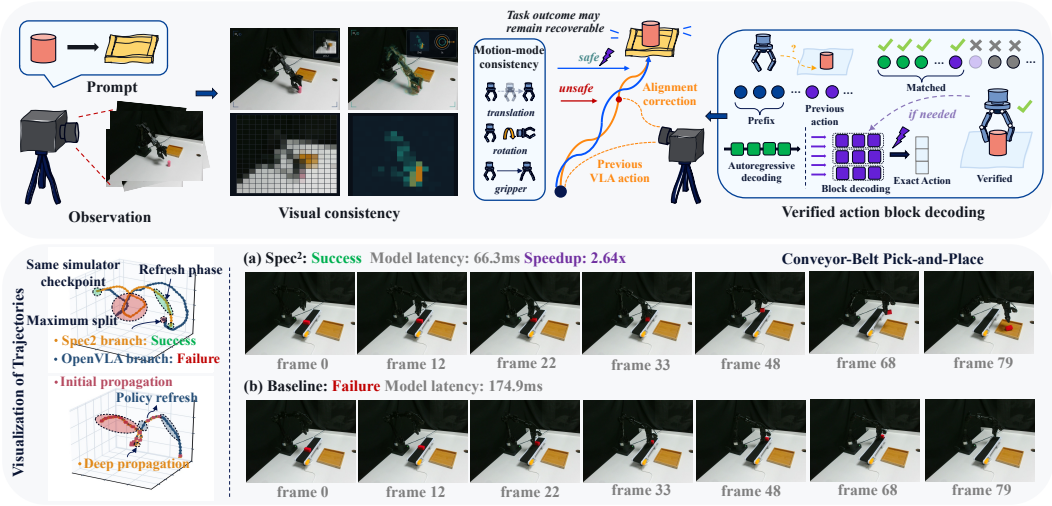



Figure 1: **Overview of the proposed Spec$^2$ framework.** The upper panel summarizes the two-level acceleration pipeline: Verified Action Block Decoding reduces the cost of VLA steps, while Visuomotor-Gated Action Propagation skips redundant VLA calls when visual and motion evidence support a reference action. The lower panel illustrates trajectory-level effects and a Conveyor-Belt Pick-and-Place real-robot rollout with the object transported at 3 m/min.

therefore consume a substantial fraction of the available latency (Kim et al., 2025b; Wang et al., 2025b). Adjacent control steps also provide a natural complete candidate: the previous VLA action. These observations motivate Verified Action Block Decoding, which first checks the previous VLA action as a complete candidate and, after the first mismatch, proposes only the unresolved suffix in parallel. Both candidate paths follow the same exact prefix-verification rule, reducing serial proposal overhead without relaxing token acceptance.

We further extend this efficiency gain across control steps by exploiting redundancy in repeated multimodal inference over consecutive observations. This leads to Visuomotor-Gated Action Propagation, which propagates a recent VLA action in a state-dependent manner when visual consistency and motion mode consistency support propagation, thereby reducing redundant VLA inference.

The two mechanisms operate under different correctness conditions. Candidate tokens in Verified Action Block Decoding are checked before execution, whereas a propagated action directly changes the physical trajectory. Under the Markov state-transition model used in Section 3, the effect of a propagated action is carried forward through the next state it induces. Subsequent evolution depends on that state, future controls, and disturbances, rather than on the identity of the preceding action alone. Moreover, the same task outcome need not correspond to a unique action sequence, as robotic manipulation often admits multiple valid action modes (Shafiullah et al., 2022; Chi et al., 2023). Exact action equality is therefore too restrictive as a criterion for action propagation. We introduce closed-loop recoverability, which asks whether subsequent causal feedback can still reach the same task outcome after a propagated action. This formulation distinguishes task recovery from action agreement and motivates paired evaluation across depths, separating negative and positive outcome flips under the same continuation policy and matched future randomness.

Figure 1 summarizes the resulting inference loop. At a VLA step, Verified Action Block Decoding checks the previous action and completes any mismatched suffix through parallel proposal and exact VLA verification. At a propagation step, the system evaluates visual consistency and motion mode consistency and executes a scaled reference action without invoking VLA inference. Spec$^2$ therefore reduces both the cost of required VLA inference and the frequency of redundant inference throughout a control trajectory.

Experiments across four LIBERO suites (Liu et al., 2023) show that Verified Action Block Decoding achieves a $1.24\times$ mean speedup, with an average success rate differing by only 0.05 percentage points from OpenVLA. Spec$^2$ achieves a $2.7\times$ mean inference speedup on LIBERO with an average

success rate 1.6 percentage points below OpenVLA. Action Propagation removes VLA inference from approximately half of LIBERO control steps; paired recovery exceeds 92% (Appendix C.2.5 and C.3), with gate sensitivity and controlled visual-robustness analyses in Appendices C.4 and C.7. We further instantiate Action Propagation with continuous-action CogACT (Li et al., 2024a), yielding about $1.7\times$ acceleration on SimplerEnv (Li et al., 2025) (Appendix C.5). In real-robot evaluations, including a conveyor-belt pick-and-place task, Spec$^2$ achieves a $2.6\times$ inference speedup and improves aggregate success from 73.3% to 81.3%; Appendix C.6 gives the data, training, deployment, and timing protocols.

The main contributions of Spec$^2$ are as follows:

- We introduce Verified Action Block Decoding, which keeps the base VLA frozen, verifies the previous action as a complete candidate, and uses a separately trained noncausal proposal module to propose only the unresolved suffix under a common exact verification rule; Appendix B gives the complete inference procedure and equivalence conditions.
- We introduce Visuomotor-Gated Action Propagation together with a formulation of closed-loop recoverability after action propagation. The propagation rule reduces redundant VLA inference according to visual consistency and motion mode consistency. The recoverability framework specifies the task-outcome criterion and motivates depth-wise paired evaluation of successes lost and gained under subsequent feedback; Appendix A gives the formal analysis.
- We establish a strong accuracy–efficiency operating point for OpenVLA acceleration on LIBERO, demonstrate the portability of Action Propagation to the continuous-action CogACT policy on SimplerEnv, and validate Spec$^2$ through static and conveyor-belt real-robot experiments on the Alicia-D robotic arm.

## 2 Related Work

**Speculative decoding for VLA actions.** Classical speculative decoding accelerates autoregressive generation by letting a lightweight proposal model produce candidates that the main model verifies in parallel (Stern et al., 2018; Leviathan et al., 2023; Chen et al., 2023). Medusa and EAGLE modify proposal topology or exploit intermediate model features, REST retrieves candidate continuations, and DFlash generates block proposals from main-model features (Cai et al., 2024; Li et al., 2024b; He et al., 2024; Chen et al., 2026). Spec-VLA extends neural speculative decoding to VLA action tokens and additionally studies relaxed acceptance between nearby discretized actions (Wang et al., 2025b). Verified Action Block Decoding instead first tests the previous policy action as a complete candidate and, after the first mismatch, jointly proposes the unresolved suffix as a single noncausal action block rather than generating it autoregressively token by token. Both candidate sources follow the same exact prefix-verification rule, changing proposal construction without relaxing token acceptance.

**Temporal redundancy and efficient VLA execution.** Prior work reduces repeated VLA computation through visual-token pruning, caching, or action reuse, including SparseVLM, FoPru, PruMerge, FastVLM, VisionZip, VLA-Cache, FlashVLA, SpecPrune-VLA, ActionCache, SP-VLA, HeiSD, ADP, and AC$^2$-VLA (Lakshminarayanan et al., 2017; Zhang et al., 2025; Jiang et al., 2024; Shang et al., 2025; Vasu et al., 2025; Yang et al., 2025a; Xu et al., 2025; Tan et al., 2025; Wang et al., 2025a; Oi et al., 2026; Li et al., 2026; Zheng et al., 2026; Pei et al., 2025; Yu et al., 2026). Other methods alter decoding or action interfaces: OpenVLA-OFT combines parallel decoding, chunking, and continuous regression; PD-VLA uses parallel fixed-point chunk decoding; CEED-VLA adds consistency training and early exit; VOTE combines parallel action prediction with trajectory ensemble voting; FAST and real-time chunking alter action representation or scheduling (Kim et al., 2025a; Song et al., 2025a,b; Lin et al., 2025; Pertsch et al., 2025; Black et al., 2025b). Because these systems differ in policy interface, training, runtime, or timing protocol, we treat their published speedups as contextual rather than hardware-matched Table 1 points; Appendix C.1 defines our local comparison protocol. In contrast, Spec$^2$ keeps the base VLA frozen: VABD trains only an auxiliary proposal for candidate construction, while Action Propagation adds no learned action generator.

## 3 CLOSED-LOOP RECOVERABILITY AFTER ACTION PROPAGATION

Verified Action Block Decoding retains exact VLA verification (Appendix B). Action propagation instead changes the physical state without computing the current VLA action. We formulate recovery under subsequent causal feedback and depth-wise paired evaluation, rather than action equality or a deployed-gate guarantee.

$$S_{t+1} = F(S_t, U_t, W_t), \qquad O_t = G(S_t), \tag{1}$$

where $S_t$, $O_t$, and $U_t$ denote the state, observation, and executed command, and $W_t$ collects future randomness. At reference time $\tau$, the VLA produces $\bar{u}_\tau$; at propagation depth $d \geq 1$, $\mathcal{P}_d(\bar{u}_\tau)$ is the propagated command. Let $\xi^\star$ be a successful reference rollout, $J^\star = J(\xi^\star)$ its outcome, and $\Pi_c$ the admissible causal continuation policies. Let $D_J$, with tolerance $\varepsilon_J$, measure discrepancy between task outcomes. For a propagation-boundary state $s$, define

$$\begin{aligned} q_d(s,\pi) &= \Pr[D_J(J(\Xi_d^{\text{prop}}(s,\pi)), J^\star) \leq \varepsilon_J \mid S_{\tau+d} = s], \\ \mathcal{R}_d &= \{s : \exists \pi \in \Pi_c,\ q_d(s,\pi) \geq 1-\rho\}, \\ p_d &= \Pr[S_{\tau+d} \in \mathcal{R}_d]. \end{aligned} \tag{2}$$

Here $q_d(s,\pi)$ is the recovery probability from a fixed state under a fixed continuation policy. $\mathcal{R}_d$ asks whether some admissible causal policy attains probability at least $1-\rho$, while $p_d$ averages membership in this latent recoverable set over states visited at depth $d$. Equality with the current VLA action is not required.

To characterize recoverability across propagation depths independently of the deployed refresh horizon, let $A_d = \{S_{\tau+d} \in \mathcal{R}_d\}$ and let $E_d$ measure normalized deviation from the reference rollout at the corresponding propagation-boundary state. Assume recoverability requires membership in a necessary margin $M_d = \{E_d \leq r_d\}$, with $r_{d+1} \leq r_d$, and that one additional propagated transition does not stochastically improve next-boundary deviation before continuation feedback. Defining the abstraction gap $\epsilon_d^{\text{abs}} = \Pr(M_d) - p_d$, we obtain

$$p_{d+1} \leq p_d + \epsilon_d^{\text{abs}}. \tag{3}$$

Theorem 1 in Appendix A.2 proves this bound under the stated assumptions, not a deployment guarantee. It does not require $p_d$ to decrease monotonically: the abstraction gap may change with depth. The bound precedes the continuation feedback in Equation (2), motivating depth-wise recovery evaluation.

Because $\mathcal{R}_d$ depends on future outcomes, it is unavailable at inference. Section 4 therefore uses visual and motion-mode consistency as observable evidence and depth-dependent scaling to limit repeated commands; probabilistic calibration to $\mathcal{R}_d$ is not claimed. The empirical analysis instead fixes the deployed continuation policy $\pi_{\text{VLA}}$, for which the policy-conditioned recoverable set $\mathcal{R}_d^{\pi_{\text{VLA}}} \subseteq \mathcal{R}_d$. From the same checkpoint, the current-action and propagation branches execute their prescribed first actions, then share $\pi_{\text{VLA}}$ and matched future randomness. If $Y_d^{\text{current}}$ and $Y_d^{\text{prop}}$ denote their binary outcomes, then

$$\begin{aligned} &\Pr[Y_d^{\text{prop}} = 1] - \Pr\big[Y_d^{\text{current}} = 1\big] \\ &= \underbrace{\Pr\big[Y_d^{\text{current}} = 0, Y_d^{\text{prop}} = 1\big]}_{\text{positive outcome flips}} - \underbrace{\Pr\big[Y_d^{\text{current}} = 1, Y_d^{\text{prop}} = 0\big]}_{\text{negative outcome flips}}. \end{aligned} \tag{4}$$

Equation (4) is a paired-outcome identity, not a consequence of Theorem 1: aggregate recovery can conceal lost and gained successes. For a single gated intervention, Proposition A.1 defines admission coverage $c_d$ and conditional positive/negative flip rates $h_d^+, h_d^-$, giving recovery change $c_d(h_d^+ - h_d^-) \geq -c_d h_d^-$. Appendix A.6 separately links scaling and refresh to task loss under explicit transition, feedback, and task-margin assumptions.

## 4 METHOD

Spec$^2$ alternates between VLA steps and propagation steps, as shown in Figure 2. A VLA step evaluates the frozen base VLA on the current observation and instruction and produces an action through

Verified Action Block Decoding. Its separately trained one-layer proposal network is used only for candidate construction, while the base VLA remains frozen during deployment. A propagation step omits VLA inference and executes a transformed reference action. The previous action sequence serves only as a candidate under the current VLA context, whereas the reference observation and action determine the command executed during propagation.

### 4.1 Verified Action Block Decoding

An OpenVLA action contains $K = 7$ tokens ordered by action dimension. Let

$$\bar{\mathbf{A}}_\tau = \left(\bar{a}_\tau^1, \ldots, \bar{a}_\tau^K\right)$$

denote the previous action sequence verified by the VLA, and let $(o_t, g)$ be the current observation and instruction.

**Previous-Action Verification.** Appending the first $K-1$ tokens of $\bar{\mathbf{A}}_\tau$ to the current vision and language context yields aligned VLA predictions $\mathbf{y}_t = (y_t^1, \ldots, y_t^K)$. Define

$$\begin{aligned} \kappa_t &= \begin{cases} \min\left\{i : \bar{a}_\tau^i \neq y_t^i\right\}, & \text{if a mismatch exists,} \\ K+1, & \text{otherwise,} \end{cases} \\ \mathcal{V}(\bar{\mathbf{A}}_\tau; o_t, g) &= \begin{cases} \bar{\mathbf{A}}_\tau, & \kappa_t = K+1, \\ \bar{\mathbf{A}}_\tau^{1:\kappa_t-1} \oplus y_t^{\kappa_t}, & \kappa_t \leq K. \end{cases} \end{aligned} \tag{5}$$

A complete match returns the previous sequence directly. Otherwise, tokens after the first mismatch are discarded, and the VLA prediction at that position extends the verified prefix.

**Parallel Suffix Proposal.** Suppose the verified prefix ends at position $a$. Selected frozen VLA layers are fused into context $C_a$ using RMSNorm (Zhang & Sennrich, 2019), and all unresolved action positions are proposed jointly:

$$\begin{aligned} C_a &= \mathrm{RMSNorm}\left(W_f \, \mathrm{Concat}_{\ell \in \mathcal{S}} \, H_{\mathrm{ctx},\leq a}^{(\ell)}\right), \\ \hat{\mathbf{A}}_{a+1:K} &= \mathrm{SlotArgMax}_{\mathcal{V}_A}\left[W_A D_\phi(\mathbf{X}_a; C_a)\right], \\ r_a &= \max\left\{r : \hat{a}_t^{a+j} = v_{a,j} \text{ for all } 1 \leq j \leq r\right\}. \end{aligned} \tag{6}$$

Here, $\mathbf{X}_a$ comprises $K - a$ slots: the verified-anchor embedding followed by fixed-mask embeddings, with action-dimension embeddings identifying predictions $a+1{:}K$; $H_{\mathrm{ctx},\leq a}^{(\ell)}$ are selected-layer outputs for prompt and verified-prefix positions from the current causal VLA prefill/verification pass. $D_\phi$ is noncausal, $W_A$ is the frozen action-row projection over $\mathcal{V}_A$, and $v_{a,j}$ is the aligned causal-verification prediction. The decoder accepts the first $r_a$ consecutive matching proposals. If the suffix remains incomplete, it appends the next VLA correction token, discards later proposals, and repeats from the extended prefix. Thus, candidate generation is parallel, while acceptance remains causal and exact. Appendix B.4 establishes equivalence to serial greedy VLA decoding under identical exact numerical execution.

Let $f_{\mathrm{VLA}}$ be the VLA inference rate, $p_{\mathrm{match}}$ the complete previous-action match probability, $C_{\mathrm{ctx}}$ the cost of context processing and Previous-Action Verification, $C_{\mathrm{mis}}$ the expected additional proposal and verification cost after a mismatch, and $C_{\mathrm{prop}}$ the cost of a propagation step. The mean computation per control step is

$$\mathbb{E}[C] = f_{\mathrm{VLA}}\left[C_{\mathrm{ctx}} + (1 - p_{\mathrm{match}})C_{\mathrm{mis}}\right] + (1 - f_{\mathrm{VLA}})C_{\mathrm{prop}}. \tag{7}$$

Verified Action Block Decoding reduces the cost of VLA steps, while action propagation reduces how often that cost is incurred.

### 4.2 Visuomotor-Gated Action Propagation

At each VLA step, Spec$^2$ stores the current observation and unscaled VLA action as a reference pair $(o_\tau, \bar{u}_\tau)$. This pair remains fixed until the next VLA step.

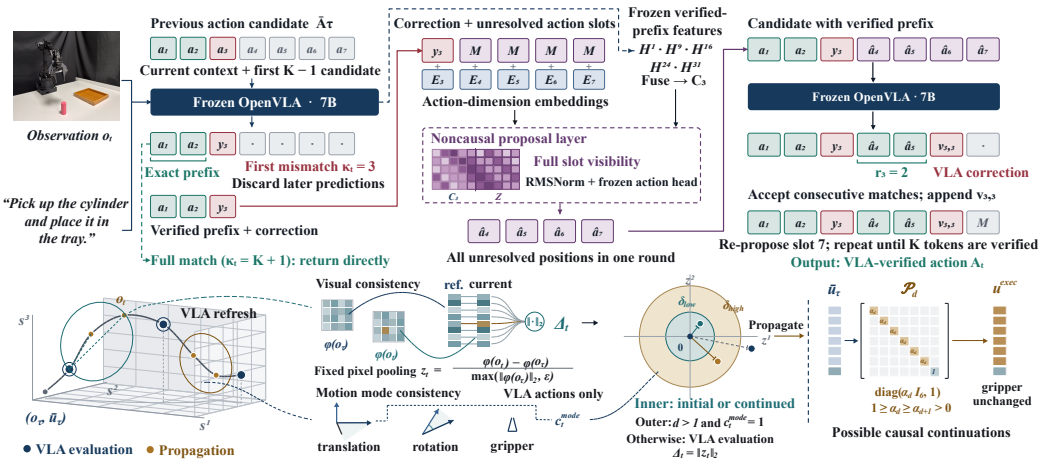



Figure 2: **Details of the Spec$^2$ framework.** The top row shows Verified Action Block Decoding, which tests a previous action candidate and completes mismatched suffixes through parallel proposal with exact VLA verification. The bottom row shows Visuomotor-Gated Action Propagation, which uses visual consistency, motion mode consistency, and depth-scaled reference actions to omit redundant VLA inference between verified VLA steps.

**Visual consistency.** A fixed spatial pooling operator produces a low-cost visual signature $\phi(o)$. Propagation is gated by the normalized, reference-relative change

$$\Delta_t = \frac{\|\phi(o_t) - \phi(o_\tau)\|_2}{\max\left(\|\phi(o_\tau)\|_2, \epsilon\right)}. \tag{8}$$

Because $o_\tau$ is refreshed only at VLA steps and remains fixed during propagation, consecutive propagation decisions are measured against the same last-VLA reference rather than the immediately preceding frame. Continued scene evolution therefore tends to increase departure from that reference, while either gate rejection or the refresh horizon triggers a fresh VLA evaluation. A common positive rescaling also cancels when the denominator is not $\epsilon$-clipped. Thresholds $\delta_{\text{low}}$ and $\delta_{\text{high}}$ define the inner, outer, and rejected regions.

**Motion mode consistency.** Spec$^2$ compares actions produced at consecutive VLA steps. Their translation and rotation directions must be nonopposing, with near-stationary vectors treated as compatible, and their gripper states must agree. Propagation steps do not update this state because no current VLA action is computed. Let $c_t^{\text{mode}} \in \{0, 1\}$ denote the resulting consistency indicator.

For candidate propagation depth $d \geq 1$, the visual–motion gate is

$$\mathcal{I}_{\text{prop}}(t, d) = \begin{cases} 1, & \Delta_t \leq \delta_{\text{low}}, \\ 1, & d > 1,\ \delta_{\text{low}} < \Delta_t \leq \delta_{\text{high}},\ c_t^{\text{mode}} = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The inner region permits initial and continued propagation. The outer region permits continued propagation only with motion mode consistency. When propagation is selected, the continuous components of the reference action are scaled with depth:

$$\mathcal{P}_d\left(u^{\text{tr}}, u^{\text{rot}}, g\right) = \left(\alpha_d u^{\text{tr}}, \alpha_d u^{\text{rot}}, g\right), \qquad 1 \geq \alpha_d \geq \alpha_{d+1} > 0. \tag{10}$$

For LIBERO, a shared configuration is used across all four suites; Appendix B.5 specifies the visual–motion settings and refresh horizon, Appendix C.4 reports sensitivity, and Appendix C.7 stress-tests the visual gate. The gripper command is unchanged. Propagation continues only while the gate admits the candidate depth and that depth does not exceed the deployed refresh horizon; otherwise, Spec$^2$ performs a fresh VLA evaluation and updates the reference pair. Propagation steps themselves never update $o_\tau$. A complete match in Previous-Action Verification remains a VLA step because the current VLA has been evaluated. Only a propagation step omits VLA inference.

# 5 EXPERIMENTS

We evaluate Spec$^2$ on LIBERO, Action Propagation with CogACT on SimplerEnv, and the full framework in real-world tasks. LIBERO uses 50 episodes per task and a common local policy-path timing protocol on an RTX 4090; Appendix C gives the complete protocols.

## 5.1 SIMULATION RESULTS

Tables 1 and 2 summarize the simulation results. On LIBERO, Verified Action Block Decoding reaches $1.24\times$ mean speedup and 75.30% success, compared with 75.25% for OpenVLA. Spec$^2$ reaches $2.69\times$ and 73.70%, only 1.55 points below OpenVLA, with suite speedups of $2.29\times$–$3.16\times$. Among locally evaluated methods, Spec$^2$ gives the strongest Table 1 accuracy–efficiency point (Appendix C.1). Figure 3 shows representative successful trajectories.

To test generalization beyond OpenVLA's discrete action tokens, we instantiate Action Propagation with continuous-action CogACT. Table 2 reports task-macro speedups of $1.74\times$, $1.73\times$, and $1.72\times$ on Google Visual Matching, Google Visual Aggregation, and WindowX, while CogACT is invoked on 59.26%, 59.65%, and 54.01% of control steps. This component-matched comparison isolates Action Propagation.

## 5.2 REAL-WORLD RESULTS

In the real-world evaluation, we test Pick, Pick-and-place, and Conveyor-Belt Pick-and-Place with 50 trials per method/task; the conveyor belt runs at 3 m/min. Spec$^2$ achieves 90.0%, 82.0%, and 72.0% success with $3.00\times$, $2.19\times$, and $2.64\times$ model-path speedups, versus OpenVLA's 94.0%, 80.0%, and 46.0%. Across all three tasks, Spec$^2$ raises aggregate success from 73.3% to 81.3% and reduces mean model-path latency from 180.25 ms to 70.32 ms, a $2.56\times$ speedup. Figures 1 and 4 show the moving and static tasks, respectively.

## 5.3 ANALYSIS

**Action decoding.** Table C.23 isolates Parallel Suffix Proposal by disabling Previous-Action Verification. Accepted candidates increase from 0.64 to 1.19 per round and verified advance from 1.64 to 2.25, cutting rounds by 27.2% and latency from 178.5 to 157.6 ms ($1.03\times$ to $1.16\times$). Previous-Action Verification raises the full VABD speedup to $1.24\times$ despite low reuse: only 15.4% of 24,055 logged VLA actions fully match the previous action, while 69.2% mismatch within the first two positions (Appendix C.2.6). Figure 5 shows a diagnostic temporal replay.

**Inference frequency.** Propagation covers 50.39% of 449,361 LIBERO steps (45.62–53.03% by suite; 32.62% continued; Table C.6), so the $2.69\times$ gain combines cheaper VLA steps with fewer VLA calls. CogACT invokes the VLA on 54.01–59.65% of SimplerEnv steps. On static real-world tasks, propagation covers 53% of steps and reduces FLOPs to 50.9%; Conveyor-Belt Pick-and-Place reaches $2.64\times$ model-path speedup.

**Propagation rule and recovery.** The LIBERO visual gate uses a shared operating point across all four suites. Sensitivity and controlled perturbations on 7,502 reference–current pairs assess its behavior (Appendix C.4 and C.7); Appendix C.3.5 decomposes archived screening outcomes into admission coverage and admitted flips using Proposition A.1. This diagnostic does not validate the deployed gate or establish calibration or optimality. Motion mode yields 73.7% SR/$2.69\times$ versus 73.4%/$2.42\times$ with exact action equality; Table 1 isolates action decoding at $1.24\times$.

Initial-only propagation reduces SR by 3.2 points and speedup by $0.27\times$, with losses confined to Goal and Long. A consistency-admitted second propagation can preserve coherent motion, whereas refreshing may switch modes. Table 5 records slightly higher propagated-action recovery and positive flips at the deployed second depth. Table C.11's paired interval includes zero; neither result establishes universal superiority of continued propagation. Removing motion mode gains $0.11\times$ but loses 1.5 SR points. Removing scaling loses 5.5 points with little speed gain, consistent with larger repeated-command deviations. Removing the visual boundary gains $0.66\times$ but loses 6.8 points. This supports selective admission within the tested configuration, without identifying optimal thresholds.

Table 1: **LIBERO accuracy–efficiency comparison.** For locally evaluated rows, cells report SR (%)/speedup over 500 episodes per suite, with speedup measured under the common local policy-path timing protocol defined in Appendix C.1. The first three rows use exact VLA verification. †: cited prior work under source protocols, not hardware-matched locally. $R$: relative-position preservation; $S$: Canny edges. Timing and provenance: Appendix C.1.

| Method | Success Rate (%, ↑) / Speedup (↑) | | | | |
|---|---|---|---|---|---|
| | Goal | Spatial | Object | Long | Average |
| OpenVLA | 74.2 / 1.00 | **87.0** / 1.00 | **88.4** / 1.00 | 51.4 / 1.00 | 75.25 / 1.00 |
| Spec-VLA | 76.8 / 1.02 | 85.0 / 1.00 | 87.6 / 1.10 | **54.4** / 0.99 | 75.95 / 1.03 |
| **VABD (ours)** | 77.6 / **1.30** | 86.4 / **1.17** | 87.2 / **1.19** | 50.0 / **1.29** | 75.30 / **1.24** |
| SparseVLM† | 74.2 / 1.33 | 83.4 / 1.33 | 84.0 / 1.33 | **52.8** / 1.33 | 73.60 / 1.33 |
| FoPru+R† | 59.8 / 1.29 | 71.6 / 1.30 | 81.2 / 1.29 | 26.2 / 1.35 | 59.70 / 1.31 |
| PruMerge+R† | 0.0 / 1.54 | 0.0 / 1.27 | 0.0 / 1.32 | 0.0 / 1.32 | 0.00 / 1.36 |
| FastVLM+R+S† | 73.2 / 1.21 | 79.8 / 1.12 | 77.0 / 1.11 | 36.6 / 1.20 | 66.65 / 1.16 |
| VisionZip+R+S† | 46.0 / 1.20 | 34.2 / 1.19 | 47.4 / 1.23 | 4.6 / 1.22 | 33.05 / 1.21 |
| SP-VLA† | 73.6 / 1.66 | 80.0 / 1.47 | 82.4 / 1.44 | 51.6 / 1.42 | 71.90 / 1.50 |
| Spec-VLA relaxed | 73.4 / 1.29 | **86.2** / 1.15 | **85.0** / 1.31 | 49.8 / 1.11 | 73.60 / 1.21 |
| **Spec$^2$ (ours)** | **74.4** / **3.16** | 82.8 / **2.29** | **85.0** / **2.30** | 52.6 / **3.01** | **73.70** / **2.69** |

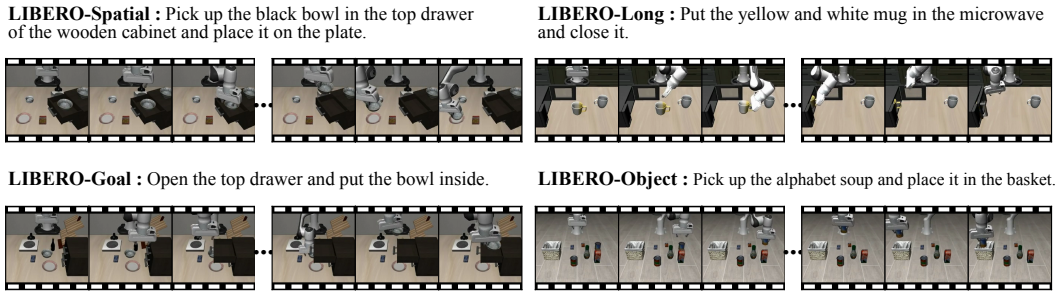



Figure 3: **Representative Spec$^2$ rollouts across four LIBERO suites.** The displayed frames span distinct phases of successful trajectories.

Table 2: **Portability of Action Propagation to continuous-action CogACT on SimplerEnv.** Local frozen CogACT is compared with Action Propagation only, without a proposal network or action generator. Speedup is the arithmetic macro average of four task-level ratios; pooled decision latencies are in Table C.17.

| Setting | Method | Success Rate (%, ↑) / Speedup (↑) | | | | Macro avg. | FLOPs (%, ↓) |
|---|---|---|---|---|---|---|---|
| | | PickCan | MoveNear | Drawer | DrawerApple | | |
| **Visual Matching** | CogACT | **93.67**/1.00 | **84.17**/1.00 | **75.93**/1.00 | 45.37/1.00 | **74.78**/1.00 | 100.00 |
| | **+ Propagation** | 88.67/**2.15** | 82.08/**2.02** | 73.61/**1.69** | **46.30/1.09** | 72.66/**1.74** | **59.26** |
| **Visual Aggregation** | CogACT | 89.58/1.00 | 76.33/1.00 | 26.72/1.00 | **52.91**/1.00 | 61.38/1.00 | 100.00 |
| | **+ Propagation** | **89.94/2.23** | **77.67/1.98** | **39.42/1.63** | 48.15/**1.07** | **63.79/1.73** | **59.65** |
| | | **PutSpoon** | **PutCarrot** | **StackBlock** | **PutEggplant** | | |
| **WindowX** | CogACT | 58.33/1.00 | 41.67/1.00 | 12.50/1.00 | 62.50/1.00 | 43.75/1.00 | 100.00 |
| | **+ Propagation** | **66.67/1.69** | **58.33/1.70** | **16.67/1.64** | **83.33/1.84** | **56.25/1.72** | **54.01** |

Visual admission selects when to propagate, scaling limits deviation, and motion mode supports continued propagation.

Matched recovery spans 92.42%–95.45%. Depths 1–2 pool to 94.60%/93.94% current/propagated recovery and 4.45%/3.79% negative/positive flips; this is policy-conditioned, not gate calibration.

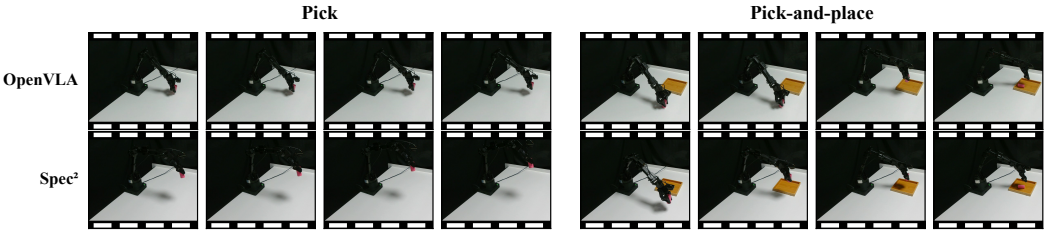



Figure 4: Static real-robot rollouts for Pick and Pick-and-place at matched video indices; the conveyor-belt rollout appears in Figure 1.

Table 3: Real-world robot results (50 trials per method/task). Cells report SR (%)/speedup; Conveyor-Belt P&P uses 3 m/min. Propagation/FLOPs use the static-task logs; latency is the three-task server-side mean excluding network and robot motion.

| **Method** | **Success Rate (%, ↑) / Speedup (↑)** | | | | **Propagation ratio (%, ↑)** | **FLOPs (%, ↓)** | **Model-path latency (ms, ↓)** |
|---|---|---|---|---|---|---|---|
| | | | **Conveyor-Belt** | | | | |
| | **Pick** | **Pick-and-place** | **P&P** | **Average** | | | |
| OpenVLA | **94.00** / 1.00 | 80.00 / 1.00 | 46.00 / 1.00 | 73.33 / 1.00 | 0.00 | 100.0 | 180.25 |
| **Spec²** | 90.00 / **3.00** | **82.00 / 2.19** | **72.00 / 2.64** | **81.33 / 2.56** | **53.00** | **50.9** | **70.32** |

Table 4: Action-propagation ablation; suite results: Table C.10.

| Condition | SR (%) | Speedup |
|---|---|---|
| **Spec²** | 75.0 | 2.58× |
| Initial only | 71.8 | 2.31× |
| w/o motion mode | 73.5 | 2.69× |
| w/o scaling | 69.5 | 2.63× |
| w/o visual bound. | 68.2 | 3.24× |

Table 5: Policy-conditioned recovery (66 checkpoints/depth; 8 matched futures/checkpoint). Δ: propagated minus current; Negative/Positive: outcome flips. Depths 3–4 are diagnostic interventions beyond deployed $H_{\text{refresh}} = 2$. Checkpoint-cluster intervals: Appendix C.3.

| Depth | Current | Propagated | Δ | Negative | Positive |
|---|---|---|---|---|---|
| 1 | 94.70 | 92.42 | −2.27 | 5.68 | 3.41 |
| 2 | 94.51 | **95.45** | +0.95 | 3.22 | 4.17 |
| 3 | 93.56 | 94.32 | +0.76 | 4.36 | 5.11 |
| 4 | 93.56 | 94.13 | +0.57 | 4.36 | 4.92 |

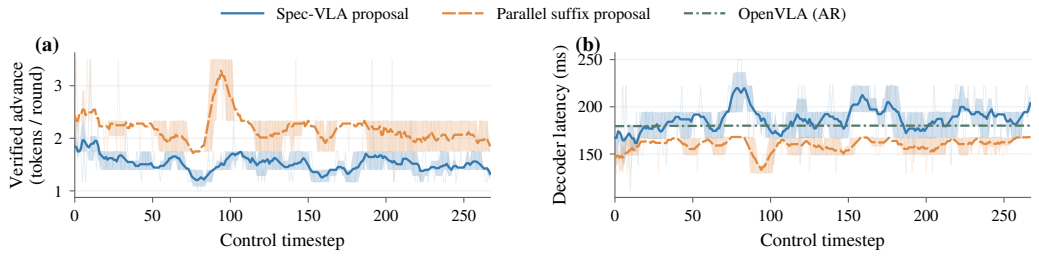



Figure 5: Diagnostic replay on 268 LIBERO observations: (a) verified advance; (b) latency. Lines/bands: 11-step means/local IQR. Replay protocol: Appendix C.8.

## 6 Conclusion

Spec² accelerates VLA control at two complementary levels: Verified Action Block Decoding reduces exact action-decoding cost, while Visuomotor-Gated Action Propagation skips redundant VLA calls across control steps. Closed-loop recoverability captures the trajectory-level effects of propagation, with recovery above 92% in paired evaluations. Spec² achieves a 2.7× mean speedup on LIBERO with a 1.6-point lower success rate, demonstrates Action Propagation portability to CogACT at about 1.7× on SimplerEnv, and reaches 2.6× in real-world evaluation while raising aggregate success from 73.3% to 81.3%.

### AI USE STATEMENT

AI tools were used during manuscript preparation and revision for language editing, drafting suggestions, organization of presentation, and assistance in summarizing and interpreting experimental outputs. Additionally, AI coding assistants were employed to support the drafting of experimental control scripts, data processing pipelines, ablation analysis code, and LaTeX formatting fragments during implementation and manuscript production. All research decisions, experimental procedures, core algorithm design, code verification and debugging, numerical results, theoretical conclusions, and the full content of the final manuscript and associated codebase were independently conceived, executed, and completed by the authors.

### REPRODUCIBILITY STATEMENT

Definitions, assumptions, proofs, and recovery analysis are provided in Section A, inference and implementation details in Section B, and evaluation protocols, pairing rules, timing scope, ablations, robustness tests, and additional results in Section C. The accompanying artifact records frozen configurations, exact command lines, analysis scripts, result manifests, checkpoint provenance, and checksums supporting all locally reported results.

## REFERENCES

K. Black, N. Brown, D. Driess, et al. $\pi_0$: A vision-language-action flow model for general robot control. In *RSS*, 2025.

K. Black, M. Y. Galliker, and S. Levine. Real-time execution of action chunking flow policies. In *NeurIPS*, 2025.

A. Brohan, N. Brown, J. Carbajal, et al. RT-1: Robotics Transformer for real-world control at scale. In *RSS*, 2023.

T. Cai, Y. Li, Z. Geng, H. Peng, J. D. Lee, D. Chen, and T. Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *ICML*, 2024.

C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper. Accelerating large language model decoding with speculative sampling. *arXiv:2302.01318*, 2023.

J. Chen, Y. Liang, and Z. Liu. DFlash: Block diffusion for flash speculative decoding. In *ICML*, 2026.

C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion Policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

Z. He, Z. Zhong, T. Cai, J. D. Lee, and D. He. REST: Retrieval-based speculative decoding. In *NAACL*, 2024.

E. J. Hu, Y. Shen, P. Wallis, et al. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

L. Jiang, W. Huang, T. Liu, Y. Zeng, J. Li, L. Cheng, and X. Xu. FoPru: Focal pruning for efficient large vision-language models. *arXiv:2411.14164*, 2024.

M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *RSS*, 2025.

M. J. Kim, K. Pertsch, S. Karamcheti, et al. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning (CoRL 2024)*, volume 270 of PMLR, pages 2679–2713, 2025.

A. S. Lakshminarayanan, S. Sharma, and B. Ravindran. Dynamic action repetition for deep reinforcement learning. In *AAAI*, 2017.

Y. Leviathan, M. Kalman, and Y. Matias. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

Q. Li, Y. Liang, Z. Wang, et al. CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv:2411.19650*, 2024.

Y. Li, F. Wei, C. Zhang, and H. Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *ICML*, 2024.

X. Li, K. Hsu, J. Gu, et al. Evaluating real-world robot manipulation policies in simulation. In *Proceedings of the 8th Conference on Robot Learning (CoRL 2024)*, volume 270 of PMLR, pages 3705–3728, 2025.

Y. Li, Y. Meng, Z. Sun, K. Ji, C. Tang, J. Fan, X. Ma, S.-T. Xia, Z. Wang, and W. Zhu. SP-VLA: A joint model scheduling and token pruning approach for VLA model acceleration. In *ICLR*, 2026.

J. Lin, A. Taherin, A. Akbari, A. Akbari, L. Lu, G. Chen, T. Padir, X. Yang, W. Chen, Y. Li, X. Lin, D. Kaeli, P. Zhao, and Y. Wang. VOTE: Vision-language-action optimization with trajectory ensemble voting. *arXiv:2507.05116*, 2025.

B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS Datasets and Benchmarks*, 2023.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

R. Oi, H. Otsuka, K. Matsushima, et al. ActionCache: Training-free acceleration for vision-language-action models with action caching and refinement. *arXiv:2607.06370*, 2026.

OpenVLA Team. OpenVLA performance troubleshooting and fine-tuning guidance. `https://github.com/openvla/openvla#vla-performance-troubleshooting`, 2024.

X. Pei, Y. Chen, S. Xu, Y. Wang, Y. Shi, and C. Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation. *arXiv:2509.22093*, 2025.

K. Pertsch, K. Stachowicz, B. Ichter, et al. FAST: Efficient action tokenization for vision-language-action models. In *RSS*, 2025.

S. Ramos, S. Girgin, L. Hussenot, et al. RLDS: An ecosystem to generate, share and use datasets in reinforcement learning. *arXiv:2111.02767*, 2021.

N. M. M. Shafiullah, Z. Cui, A. Altanzaya, and L. Pinto. Behavior Transformers: Cloning $k$ modes with one stone. In *NeurIPS*, 2022.

Y. Shang, M. Cai, B. Xu, Y. J. Lee, and Y. Yan. LLaVA-PruMerge: Adaptive token reduction for efficient large multimodal models. In *ICCV*, 2025.

W. Song, J. Chen, P. Ding, et al. PD-VLA: Accelerating vision-language-action model integrated with action chunking via parallel decoding. In *IROS*, 2025.

W. Song, J. Chen, P. Ding, Y. Huang, H. Zhao, D. Wang, and H. Li. CEED-VLA: Consistency vision-language-action model with early-exit decoding. *arXiv:2506.13725*, 2025.

M. Stern, N. Shazeer, and J. Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *NeurIPS*, 2018.

X. Tan, Y. Yang, P. Ye, J. Zheng, B. Bai, X. Wang, J. Hao, and T. Chen. Think twice, act once: Token-aware compression and action reuse for efficient inference in vision-language-action models. *arXiv:2505.21200*, 2025.

P. K. A. Vasu, F. Faghri, C.-L. Li, et al. FastVLM: Efficient vision encoding for vision-language models. In *CVPR*, 2025.

A. Vaswani, N. Shazeer, N. Parmar, et al. Attention is all you need. In *NeurIPS*, 2017.

H. Wang, J. Xu, J. Pan, Y. Zhou, and G. Dai. SpecPrune-VLA: Accelerating vision-language-action models via action-aware self-speculative pruning. *arXiv:2509.05614*, 2025a.

S. Wang, R. Yu, Z. Yuan, C. Yu, F. Gao, Y. Wang, and D. F. Wong. Spec-VLA: Speculative decoding for vision-language-action models with relaxed acceptance. In *EMNLP*, 2025b.

E. B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

S. Xu, Y. Wang, C. Xia, D. Zhu, T. Huang, and C. Xu. VLA-Cache: Towards efficient vision-language-action model via adaptive token caching in robotic manipulation. *arXiv:2502.02175*, 2025.

S. Yang, Y. Chen, Z. Tian, C. Wang, J. Li, B. Yu, and J. Jia. VisionZip: Longer is better but not necessary in vision-language models. In *CVPR*, 2025.

Y. Yang, Y. Wang, Z. Wen, L. Zhongwei, C. Zou, Z. Zhang, C. Wen, and L. Zhang. EfficientVLA: Training-free acceleration and compression for vision-language-action models. In *NeurIPS*, 2025.

W. Yu, T. Wang, F. Li, J. Li, and L. Zhu. AC$^2$-VLA: Action-context-aware adaptive computation in vision-language-action models for efficient robotic manipulation. *arXiv:2601.19634*, 2026.

B. Zhang and R. Sennrich. Root mean square layer normalization. In *NeurIPS*, 2019.

Y. Zhang, C.-K. Fan, J. Ma, et al. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *ICML*, 2025.

Z. Zheng, Z. Mao, S. Tian, et al. HeiSD: Hybrid speculative decoding for embodied vision-language-action models with kinematic awareness. *arXiv:2603.17573*, 2026.

B. Zitkovich, T. Yu, S. Xu, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023.

# A Formal Analysis of Closed-Loop Recoverability

This appendix develops the analysis used in Section 3: task-outcome recoverability, depth-wise behavior, and paired evidence under a fixed continuation policy. The admission decomposition defines the corresponding evaluation quantities, while deployment thresholds and gate behavior are treated separately.

## A.1 Formal Setup and Assumptions

Fix a reference time $\tau$, reference action $\bar{u}_\tau$, and successful reference rollout $\xi^\star$. Let $D_J$ be a non-negative discrepancy on task outcomes with tolerance $\varepsilon_J$, and let $\rho$ be the recovery-risk tolerance. Under a fixed execution rule, trajectories are defined on a filtered probability space

$$(\Omega, \mathcal{F}, \Pr), \qquad (\mathcal{F}_t)_{t\geq 0}.$$

Their law is induced by the initial-state distribution, robot dynamics, low-level controller, action-propagation mechanism, and future disturbances. We assume the conditional probabilities and sets below admit measurable versions.

Let $\Pi_c$ be the admissible causal continuation policies. For every $\pi \in \Pi_c$, the command selected at time $t$ is $\mathcal{F}_t$-measurable. Thus $\pi$ is a causal rule fixed before future disturbances are realized and cannot be chosen retrospectively for each disturbance sequence.

At depth $d$,

$$\Xi_d^{\mathrm{prop}}(s, \pi)$$

contains the realized trajectory prefix through propagation-boundary state $s$, one transition under $\mathcal{P}_d(\bar{u}_\tau)$, and the subsequent continuation under $\pi$. If $J$ is history dependent, either $s$ contains the task-relevant history or probabilities are conditioned on the fixed realized prefix.

The principal quantities are

$$\begin{aligned} q_d(s,\pi) &= \Pr[D_J(J(\Xi_d^{\mathrm{prop}}(s,\pi)), J^\star) \leq \varepsilon_J \mid S_{\tau+d} = s], \\ \mathcal{R}_d &= \{s : \exists \pi \in \Pi_c,\ q_d(s,\pi) \geq 1-\rho\}, \\ A_d &= \{S_{\tau+d} \in \mathcal{R}_d\}, \qquad p_d = \Pr(A_d). \end{aligned} \tag{A.1}$$

For fixed $s$ and $\pi$, $q_d(s, \pi)$ averages over future randomness after the boundary state. $\mathcal{R}_d$ collects states for which some admissible causal continuation reaches the reference outcome with probability at least $1 - \rho$. In contrast, $p_d$ averages membership in $\mathcal{R}_d$ over the state-visitation distribution at propagation depth $d$.

**Assumption 1** (Necessary recoverability margin)**.** *For each propagation depth $d$, there exists a measurable normalized deviation $e_d$ and a finite radius $r_d$ such that*

$$\mathcal{R}_d \subseteq \{s : e_d(s) \le r_d\}, \qquad r_{d+1} \le r_d. \tag{A.2}$$

*Writing*

$$E_d = e_d(S_{\tau+d}), \qquad M_d = \{E_d \le r_d\},$$

*Assumption 1 gives*

$$A_d \subseteq M_d.$$

*This margin is necessary, not sufficient: states outside $M_d$ are excluded by the abstraction, while membership in $M_d$ alone does not imply membership in $\mathcal{R}_d$.*

**Assumption 2** (Stochastic degradation across propagation depths)**.** *Let $E_{d+1} = e_{d+1}(S_{\tau+d+1})$ be the normalized deviation at the next propagation-boundary state, reached after one additional propagated transition and before any continuation-policy action. For each $d$, assume*

$$\Pr[E_{d+1} \le x] \le \Pr[E_d \le x], \qquad \forall x \ge 0. \tag{A.3}$$

*Thus an additional open-loop propagation step, before continuation feedback acts, does not stochastically improve the boundary-deviation distribution.*

### A.2 Proof of the Propagation-Depth Bound

Define

$$\epsilon_d^{\mathrm{abs}} = \Pr(M_d) - p_d = \Pr(M_d \setminus A_d). \tag{A.4}$$

**Theorem 1** (Recoverability across propagation depth)**.** *Under Assumptions 1 and 2,*

$$p_{d+1} \le p_d + \epsilon_d^{\mathrm{abs}}.$$

*Proof.* Assumption 1 gives $A_{d+1} \subseteq M_{d+1}$. Therefore,

$$\begin{aligned} p_{d+1} &\le \Pr(M_{d+1}) \\ &= \Pr[E_{d+1} \le r_{d+1}] \\ &\le \Pr[E_d \le r_{d+1}] \\ &\le \Pr[E_d \le r_d] \\ &= \Pr(M_d) \\ &= p_d + \epsilon_d^{\mathrm{abs}}. \end{aligned} \tag{A.5}$$

The first inequality uses the necessary margin, the second Assumption 2, and the third $r_{d+1} \le r_d$. □

The bound compares successive propagation-boundary states; $S_{\tau+d+1}$ is not a post-feedback state. Future feedback is represented in $\mathcal{R}_d$ through the causal continuation policy and is evaluated empirically in Appendix A.4.

The bound does not imply monotonic decrease of $p_d$. A sharper decomposition separates contraction of the necessary margin from changes in the abstraction gap. Define

$$\kappa_d = \Pr(M_d) - \Pr(M_{d+1}) \ge 0.$$

Nonnegativity follows from Assumption 2 and $r_{d+1} \le r_d$. Since $p_j = \Pr(M_j) - \epsilon_j^{\mathrm{abs}}$,

$$p_{d+1} = p_d - \kappa_d + \epsilon_d^{\mathrm{abs}} - \epsilon_{d+1}^{\mathrm{abs}}. \tag{A.6}$$

Consequently,

$$\kappa_d + \epsilon_{d+1}^{\mathrm{abs}} \ge \epsilon_d^{\mathrm{abs}} \quad \Longrightarrow \quad p_{d+1} \le p_d. \tag{A.7}$$

Thus $p_{d+1} \le p_d$ only under condition (A.7). Because the scalar margin only abstracts the recoverable set, this condition need not hold uniformly across depths.

### A.3 Observable Propagation Decisions

The latent event $A_d$ depends on future task outcomes and therefore cannot be observed at inference. Let $Z_d$ denote an observable signal and $\mathcal{B}_d$ the region in which propagation is admitted. Define the conditional recoverability function

$$\eta_d(z) = \Pr(A_d \mid Z_d = z)$$

and assume

$$\Pr(Z_d \in \mathcal{B}_d) > 0.$$

If the admitted region satisfies

$$\operatorname*{ess\,inf}_{z \in \mathcal{B}_d} \eta_d(z) \geq 1 - \delta_d, \tag{A.8}$$

then

$$\begin{aligned} \Pr(A_d \mid Z_d \in \mathcal{B}_d) &= \mathbb{E}\left[\eta_d(Z_d) \mid Z_d \in \mathcal{B}_d\right] \\ &\geq 1 - \delta_d. \end{aligned} \tag{A.9}$$

Equation (A.9) follows from conditional expectation and gives a sufficient distribution-level condition for an already specified admission set $\mathcal{B}_d$. It does not construct $\mathcal{B}_d$ or imply pointwise inclusion of admitted states in $\mathcal{R}_d$. The two probability levels are also distinct: $\rho$ concerns future randomness from a fixed state, whereas $\delta_d$ concerns recoverability over the visited-state distribution conditioned on admission. Accordingly, Equation (A.9) does not by itself calibrate the deployed visual–motion gate.

### A.4 Policy-Conditioned Paired Evidence

The existential set $\mathcal{R}_d$ cannot be exhaustively labeled experimentally because doing so would require searching over all policies in $\Pi_c$. The paired study therefore fixes the deployed VLA continuation policy

$$\pi_{\mathrm{VLA}} \in \Pi_c$$

and defines

$$\mathcal{R}_d^{\pi_{\mathrm{VLA}}} = \left\{ s : q_d(s, \pi_{\mathrm{VLA}}) \geq 1 - \rho \right\}. \tag{A.10}$$

Since $\pi_{\mathrm{VLA}}$ is one admissible causal continuation policy,

$$\mathcal{R}_d^{\pi_{\mathrm{VLA}}} \subseteq \mathcal{R}_d. \tag{A.11}$$

Recovery under $\pi_{\mathrm{VLA}}$ is thus conservative evidence for existential recoverability, whereas failure under $\pi_{\mathrm{VLA}}$ does not show that every admissible continuation would fail.

For each restored simulator checkpoint, the current-action branch first executes the VLA action computed from the current observation, while the propagation branch first executes $\mathcal{P}_d(\bar{u}_\tau)$. After these prescribed interventions, both branches use $\pi_{\mathrm{VLA}}$ and matched future randomness. Let $Y_d^{\mathrm{current}}$ and $Y_d^{\mathrm{prop}}$ be their binary task outcomes. Then

$$\begin{aligned} &\Pr[Y_d^{\mathrm{prop}} = 1] - \Pr\left[Y_d^{\mathrm{current}} = 1\right] \\ &= \Pr\left[Y_d^{\mathrm{current}} = 0, Y_d^{\mathrm{prop}} = 1\right] - \Pr\left[Y_d^{\mathrm{current}} = 1, Y_d^{\mathrm{prop}} = 0\right]. \end{aligned} \tag{A.12}$$

The first term is the positive-outcome-flip probability and the second the negative-outcome-flip probability. This identity is exact and does not use Assumptions 1 or 2.

Positive flips are cases in which the propagation branch succeeds while the current-action branch fails under the same deployed continuation policy and matched future randomness. They therefore measure a task-level consequence of the altered first transition together with subsequent feedback, rather than validating the propagation-boundary bound of Theorem 1.

The study estimates policy-conditioned recovery and outcome-flip frequencies from eight future paths per checkpoint. These quantities characterize the fixed continuation policy used in the experiment; they do not estimate the existential set mass $p_d$ or calibrate $\mathcal{B}_d$. We therefore treat them as policy-conditioned empirical evidence.

### A.5 Selective Single-Intervention Recovery

Fix a checkpoint distribution and the paired interventions of Appendix A.4. Let $G_d = \{Z_d \in \mathcal{B}_d\}$ be an admission event determined by information available before the intervention and future disturbances. A refresh-horizon check may be included in $G_d$. Both branches subsequently follow the same fixed causal continuation policy; only the first action is selected by $G_d$.

**Proposition A.1** (Admission and paired outcome changes)**.** *Define* $F_d^+ = \{Y_d^{\text{current}} = 0, Y_d^{\text{prop}} = 1\}$, $F_d^- = \{Y_d^{\text{current}} = 1, Y_d^{\text{prop}} = 0\}$ *and* $Y_d^G = \mathbf{1}_{G_d} Y_d^{\text{prop}} + \mathbf{1}_{G_d^c} Y_d^{\text{current}}$. *Then*

$$\Pr(Y_d^G = 1) - \Pr(Y_d^{\text{current}} = 1) = \Pr(G_d \cap F_d^+) - \Pr(G_d \cap F_d^-) \geq -\Pr(G_d \cap F_d^-). \quad \text{(A.13)}$$

*For* $c_d = \Pr(G_d) > 0$ *and* $h_d^{\pm} = \Pr(F_d^{\pm} \mid G_d)$*, the change is* $c_d(h_d^+ - h_d^-)$ *and its lower bound is* $-c_d h_d^-$*. When* $c_d = 0$*, both joint masses and the change are zero.*

*Proof.* Pointwise, $Y_d^G - Y_d^{\text{current}} = \mathbf{1}_{G_d}(\mathbf{1}_{F_d^+} - \mathbf{1}_{F_d^-})$. Taking expectations proves the identity; dropping the nonnegative positive-flip term proves the inequality. The conditional form follows by factoring each joint probability. No independence between admission and outcomes is assumed, and Assumptions 1 and 2 are not required. □

This decomposition specifies what a propagation gate should be evaluated against: admission coverage together with admitted negative and positive outcome flips. A larger admitted set need not improve recovery, even if its visual deviations are small. For a single intervention, the negative-flip term gives a conservative bound on recovery lost among admitted records. Its interpretation remains tied to the sampled checkpoint population and does not extend to repeated deployment. Appendix C.3.5 evaluates this decomposition on the archived screening mask and documents its differences from the deployed gate.

### A.6 State Deviation, Feedback Recovery, and Task Loss

The preceding outcome identities specify what to measure. We next give a conditional incremental-gain analysis of how propagation errors can persist or decay through physical transitions and feedback. The required transition- and feedback-gain assumptions are additional to Assumptions 1–2 and are not empirically asserted for the reported VLA deployments; in particular, they do not identify small visual deviation with task safety.

**Coupled trajectories and scope.** Fix a finite decision horizon $T$ and couple a propagation trajectory $X_t$ with an always-evaluated VLA trajectory $X_t^0$ using the same initial state and disturbance sequence. Write the physical transition as $F_t(x, u, w)$, allowing time dependence. The state includes any controller memory and task history needed to make this transition and the task outcome well defined; propagation bookkeeping remains part of the scheduling rule. The deterministic causal VLA control rule $\mu_t$ uses the same observation preprocessing and numerical execution in both branches; it is not an assumption of equality between separately implemented decoders. The reference branch always executes $\mu_t(X_t^0)$. At VLA steps the propagation branch executes $\mu_t(X_t)$; at propagation steps it executes its scaled stored command $U_t^{\text{prop}}$. Fix the state norm and Euclidean action coordinates, including physical-unit weights, before assessing gains. Let $D_t = \|X_t - X_t^0\|$. This physical-state separation is distinct from both visual $\Delta_t$ and the abstract $E_d$.

**Local transition and feedback assumptions.** On a specified domain containing the compared states and transitions, suppose finite constants $L_s, L_u, \beta \geq 0$ satisfy, for every relevant time, state pair, action pair, and shared disturbance,

$$\begin{aligned} \|F_t(x, u, w) - F_t(y, v, w)\| &\leq L_s\|x - y\| + L_u\|u - v\|, \\ \|F_t(x, \mu_t(x), w) - F_t(y, \mu_t(y), w)\| &\leq \beta\|x - y\|. \end{aligned} \quad \text{(A.14)}$$

The second inequality measures the complete closed-loop transition, not action similarity. Contraction requires $\beta < 1$ and is not presumed to hold globally: contact changes, discrete policy switches, and irreversible task events can invalidate it. Every pathwise conclusion below is restricted to paths remaining in the stated domain. A global claim would additionally require domain invariance or an explicit domain-exit term.

**Proposition A.2** (Propagation injection and feedback attenuation)**.** *Under Equation equation A.14, define the comparison residual at a propagation step by* $\zeta_t = \|U_t^{\text{prop}} - \mu_t(X_t^0)\|$*. Set* $(a_t, b_t) =$

*$(L_s, L_u\zeta_t)$ at propagation steps and $(a_t, b_t) = (\beta, 0)$ at VLA steps. For any realized causal schedule and $D_0 = 0$,*

$$D_T \le B_T := \sum_{t=0}^{T-1} b_t \prod_{k=t+1}^{T-1} a_k. \tag{A.15}$$

*Empty products equal one. In particular, a streak of $h \le H_{\mathrm{refresh}}$ propagated commands starting at time $s$, followed by $m$ consecutive VLA steps, satisfies*

$$D_{s+h+m} \le \beta^m \left( L_s^h D_s + L_u \sum_{j=0}^{h-1} L_s^{h-1-j} \zeta_{s+j} \right). \tag{A.16}$$

*Proof.* At a propagation step, the first gain inequality applied to $X_t, X_t^0$ and their executed actions gives $D_{t+1} \le L_s D_t + L_u \zeta_t$. At a VLA step both branches use their own feedback action, so the second inequality gives $D_{t+1} \le \beta D_t$. Iterating $D_{t+1} \le a_t D_t + b_t$ proves the first bound. Applying the propagation recurrence $h$ times and the feedback recurrence $m$ times proves the second. The argument is pathwise, so the schedule need not be independent of states or disturbances. □

**What the refresh boundary controls.** The horizon bounds the number of consecutive open-loop gain factors and residual injections, not their magnitudes. A refreshed reference does not reset $D_t$ to zero. For example, if $\zeta_t \le \bar{\zeta}$ uniformly and every streak is followed by at least one VLA step, define

$$A = \beta \max\{1, L_s^{H_{\mathrm{refresh}}}\}, \qquad C = \beta L_u \bar{\zeta} \sum_{j=0}^{H_{\mathrm{refresh}}-1} L_s^j. \tag{A.17}$$

At successive post-refresh states, the recurrence is $D^{(k+1)} \le AD^{(k)} + C$ (including a zero-length streak). When $A < 1$, iteration gives

$$D^{(k)} \le A^k D^{(0)} + C \frac{1 - A^k}{1 - A}. \tag{A.18}$$

Thus bounded streaks support a uniform residual bound only together with appropriate transition, residual, and feedback gains. A finite horizon alone is insufficient. These gains have not been measured here, so this condition neither certifies the deployed horizon nor selects its numerical value.

**What action scaling controls.** At a fixed comparison state, let $u$ be the stored six-dimensional continuous command and $v$ the reference VLA's current continuous command. In fixed Euclidean action coordinates, the continuous contribution to $\zeta_t$ is $\|\alpha_d u - v\|_2$, not $\alpha_d \|u\|_2$. Direct expansion gives

$$\|\alpha_d u - v\|_2^2 - \|u - v\|_2^2 = (1 - \alpha_d) \left[ 2\langle u, v \rangle - (1 + \alpha_d) \|u\|_2^2 \right]. \tag{A.19}$$

For $0 \le \alpha_d < 1$, scaling does not increase this local residual exactly when $2\langle u, v \rangle \le (1 + \alpha_d)\|u\|_2^2$, with equality giving no change. If $v = u \ne 0$, scaling instead introduces error. The unchanged gripper can still differ from the reference gripper and must be included in the full residual; the continuous calculation does not eliminate that contribution. This comparison fixes $u, v$ and does not compare the full state distributions of differently scaled rollouts. The rule $\alpha_d = 1/d$ therefore enters the bound explicitly, but is not universally preferable to unscaled propagation.

**From state separation to task loss.** If a terminal task cost $\ell$ is $L_\ell$-Lipschitz on the stated domain, then

$$|\ell(X_T) - \ell(X_T^0)| \le L_\ell B_T. \tag{A.20}$$

Binary success is not generally Lipschitz. A separate margin condition is needed: if success is exactly $g(X_T) \ge 0$ for a $K$-Lipschitz signed task margin $g$, a lost reference success requires $0 \le g(X_T^0) \le KB_T$. Let $\mathcal{E}$ denote that all required states and transitions stay in the domain. With $Y^0, Y$ the two terminal success indicators, this yields

$$\Pr(Y^0 = 1, Y = 0) \le \Pr\big(\mathcal{E} \cap \{0 \le g(X_T^0) \le KB_T\}\big) + \Pr(\mathcal{E}^c). \tag{A.21}$$

Here $B_T$ is the realized comparison envelope, not an independent random variable. The cost bound follows from Lipschitz continuity; the success bound follows by event inclusion and separating domain exits. Neither the task-margin representation nor its distribution is supplied by the visual gate. Without such a margin, the state bound must not be interpreted as a success-rate guarantee.

**Falsifiable checks and present evidence.** Testing these gain conditions would require matched nearby-state transition comparisons, policy-dependent feedback tests, executed-command residuals, and explicit task-margin accounting. Held-out pairs can expose violations but do not establish uniform gain bounds. The present single-intervention recovery study does not provide these multi-step measurements; this analysis instead identifies a conditional mechanism and the quantities that a dedicated stability study would need to measure.

# B Method and Implementation Details

## B.1 Complete Inference Procedure

**Reference persistence and refresh.** Propagation does not update $(o_\tau, \bar{u}_\tau)$, so every visual decision within a propagation streak compares the current observation with the same last-VLA reference. A streak terminates when the gate rejects the candidate propagation or when the next candidate depth would exceed the configured refresh horizon, after which a fresh VLA evaluation updates the reference pair. Thus, the gate may trigger an earlier refresh, while $H_{\text{refresh}}$ bounds the number of consecutive propagation steps.

## B.2 Parallel Suffix Proposal

The frozen VLA supplies both the context and supervision used by the proposal network. Training examples enumerate verified prefix positions so that the same network can be invoked after a mismatch at any action position.

For a prefix ending at position $a$, let $Z_{a,j}$ be the proposal representation for unresolved slot $j$, and let $G_{a,j}$ be the corresponding frozen VLA action representation. The training objective is

$$\mathcal{L}_{\text{proposal}} = \mathcal{L}_{\text{hid}} + 0.05\mathcal{L}_{\cos} + 0.05\mathcal{L}_{\text{soft}}, \tag{B.1}$$

where

$$\mathcal{L}_{\text{hid}} = \langle \text{SmoothL1}\left(Z_{a,j}, G_{a,j}\right)\rangle_w, \tag{B.2}$$

$$\mathcal{L}_{\cos} = \langle 1 - \cos\left(Z_{a,j}, G_{a,j}\right)\rangle_w, \tag{B.3}$$

and

$$\begin{aligned} \mathcal{L}_{\text{soft}} &= \left\langle T^2 D_{\text{KL}}\left(p^{(T)}_{\text{VLA},a,j} \| p^{(T)}_{\phi,a,j}\right)\right\rangle_w, \\ w_{a,j} &= \frac{0.9^j}{a+j+1}, \qquad T = 2. \end{aligned} \tag{B.4}$$

The brackets denote averaging with weights $w_{a,j}$. The hidden-state and cosine losses align proposal representations with the frozen VLA. The softened distribution loss distills the VLA's preferences over action tokens. Earlier unresolved positions receive greater weight because an early mismatch prevents later proposals from being accepted in the same round.

An autoregressive proposal model constructs each suffix token from its preceding proposal outputs:

$$\hat{a}_{a+j} = D_{\text{AR}}\left(o_t, g, \mathbf{A}_t^{1:a}, \hat{\mathbf{A}}_{a+1:a+j-1}\right). \tag{B.5}$$

Parallel Suffix Proposal instead evaluates all unresolved slots in one noncausal layer. Action-dimension embeddings distinguish the seven output positions without imposing a proposal-to-proposal causal order. Exact VLA verification then accepts only

$$r_a = \max\left\{r : \hat{a}_{a+j} = v_{a,j} \text{ for all } 1 \le j \le r\right\}. \tag{B.6}$$

A later match can therefore never bypass an earlier mismatch. When $r_a = 0$, the VLA correction token still advances decoding by one position.

## B.3 Architecture, Training, and Overhead

The proposal network contains one Transformer layer (Vaswani et al., 2017) over the action slots, a projection that fuses five frozen VLA layers, one embedding for each action dimension, and an output projection restricted to the 256-token action vocabulary. The selected VLA layers are

$$\mathcal{S} = \{1, 9, 16, 24, 31\}. \tag{B.7}$$

**Algorithm B.1** Spec$^2$ inference

**Require:** Observation $o_t$, instruction $g$, previous verified sequence $\bar{\mathbf{A}}$, reference pair $(o_\tau, \bar{u}_\tau)$, current propagation depth $d$, motion mode state, and refresh horizon $H_{\text{refresh}}$

**if** no previous verified action sequence exists **then**
  evaluate the frozen VLA on $(o_t, g)$
  initialize $\mathbf{A}_t$ with its first greedy VLA action token
**else**
  set the candidate propagation depth to $d_{\text{next}} \leftarrow d + 1$
  **if** $d_{\text{next}} \leq H_{\text{refresh}}$ **then**
    compute $\Delta_t$ from $o_t$ and $o_\tau$
    evaluate $\mathcal{I}_{\text{prop}}(t, d_{\text{next}})$ using Equation (9)
    **if** $\mathcal{I}_{\text{prop}}(t, d_{\text{next}}) = 1$ **then**
      execute $u_t^{\text{exec}} \leftarrow \mathcal{P}_{d_{\text{next}}}(\bar{u}_\tau)$
      keep $\bar{\mathbf{A}}$ and $(o_\tau, \bar{u}_\tau)$ unchanged
      set $d \leftarrow d_{\text{next}}$ and **return**
    **end if**
  **end if**
  evaluate the frozen VLA on $(o_t, g)$
  apply Previous-Action Verification
  **if** the sequence matches completely **then**
    set $\mathbf{A}_t \leftarrow \bar{\mathbf{A}}$
  **else**
    retain the verified prefix and the first VLA correction token
  **end if**
**end if**
**while** $\mathbf{A}_t$ is incomplete **do**
  jointly propose every unresolved position with Parallel Suffix Proposal
  run causal VLA verification
  accept the longest consecutive matching proposal prefix
  **if** the action remains incomplete **then**
    append the next VLA correction token
  **end if**
**end while**
detokenize $\mathbf{A}_t$ to obtain $u_t$, and execute $u_t$
**if** no previous verified action sequence existed **then**
  initialize the motion-mode consistency streak to one, with $u_t$ as its first VLA action
**else**
  update motion mode consistency using the previous and current VLA actions
**end if**
$\bar{\mathbf{A}} \leftarrow \mathbf{A}_t$, $(o_\tau, \bar{u}_\tau) \leftarrow (o_t, u_t)$, and $d \leftarrow 0$

**Online slot construction and alignment.** For a verified prefix ending at action position $a$, the current causal VLA prefill/verification pass supplies an $H$-dimensional hidden state at every prompt and verified action-prefix position, including the anchor. The implementation selects `hidden_states[`$\ell + 1$`]` for each $\ell \in \mathcal{S}$ (the zeroth entry is the embedding output), concatenates the five states feature-wise at each retained position, and applies $W_f \in \mathbb{R}^{H \times 5H}$ followed by RMSNorm. For OpenVLA-7B, $H = 4096$, so this is a $20{,}480 \to 4{,}096$ projection. The $K - a$ inputs in $\mathbf{X}_a$ comprise the verified-anchor token embedding followed by $K - a - 1$ embeddings of one fixed mask token; slot-specific action-dimension embeddings index their action positions. The one-layer proposal Transformer uses noncausal attention, so every proposal slot attends to all context positions and all proposal slots. Its $K - a$ outputs are mapped through the frozen VLA language-model head restricted to its 256 action-token rows, yielding all unresolved proposals in one round. The selected features are already emitted by the causal VLA prefill/verification pass, so constructing $C_a$ requires no additional VLA forward pass; exact causal verification remains unchanged.

Each LIBERO suite uses one suite-specific proposal checkpoint with the same frozen OpenVLA 7B architecture. Training supervision is extracted from the corresponding LIBERO RLDS subset (Ramos et al., 2021) after removing no-operation transitions. The VLA supplies action tokens, final action representations, and the selected intermediate features. Simulator outcomes and action-propagation trajectories are not used for proposal training.

The proposal network is optimized with AdamW (Loshchilov & Hutter, 2019); the complete configuration is given below.

All VLA parameters remain frozen. Proposal training changes candidate quality but cannot alter the token selected by the VLA verifier.

The BF16 proposal checkpoint contains 286,306,304 parameters and occupies approximately 0.57 GB. Only the checkpoint for the active suite is loaded.

The reported 0.57 GB value refers to stored BF16 checkpoint weights.

### B.4 Greedy Equivalence

**Proposition B.1.** *Assume identical VLA parameters, positional encodings, causal masks, preprocessing, and exact numerical execution. Verified Action Block Decoding returns the same action sequence as serial greedy VLA decoding.*

*Proof.* During Previous-Action Verification, the prediction at position $j$, before the first mismatch, is conditioned on

$$\left[o_t, g, \bar{\mathbf{A}}_\tau^{1:j-1}\right].$$

Every preceding historical token has matched the VLA prediction, so this is the same prefix used by serial greedy decoding. The logits and selected token are therefore identical. Induction over $j$ shows that Equation (5) returns either the complete greedy sequence or its longest matching prefix followed by the first greedy correction token.

Now consider a Parallel Suffix Proposal round beginning from a verified prefix. At every accepted proposal position, all preceding proposals have matched the VLA, so the causal context again equals that of serial greedy decoding. Exact prefix verification stops at the first mismatch and appends the VLA token at that position. The extended prefix therefore remains identical to the serial greedy prefix. Repeating this argument until all $K$ positions are complete proves the claim. □

The proposition assumes exact arithmetic. In BF16 execution, different matrix shapes can produce small rounding differences when the two largest logits are nearly tied. The online VLA verifier therefore defines the deployed exact-match semantics: a proposal is accepted only when it matches the token returned by the actual verification path.

### B.5 Visual and Motion Mode Consistency

**Visual signature.** The LIBERO implementation applies adaptive average pooling to the 6-channel fused preprocessed image tensor, preserving channels to produce a $6 \times 16 \times 16$ tensor that is flattened to a 1,536-dimensional $\phi(o)$. For LIBERO, the normalized reference-relative visual gate uses $\delta_{\text{low}} = 0.075$ and $\delta_{\text{high}} = 0.15$; the normalization floor in Equation (8) is $\epsilon = 10^{-6}$, distinct from the motion threshold $\epsilon_u$ below. These thresholds are fixed operating points on the normalized reference-relative change in Equation (8), rather than thresholds on absolute scene appearance. Pixel pooling is deliberately used as a low-cost temporal consistency signal because the gate is evaluated on every control step, including steps on which VLA inference may be skipped; introducing a separate learned visual encoder would add neural computation to this path. The same $(\delta_{\text{low}}, \delta_{\text{high}})$ pair is used across all four LIBERO suites without online or suite-specific adaptation. Its role is to determine whether current visual evidence supports propagation, not to replace semantic action reasoning by the VLA. Continued propagation in the outer visual region is additionally conditioned on VLA-derived motion-mode consistency, while rejection of either gate returns control to a fresh VLA evaluation.

**Motion mode.** Let $a$ and $b$ be two consecutive actions produced at VLA steps. For translation or rotation group $G$, define

$$c_G(a, b) = \mathbb{I}\left[\|a_G\|_2 \le \epsilon_u \ \vee \ \|b_G\|_2 \le \epsilon_u \ \vee \ a_G^\top b_G \ge 0\right]. \tag{B.8}$$

The first two conditions treat near-stationary vectors as compatible. For LIBERO, we use $\epsilon_u = 0.002$ for both the translation and rotation groups. The gripper-state threshold is $0.5$. Thus, sufficiently

Table B.1: Proposal-network training configuration.

| Setting | Value |
|---|---|
| Optimizer | AdamW |
| Learning rate | $2 \times 10^{-5}$ |
| $\beta_1, \beta_2$ | $0.9, 0.98$ |
| Weight decay | $0.05$ |
| Gradient clipping | $0.5$ |
| Warm-up steps | 1,000 |
| Batch size | 32 |
| Gradient accumulation | None |
| Hidden-state noise | Gaussian, standard deviation $0.03$ |
| Selected checkpoint | 100 epochs |

Table B.2: Additional state used by Spec$^2$.

| Component | Learned parameters | Stored weights |
|---|---|---|
| Previous-Action Verification | 0 | 0 |
| Parallel Suffix Proposal | 286,306,304 | 0.57 GB |
| Visuomotor-Gated Action Propagation | 0 | 0 |

small action vectors are treated as compatible irrespective of their numerically unstable direction, whereas non-stationary vectors are required to have a nonnegative inner product. Gripper consistency is

$$c_g(a, b) = \mathbb{I}\left[\mathbb{I}(g_a \geq 0.5) = \mathbb{I}(g_b \geq 0.5)\right], \tag{B.9}$$

and the complete motion mode comparison is

$$c(a, b) = c_{\mathrm{tr}}(a, b) c_{\mathrm{rot}}(a, b) c_g(a, b). \tag{B.10}$$

Let $\nu_j$ index VLA steps. The consistency streak is

$$L_j = \begin{cases} L_{j-1} + 1, & c(\bar{u}_{\nu_{j-1}}, \bar{u}_{\nu_j}) = 1, \\ 1, & \text{otherwise}, \end{cases} \qquad c_j^{\mathrm{mode}} = \mathbb{I}[L_j \geq 2]. \tag{B.11}$$

Only actions produced at VLA steps update $L_j$. Propagated commands are excluded because they do not reveal what the VLA would have produced from the current observation.

**Action scaling.** The depth-indexed propagation rule uses

$$\alpha_d = \frac{1}{d} \tag{B.12}$$

for the six continuous action dimensions. Under the deployed refresh horizon, the first propagated command retains full magnitude and the second uses half; deeper recovery interventions apply the same $1/d$ rule. The gripper remains unchanged because it is an absolute open/close command rather than a displacement, with state consistency checked by Equation B.9.

**Refresh horizon.** The LIBERO deployment uses $H_{\mathrm{refresh}} = 2$: the gate may refresh the VLA earlier, and no streak contains more than two propagated commands. This setting was motivated by the depth-wise paired diagnostic in Table 5, where depth 2 has the highest propagated-action recovery among four evaluated intervention depths. This local recovery diagnostic is not an end-to-end horizon sweep and does not establish a globally optimal $H_{\mathrm{refresh}}$.

**LIBERO propagation configuration.** All four LIBERO suites share the same deployment configuration: $\delta_{\mathrm{low}} = 0.075$, $\delta_{\mathrm{high}} = 0.15$, $\epsilon_u = 0.002$, $\alpha_1 = 1$, $\alpha_2 = 0.5$, and $H_{\mathrm{refresh}} = 2$. These settings remain unchanged throughout evaluation, without online adaptation or suite-specific tuning. Appendix C.4 reports local sensitivity of the gate and scaling parameters.

# C Additional Experimental Details and Results

This appendix provides the evaluation protocols and supplementary results supporting Section 5. It covers the metric scope for LIBERO, SimplerEnv, and Alicia-D and provides additional analyses of Verified Action Block Decoding and Action Propagation. It also documents the paired closed-loop evaluation, the CogACT portability study, and the real-robot data, training, and deployment protocol.

## C.1 Evaluation Protocol and Metric Scope

**LIBERO.** The primary simulation evaluation uses frozen OpenVLA-7B policies on LIBERO Goal, Spatial, Object, and Long. Each suite contains ten tasks with 50 episodes per task, giving 500 episodes per suite and 2,000 episodes for Spec$^2$. All locally evaluated methods use the same policy weights, image preprocessing, task order, initial-state sequence, and random seed.

Success rate is computed over all 500 episodes in each suite. Action-inference latency is measured under a fixed policy-path timing protocol on the same RTX 4090 environment. For each LIBERO suite, timing is collected on one fixed evaluation task shared by all locally evaluated methods. The timer starts when the policy path receives the processed observation and ends when the action is returned; simulator stepping is excluded.

For Spec$^2$, the reported latency averages both VLA steps and propagation steps. Speedup is normalized by the autoregressive OpenVLA latency measured within the same suite. These measurements characterize policy-path inference latency rather than end-to-end robot-control time. Results transcribed from prior work retain the protocols of their source papers and are not treated as hardware-matched latency measurements. Accordingly, we do not construct a universal Pareto frontier by mixing published latency numbers from systems that use different action interfaces, checkpoints, batching strategies, or inference runtimes.

**SimplerEnv.** The CogACT portability study is evaluated under a separately fixed SimplerEnv protocol. Verified Action Block Decoding is not applied because CogACT outputs continuous actions. Only Action Propagation is evaluated for cross-policy portability.

The local collection contains 96 WindowX episodes, 864 Google Visual Matching episodes, and 1,992 Google Visual Aggregation episodes. Success is macro-averaged across the four tasks of each setting. Decision latency and speedup are compared only against CogACT measurements collected in the same local environment. Published CogACT results are retained separately as contextual references and are not treated as paired or hardware-matched measurements.

**Alicia-D.** The real-robot evaluation contains three language-conditioned manipulation tasks, with 50 trials per method and task, giving 300 physical trials in total. The Conveyor-Belt Pick-and-Place task uses a belt speed of 3 m/min during both data collection and policy evaluation. Data construction, policy training, deployment, success evaluation, and timing are detailed in Section C.6. Results from LIBERO, SimplerEnv, and Alicia-D are reported independently. No cross-environment latency measurement is presented as hardware matched.

## C.2 Additional LIBERO Results

### C.2.1 Parallel Suffix Proposal

To isolate Parallel Suffix Proposal, Previous-Action Verification is disabled and the resulting decoder is compared with the autoregressive proposal mechanism of Spec-VLA. Both conditions use the same frozen OpenVLA policies, exact token-verification rule, evaluation tasks, initial states, random seed, and timing scope.

Parallel Suffix Proposal increases the mean number of accepted candidate tokens from 0.64 to 1.19 per proposal–verification round and the verified token advance from 1.64 to 2.25. The pooled number of proposal–verification rounds decreases from 202,399 to 147,435, a reduction of 27.2%. The parallel condition advances 330,519 verified action tokens in aggregate, compared with 326,648 for Spec-VLA, so the lower round count is not explained by a smaller decoded workload. Mean policy-

Table C.1: Summary of evaluation scope.

| Environment | Policy | Evaluation size | Success metric | Latency scope |
|---|---|---|---|---|
| LIBERO | OpenVLA 7B | 4 suites, 2,000 episodes | Episode SR over all tasks | Fixed local policy path on RTX 4090 |
| SimplerEnv | CogACT | 2,952 episodes | Four-task macro SR | Same-environment decision latency |
| Alicia-D | Task-specific OpenVLA | 3 tasks, 300 physical trials | Trial-level task success | RTX 4090 server-side model path |

Table C.2: Suite-level analysis of Parallel Suffix Proposal under exact VLA verification. "Accepted" is the mean number of proposal tokens accepted per proposal–verification round. "Advance" is the total number of verified action tokens added per round.

| Suite | Accepted, Spec-VLA | Accepted, parallel | Advance, Spec-VLA | Advance, parallel | Rounds, Spec-VLA | Rounds, parallel | Parallel-only speedup |
|---|---|---|---|---|---|---|---|
| Goal | 0.63 | 1.11 | 1.63 | 2.17 | 43,825 | 33,351 | 1.14× |
| Spatial | 0.58 | 1.12 | 1.58 | 2.17 | 34,390 | 25,504 | 1.15× |
| Object | 0.81 | 1.34 | 1.81 | 2.41 | 27,496 | 20,398 | 1.18× |
| Long | 0.56 | 1.19 | 1.56 | 2.25 | 96,688 | 68,182 | 1.18× |
| Mean / pooled | 0.64 | 1.19 | 1.64 | 2.25 | 202,399 | 147,435 | 1.16× |

path latency decreases from 178.5 ms to 157.6 ms. Adding Previous-Action Verification gives the complete Verified Action Block Decoding result reported in Table 1.

Per-position proposal accuracy is not used as the primary comparison because the autoregressive and parallel proposal mechanisms induce different conditional prediction problems. A later Spec-VLA proposal depends on preceding proposal outputs, whereas Parallel Suffix Proposal predicts all unresolved positions jointly. Verified token advance under the same causal VLA verifier provides a directly comparable decoding-efficiency measure.

### C.2.2 Policy-Path Latency

Spec$^2$ contains two execution paths. VLA steps retain the multimodal inference path, whereas propagation steps require only the propagation decision and deterministic action transformation. Table C.3 reports the resulting mean policy-path latency under the fixed local timing convention defined in Section C.1; it should not be interpreted as device-independent wall-clock latency.

### C.2.3 Per-Task Success Rates

Autoregressive OpenVLA and Spec$^2$ use the same random seed and the first 50 fixed LIBERO initial states for every task. Spec$^2$ logs record the initial-state index explicitly. The earlier OpenVLA evaluation follows the same deterministic state order but does not redundantly store that index. We therefore describe this comparison as a matched-index audit, rather than as a randomized paired experiment.

### C.2.4 Descriptive Uncertainty and Matched-Index Outcomes

Table C.5 reports Wilson intervals (Wilson, 1927) for the Spec$^2$ success rate and task-cluster-resampled intervals for the matched-index difference from autoregressive OpenVLA. The latter are descriptive uncertainty intervals rather than equivalence or noninferiority tests. One-sided outcomes are expected because a change in an executed action alters the subsequent state and hence later observations. Goal and Long have nearly balanced one-sided outcomes, whereas Spatial exhibits the clearest negative imbalance. The suite-level pattern therefore supports a heterogeneous effect across task families rather than equality of the two controllers.

Table C.3: Policy-path action-inference latency for Spec$^2$.

| Suite | Spec$^2$ mean | AR mean | Speedup |
|---|---|---|---|
| Goal | 57.79 ms | 182.72 ms | 3.16× |
| Spatial | 80.01 ms | 183.46 ms | 2.29× |
| Object | 79.68 ms | 183.30 ms | 2.30× |
| Long | 60.97 ms | 183.41 ms | 3.01× |

Table C.4: Spec$^2$ success rate by LIBERO task. Each cell contains 50 episodes.

| Suite | Task 0 | Task 1 | Task 2 | Task 3 | Task 4 | Task 5 | Task 6 | Task 7 | Task 8 | Task 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Goal | 0.56 | 0.86 | 0.84 | 0.32 | 0.92 | 0.80 | 0.78 | 0.96 | 0.74 | 0.66 |
| Spatial | 0.88 | 0.82 | 0.94 | 0.92 | 0.70 | 0.84 | 0.90 | 0.90 | 0.76 | 0.62 |
| Object | 0.88 | 0.80 | 0.90 | 0.62 | 0.86 | 0.80 | 0.90 | 0.94 | 0.86 | 0.94 |
| Long | 0.60 | 0.76 | 0.58 | 0.46 | 0.32 | 0.72 | 0.52 | 0.64 | 0.24 | 0.42 |

Table C.5: Descriptive success intervals and matched-index outcome partitions.

| Suite | Spec$^2$ SR interval | Spec$^2$ minus AR interval | Both succeed | AR only | Spec$^2$ only | Both fail |
|---|---|---|---|---|---|---|
| Goal | [70.4, 78.0] | $[-6.2, +6.6]$ pp | 297 | 74 | 75 | 54 |
| Spatial | [79.2, 85.9] | $[-8.8, +0.2]$ pp | 379 | 56 | 35 | 30 |
| Object | [81.6, 87.9] | $[-8.4, +1.4]$ pp | 381 | 61 | 44 | 14 |
| Long | [48.2, 56.9] | $[-6.2, +8.4]$ pp | 164 | 93 | 99 | 144 |

### C.2.5 VLA and Propagation Steps

Every Spec$^2$ control step belongs to exactly one execution mode, and initial plus continued propagation equals total propagation. Across all 2,000 LIBERO episodes, 222,919 of 449,361 control steps invoke the VLA (49.61%); the complementary action-propagation rate is 50.39%.

Suite-level VLA inference rates span 46.97%–54.38%, so the reduction is not confined to one task family. Successful and failed episodes have propagation rates of 46.84% and 54.76%, respectively. Because outcome affects trajectory length and visited states, this success-conditioned difference is post-treatment and not interpreted causally.

Table C.6: Execution counts, execution-mode rates, and measured speedup reconstructed from all 2,000 Spec$^2$ episodes.

| Suite | Control steps | VLA steps | Propagation steps | Initial propagation | Continued propagation |
|---|---|---|---|---|---|
| Goal | 86,384 | 40,578 | 45,806 | 30,704 | 15,102 |
| Spatial | 69,479 | 37,785 | 31,694 | 21,484 | 10,210 |
| Object | 89,370 | 45,269 | 44,101 | 29,779 | 14,322 |
| Long | 204,128 | 99,287 | 104,841 | 70,610 | 34,231 |
| All | 449,361 | 222,919 | 226,442 | 152,577 | 73,865 |

| Suite | VLA inference rate | Action-propagation rate | Continued / propagation | Speedup |
|---|---|---|---|---|
| Goal | 46.97% | 53.03% | 32.97% | 3.16× |
| Spatial | 54.38% | 45.62% | 32.21% | 2.29× |
| Object | 50.65% | 49.35% | 32.48% | 2.30× |
| Long | 48.64% | 51.36% | 32.65% | 3.01× |
| All | 49.61% | 50.39% | 32.62% | — |

### C.2.6 Previous-Action Mismatch Regimes

To characterize VABD when previous-action reuse is weak, we examine the first mismatch position in the saved Previous-Action Verification records. For a seven-token action, we define $\kappa \in \{1, \ldots, 7\}$ as the first mismatching position and $\kappa = 8$ as a complete match. Across 24,055 logged VLA actions, only 15.38% completely match the previous action, whereas 69.17% mismatch within the first two positions.

As shown in Table C.7, the mismatch distribution is strongly skewed toward early positions. Spatial and Object have early-mismatch rates of 83.80% and 78.20%, yet retain measured VABD speedups of 1.17× and 1.19×. Thus, the measured suite-level acceleration persists despite frequent early mismatches. Because the retained logs do not contain comparable decoder timings indexed by $\kappa$, this analysis supports a suite-level rather than a per-action latency conclusion.

## C.3 Action Propagation Ablations and Closed-Loop Recovery

The component ablations use matched suite, task, and initial-state indices. The closed-loop recovery study follows the policy-conditioned intervention defined in Appendix A.4: from each restored simulator checkpoint, the current-action branch executes the action computed from the current observation, whereas the propagation branch executes the prescribed propagated reference action. After these first actions, both branches use the same deployed VLA continuation policy and matched future randomness.

For the recovery study, each depth contains 66 restored checkpoints, with eight matched future paths evaluated per checkpoint (528 future paths per depth). Uncertainty is summarized with descriptive 95% checkpoint-cluster bootstrap intervals that resample restored checkpoints within the original sampling strata while retaining all eight paired future paths within each checkpoint. Here, depth indexes a prescribed intervention rather than an observed deployment streak: depths 1–2 lie within the deployed refresh horizon, whereas depths 3–4 are counterfactual diagnostics beyond it. These measurements estimate policy-conditioned recovery and outcome-flip frequencies. They neither search over all causal continuation policies nor calibrate the observable propagation rule.

Pooling depths 1–2 by base anchor gives 94.60 [92.05, 96.97]% current-action recovery, 93.94 [90.81, 96.59]% propagated-action recovery, a $-0.66$ [$-2.84$, 1.42] pp difference, 4.45 [2.46, 6.82]% negative flips, and 3.79 [2.08, 5.78]% positive flips.

### C.3.1 Motion Mode Consistency

Motion mode consistency compares translation direction, rotation direction, and gripper state across actions obtained at VLA steps. The alternative requires exact equality of consecutive VLA actions as the cross-step propagation gate. Both conditions retain Verified Action Block Decoding with exact token verification at every VLA step.

Motion mode consistency provides the stronger aggregate accuracy–efficiency operating point: 73.7% SR at $2.69\times$ speedup, compared with 73.4% at $2.42\times$ under exact action equality. The effect is heterogeneous across suites, with exact equality performing better on Goal and Object and motion mode consistency performing better on Spatial and Long. The comparison motivates the selected aggregate configuration but does not establish task-wise dominance.

### C.3.2 Component Ablations

The main text reports the aggregate success and speed results of the Action Propagation ablations. Table C.10 gives the complete suite-level values, and Table C.11 gives matched outcome partitions relative to the 400-episode Spec$^2$ rerun.

The initial-only condition permits the first propagation step after a VLA action but forces a VLA refresh when the full rule would consider continued propagation. Its SR decrease is confined to Goal and Long, while Spatial and Object remain unchanged. Under the full rule, a second propagation remains subject to visual admission, requires motion mode consistency in the outer visual region, and scales the continuous command by $\alpha_2 = 1/2$. It therefore acts as a gated, lower-magnitude continuation of a coherent motion, whereas a forced refresh can change the VLA action mode.

Table C.7: Previous-action mismatch statistics for VABD on LIBERO. Early mismatch denotes $\kappa \in \{1, 2\}$, and full match denotes $\kappa = 8$. The Overall row pools mismatch statistics across records; its speedup is the four-suite mean reported in Table 1.

| Suite | Records | Full (%) | Early mismatch (%) | Mean $\kappa$ | Speedup |
|---|---|---|---|---|---|
| Goal | 4,722 | 21.60 | 62.81 | 3.18 | 1.30× |
| Spatial | 4,359 | 5.28 | 83.80 | 1.81 | 1.17× |
| Object | 4,256 | 8.88 | 78.20 | 2.14 | 1.19× |
| Long | 10,718 | 19.33 | 62.43 | 3.12 | 1.29× |
| Overall | 24,055 | 15.38 | 69.17 | 2.72 | 1.24× |

Table C.8: Descriptive 95% checkpoint-cluster bootstrap intervals for the depth-wise paired recovery diagnostic. Each depth contains 66 restored checkpoints with eight matched futures per checkpoint; resampling preserves the original suite/task strata and all within-checkpoint pairings. Entries are percentages: point estimate [interval].

| Depth | Current-action recovery | Propagated-action recovery | Difference | Negative flips | Positive flips |
|---|---|---|---|---|---|
| 1 | 94.70 [91.48, 97.35] | 92.42 [88.26, 95.83] | −2.27 [−6.06, 1.52] | 5.68 [2.84, 9.09] | 3.41 [1.33, 6.06] |
| 2 | 94.51 [91.48, 97.16] | 95.45 [92.99, 97.73] | 0.95 [−0.95, 2.84] | 3.22 [1.52, 5.30] | 4.17 [2.08, 6.63] |
| 3 | 93.56 [90.15, 96.59] | 94.32 [91.29, 96.97] | 0.76 [−1.52, 3.03] | 4.36 [2.27, 6.82] | 5.11 [3.03, 7.39] |
| 4 | 93.56 [90.53, 96.40] | 94.13 [91.48, 96.40] | 0.57 [−1.52, 2.84] | 4.36 [2.65, 6.25] | 4.92 [2.84, 7.20] |

Table C.9: Exact action equality versus motion mode consistency as propagation gates. Each entry reports SR (%)/speedup.

| Suite | Exact action equality | Motion mode consistency |
|---|---|---|
| Goal | 79.2 / 2.88× | 74.4 / 3.16× |
| Spatial | 78.8 / 2.09× | 82.8 / 2.29× |
| Object | 86.6 / 2.02× | 85.0 / 2.30× |
| Long | 49.0 / 2.69× | 52.6 / 3.01× |
| Average | 73.4 / 2.42× | 73.7 / 2.69× |

Table C.10: Complete suite-level ablation of the action-propagation rule. Each suite entry reports SR (%)/policy-path speedup over 100 episodes per suite.

| Condition | Goal | Spatial | Object | Long | Avg. SR | Avg. speedup |
|---|---|---|---|---|---|---|
| **Spec$^2$** | 76.0/3.08 | 80.0/2.27 | 89.0/2.27 | 55.0/2.69 | 75.0 | 2.58 |
| Initial propagation only | 72.0/2.59 | 80.0/2.01 | 89.0/2.02 | 46.0/2.60 | 71.8 | 2.31 |
| w/o motion mode consistency | 77.0/3.10 | 84.0/2.28 | 85.0/2.38 | 48.0/3.00 | 73.5 | 2.69 |
| w/o depth-dependent action scaling | 69.0/3.15 | 81.0/2.36 | 82.0/2.36 | 46.0/2.64 | 69.5 | 2.63 |
| w/o visual consistency boundary | 71.0/3.32 | 79.0/3.00 | 84.0/3.14 | 39.0/3.50 | 68.2 | 3.24 |

Table C.11: Matched outcomes against the 400-episode Spec$^2$ rerun. Intervals use task-cluster resampling and are descriptive.

| Variant | Both succeed | Spec$^2$ only | Variant only | Both fail | Variant–Spec$^2$ SR difference |
|---|---|---|---|---|---|
| Initial propagation only | 240 | 60 | 47 | 53 | −3.2 [−9.0, +2.0] pp |
| w/o motion mode consistency | 276 | 24 | 18 | 82 | −1.5 [−5.2, +2.0] pp |
| w/o depth-dependent action scaling | 232 | 68 | 46 | 54 | −5.5 [−12.5, +1.0] pp |
| w/o visual consistency boundary | 240 | 60 | 33 | 67 | −6.8 [−11.8, −2.0] pp |

Consistently, the depth-two paired result in Table 5 shows slightly higher propagated-action than current-action recovery and includes positive outcome flips. Nevertheless, the paired interval for the initial-only difference in Table C.11 includes zero. We therefore treat this as a plausible closed-loop explanation rather than evidence that continued propagation universally improves success.

The w/o visual consistency boundary condition effectively removes the usual visual-admission restriction over the observed evaluation range. As a consequence, states that would normally require additional motion-mode support can be admitted directly. This condition should therefore be interpreted as a joint relaxation of the propagation decision rather than as an isolated ablation of the visual descriptor.

Only the visual-boundary comparison has a descriptive interval excluding zero. The other intervals overlap zero, so we use these ablations to characterize component behavior rather than to rank the components universally.

### C.3.3 Depth-Dependent Action Scaling

To isolate depth-dependent scaling, the recovery intervention repeats the same reference action without scaling its continuous magnitude at later depths. Depths beyond $H_{\mathrm{refresh}} = 2$ are diagnostic rather than deployed streaks. Depth 1 is omitted from Table C.12 because $\alpha_1 = 1$, so scaled and unscaled commands coincide at the first propagation step.

The descriptive 95% checkpoint-cluster intervals for the scaled-minus-unscaled recovery differences are $+3.03$ [0.76, 5.49] pp at depth 2, $+0.95$ [$-1.89$, 3.79] pp at depth 3, and $+1.14$ [$-1.70$, 3.98] pp at depth 4.

Depth-dependent scaling yields higher measured recovery at every evaluated depth beyond the first, with the largest difference at depth 2, supporting the deployed scaling choice within the evaluated depths.

### C.3.4 Recovery Across Suites

Table C.13 aggregates the policy-conditioned paired recovery measurements by LIBERO suite across the evaluated propagation depths. Object and the completed Long subset contain more positive than negative outcome flips, whereas Goal and Spatial show the opposite ordering. The suite-level effect of Action Propagation is therefore heterogeneous.

### C.3.5 Diagnostic by Observable Region

For propagation depths 1 and 2, checkpoints are partitioned using the archived screening labels. Their visual thresholds match deployment, but the screening motion flag is generated from reference-step count (`target_run_length` $\geq 2$), not the deployed direction/gripper consistency test. Thus the outer-region label below denotes the archived flag, not verified deployment admission. This analysis is descriptive.

The outer visual region with the screening flag has the highest observed propagated-action recovery and the lowest negative-outcome-flip rate, but it contains only 19 checkpoints. States beyond the visual consistency boundary also exhibit positive outcome flips. The resulting region ordering is therefore descriptive rather than a calibrated recovery score.

**Admission-weighted paired outcomes.** Proposition A.1 can be evaluated by selecting the propagated outcome on admitted records and the current-action outcome otherwise. For the archived screening mask, depths 1–2 contain 112 admitted checkpoint-depth records out of 132 (84.85%). The admitted negative and positive flip masses are 4.17% and 2.94% of all paired futures, respectively. Selected recovery is 93.37%, versus 94.60% for the current-action branch, a $-1.23$ pp difference with a descriptive 95% checkpoint-cluster bootstrap interval of [$-3.12$, 0.38] pp. The corresponding single-intervention lower bound is 90.44% (94.60% minus 4.17%, computed before rounding). Bootstrap resampling retains both depths and all futures for each of the 66 base anchors, within the original suite/task strata (10,000 replicates; seed 20260824). These are post-hoc summaries of the archived screening sample, not additional deployed-policy results; depth-wise values and intervals are retained in the analysis artifact.

**Diagnostic scope.** Boundary enrichment and coverage of 11 of 12 planned suite/task units limit the sampled population, and sample admission differs from the online propagation rate. A strict replay audit records 40 branch rows above the $10^{-6}$ transition-visual tolerance (maximum $6.28 \times 10^{-4}$)

Table C.12: Recovery with and without depth-dependent action scaling.

| Propagation depth | Unscaled reference action | Depth-dependent scaling | Difference |
|---|---|---|---|
| 2 | 92.42% | 95.45% | +3.03 pp |
| 3 | 93.37% | 94.32% | +0.95 pp |
| 4 | 92.99% | 94.13% | +1.14 pp |

Table C.13: Recovery under the deployed VLA continuation policy by LIBERO suite. The Long entry contains two completed task units.

| Suite | Current-action recovery | Propagated-action recovery | Negative outcome flips | Positive outcome flips |
|---|---|---|---|---|
| Goal | 95.00% | 94.31% | 3.47% | 2.78% |
| Spatial | 93.89% | 93.61% | 4.72% | 4.44% |
| Object | 97.08% | 97.50% | 1.67% | 2.08% |
| Long | 91.25% | 92.08% | 4.79% | 5.63% |

Table C.14: Recovery by observable region.

| Region | Checkpoints | Future paths | Current-action recovery | Propagated-action recovery | Negative outcome flips | Positive outcome flips |
|---|---|---|---|---|---|---|
| Inner visual region | 93 | 744 | 95.16% | 93.15% | 5.65% | 3.63% |
| Outer visual region with screening flag | 19 | 152 | 96.71% | 98.03% | 1.32% | 2.63% |
| Beyond the visual consistency boundary | 20 | 160 | 90.00% | 93.75% | 1.88% | 5.63% |

Table C.15: Descriptive 95% checkpoint-cluster bootstrap intervals for the suite and observable-region summaries. Suite rows pool depths 0–4 and cluster repeated depths by base anchor; region rows use depth-specific restored checkpoints at depths 1–2. Entries are percentages: point estimate [interval].

| Group | Current-action recovery | Propagated-action recovery | Negative flips | Positive flips |
|---|---|---|---|---|
| Goal | 95.00 [89.86, 98.47] | 94.31 [89.86, 97.64] | 3.47 [1.67, 5.42] | 2.78 [0.97, 5.28] |
| Spatial | 93.89 [90.28, 97.22] | 93.61 [90.00, 96.94] | 4.72 [2.22, 7.36] | 4.44 [1.81, 7.36] |
| Object | 97.08 [95.28, 98.75] | 97.50 [95.69, 99.03] | 1.67 [0.56, 2.78] | 2.08 [0.97, 3.47] |
| Long | 91.25 [82.29, 98.13] | 92.08 [81.46, 99.58] | 4.79 [0.42, 10.63] | 5.63 [1.67, 10.63] |
| Inner visual region | 95.16 [92.88, 97.18] | 93.15 [90.19, 95.83] | 5.65 [3.36, 8.33] | 3.63 [2.02, 5.51] |
| Outer visual + screening flag | 96.71 [93.42, 99.34] | 98.03 [96.71, 99.34] | 1.32 [0.00, 2.63] | 2.63 [0.66, 5.26] |
| Beyond visual boundary | 90.00 [88.13, 91.88] | 93.75 [91.25, 96.25] | 1.88 [0.00, 3.75] | 5.63 [5.00, 6.88] |

and two depth-zero outcome disagreements. Together with the screening-flag mismatch, these facts restrict the analysis to the archived sample rather than deployment-wide validation.

### C.4 Hyperparameter Sensitivity

We evaluate local sensitivity around the shared LIBERO propagation configuration using the same matched 400-episode evaluation subset, task order, initial-state indices, random seed, and latency protocol as the component ablations. The OpenVLA and proposal checkpoints remain frozen. Each non-canonical row varies one parameter while holding all others at $(\delta_{\text{low}}, \delta_{\text{high}}, \epsilon_u, \alpha_1, \alpha_2) = (0.075, 0.15, 0.002, 1.0, 0.5)$, and the canonical configuration is evaluated only once.

The sensitivity results indicate an operating-point tradeoff rather than suite-specific threshold fitting. Varying $\delta_{\text{low}}$ changes propagation coverage in the expected direction—41.24%, 49.44%, and 54.47% for 0.05, 0.075, and 0.10—while exposing the corresponding speed–accuracy operating tradeoff. $\delta_{\text{high}}$ is comparatively insensitive over the tested range. Each non-canonical setting varies one global parameter while keeping the same configuration across the four LIBERO suites. The second-step scaling factor $\alpha_2$ materially affects success, with all three alternatives reducing SR on

Table C.16: One-factor-at-a-time sensitivity of Action Propagation on LIBERO. SR and speedup are macro averages across Goal, Spatial, Object, and Long, with 100 episodes per suite. ΔSR is measured relative to the canonical configuration. Propagation is the fraction of control steps that omit VLA inference.

| Varied quantity | Value | SR (%) | ΔSR (pp) | Speedup | Propagation (%) |
|---|---|---|---|---|---|
| Canonical | $(0.075, 0.15, 0.002, 1.0, 0.5)$ | 75.00 | 0.00 | 2.57× | 49.44 |
| $\delta_{\text{low}}$ | 0.05 | 71.25 | −3.75 | 2.30× | 41.24 |
| $\delta_{\text{low}}$ | 0.10 | 71.00 | −4.00 | 2.84× | 54.47 |
| $\delta_{\text{high}}$ | 0.10 | 73.50 | −1.50 | 2.50× | 48.19 |
| $\delta_{\text{high}}$ | 0.20 | 75.75 | +0.75 | 2.54× | 49.21 |
| $\epsilon_u$ | 0 | 75.00 | 0.00 | 2.57× | 49.44 |
| $\epsilon_u$ | 0.001 | 75.00 | 0.00 | 2.57× | 49.44 |
| $\epsilon_u$ | 0.004 | 75.00 | 0.00 | 2.57× | 49.44 |
| $\alpha_2$ | $1/3$ | 70.75 | −4.25 | 2.83× | 52.45 |
| $\alpha_2$ | $2/3$ | 72.00 | −3.00 | 2.68× | 50.03 |
| $\alpha_2$ | 1 | 69.50 | −5.50 | 2.62× | 49.34 |

the matched subset. Changing $\epsilon_u$ produced identical episode outcomes and execution-mode counts over the tested range. This equality is specific to the evaluated trajectories and values. Overall, the canonical configuration serves as the shared operating point within the tested neighborhood.

### C.5 SimplerEnv Portability Details

Table C.17: Local SimplerEnv decision latency and execution modes. The propagation-augmented latency is the mean decision latency of CogACT with Action Propagation enabled. It is not the latency of a propagation step in isolation. The two rates are fractions of control steps.

| Setting | Task | CogACT latency | + propagation latency | VLA inference rate | Action-propagation rate |
|---|---|---|---|---|---|
| WindowX | PutSpoon | 0.126 s | 0.081 s | 54.79% | 45.21% |
| | PutCarrot | 0.129 s | 0.080 s | 54.86% | 45.14% |
| | StackBlock | 0.126 s | 0.082 s | 56.98% | 43.02% |
| | PutEggplant | 0.127 s | 0.073 s | 49.41% | 50.59% |
| | Pooled | 0.127 s | 0.079 s | 54.01% | 45.99% |
| Google Visual Matching | PickCan | 0.126 s | 0.058 s | 43.27% | 56.73% |
| | MoveNear | 0.126 s | 0.062 s | 47.56% | 52.44% |
| | Drawer | 0.132 s | 0.078 s | 55.83% | 44.17% |
| | DrawerApple | 0.134 s | 0.123 s | 91.31% | 8.69% |
| | Pooled | 0.130 s | 0.080 s | 59.26% | 40.74% |
| Google Visual Aggregation | PickCan | 0.129 s | 0.058 s | 44.51% | 55.49% |
| | MoveNear | 0.132 s | 0.066 s | 51.49% | 48.51% |
| | Drawer | 0.134 s | 0.082 s | 60.39% | 39.61% |
| | DrawerApple | 0.134 s | 0.126 s | 95.64% | 4.36% |
| | Pooled | 0.132 s | 0.079 s | 59.65% | 40.35% |

The SimplerEnv experiment is designed as a component-level portability study. CogACT outputs continuous actions, so Verified Action Block Decoding and the proposal network are not used. The instantiated system consists of the frozen CogACT policy augmented only with Action Propagation.

This section reports the local CogACT reproduction used to interpret the portability study and the decision-latency and execution-mode statistics underlying the local Action Propagation results.

#### C.5.1 Local CogACT Reproduction

For WindowX, Tasks 1 through 4 are PutSpoon, PutCarrot, StackBlock, and PutEggplant. For Google Visual Matching and Google Visual Aggregation, they are PickCan, MoveNear, Drawer, and DrawerApple.

The local CogACT baseline and Action Propagation results in Table 2 are single fixed collections using the corresponding official settings and are not paired episode by episode. Changes in success rate are therefore reported descriptively rather than interpreted as paired causal effects.

#### C.5.2 Decision Latency and VLA Inference Rate

The deployed CogACT adapter is distinct from the LIBERO implementation. It resizes RGB observations to $16 \times 16$, normalizes them separately with DINO and SigLIP statistics, and concatenates the two $3 \times 16 \times 16$ tensors into a 1,536-dimensional signature. Relative $\ell_2$ deviation uses a $10^{-12}$ normalization floor. Continuous translation and rotation groups use a $10^{-12}$ near-stationary direction test, and the gripper uses sign consistency; this adapter therefore has no LIBERO-style $\epsilon_u$ setting. Propagation reuses a grounded CogACT action chunk at unit continuous authority, leaving the gripper unchanged. Table C.18 gives the fixed visual settings used for the reported collections; $\delta_{\text{init}}$ and $\delta_{\text{cont}}$ denote the initial and continued relative-deviation bounds of this adapter. The reported WindowX and non-DrawerApple Google deployments use a four-step refresh horizon; DrawerApple uses a one-step horizon.

Table C.18: Fixed Action Propagation configurations for the CogACT portability study. All rows use the shared 1,536-dimensional two-branch signature, a $10^{-12}$ normalization floor, unit continuous authority, and the motion certificate described in text.

| Setting | Tasks | $\delta_{\text{init}}$ | $\delta_{\text{cont}}$ |
|---|---|---|---|
| WindowX | PutSpoon, PutCarrot, StackBlock, PutEggplant | 0.075 | 0.076 |
| Google Visual Matching / Aggregation | PickCan, MoveNear, Drawer | 0.075 | 0.076 |
| Google Visual Matching / Aggregation | DrawerApple | 0.035 | 0.036 |

For DrawerApple, the visual-strict certificate additionally requires a three-step warm-up, two consecutive compatible Target outputs, target gap $0.75$, grounded-chunk cosine $0.90$, grounded-chunk gap $0.50$, and low-motion norms $0.06/0.09$ for translation/rotation. These configurations were fixed before the reported evaluation collections.

Because CogACT does not decode discrete action tokens, the portability study uses no proposal network. A propagation step therefore invokes no neural action generator, and FLOPs equal the VLA inference rate. The deterministic visual-consistency and motion-mode computations are included in measured decision latency but are not counted in FLOPs.

### C.6 Real-Robot Data, Training, and Deployment

#### C.6.1 Tasks and Data

The real-world evaluation uses the Alicia-D arm on three language-conditioned tasks:

- **Pick:** *Pick up the cylinder.*
- **Pick-and-place:** *Pick up the cylinder and place it in the tray.*
- **Conveyor-Belt Pick-and-Place:** pick up the moving cylinder from the conveyor belt and place it in the tray.

The robot is a Synria Alicia-D follower arm equipped with a parallel gripper. A fixed third-person RGB camera provides the observation. Although both front and wrist images are recorded during data collection, the policy receives only the front RGB image and the language instruction.

For the two static tasks, 150 teleoperated demonstrations are recorded at 20 Hz. Complete trajectories are divided into 142 training trajectories and eight held-out trajectories. Splitting is performed by trajectory rather than by frame. For Conveyor-Belt Pick-and-Place, demonstrations and physical evaluations are both conducted with the belt moving at 3 m/min.

OpenVLA's official fine-tuning guidance recommends collecting demonstrations at approximately 5–10 Hz (OpenVLA Team, 2024). Motivated by this low-frequency control regime, the 20 Hz recordings are converted into a future-endpoint action representation. Consecutive selected control targets in the constructed dataset are separated by six or seven source frames.

Table C.19: Real-robot data and training inventory. "Low-frequency transitions" reports training / held-out samples. VLA supervision records are deterministic seven-token traces used for proposal-network training.

| Task | Demos | Source frames | Train / held-out trajectories | Low-frequency transitions | VLA updates | Supervision records |
|---|---|---|---|---|---|---|
| Pick | 150 | 45,454 | 142 / 8 | 4,048 / 227 | 7,590 | 18,360 |
| Pick-and-place | 150 | 38,244 | 142 / 8 | 4,377 / 267 | 8,220 | 15,130 |
| Conveyor-Belt Pick-and-Place | 150 | 37,781 | 142 / 8 | 4,589 / 257 | 8,610 | 16,773 |

Let $t^+$ denote the next selected control target after observation time $t$. The six joint labels and the gripper label are

$$\Delta q_t = q_{t^+}^{\text{leader}} - q_t^{\text{follower}}, \qquad g_t = g_{t^+}^{\text{leader}}. \tag{C.1}$$

The policy therefore predicts the displacement from the currently observed follower configuration to the demonstrator's selected future joint endpoint, rather than the simultaneous leader–follower tracking error. The final observation of a trajectory is discarded when no future target exists, and labels never cross trajectory boundaries.

Normalization statistics for the six joint dimensions are computed from training trajectories only. The gripper remains an absolute command. For Pick, the transformation retains the complete lifting motion and a terminal closed-gripper hold. For Pick-and-place, it retains the complete release and a terminal open-gripper hold. Idle reset tails are removed.

### C.6.2 Policy Training and Deployment

Each real-world task fine-tunes a separate OpenVLA-7B policy. The static-task policies share a base model and use LoRA (Hu et al., 2022) rank 32, batch size 16, learning rate $5 \times 10^{-4}$, image augmentation, no weight quantization, and 30 data exposures. The selected Pick and Pick-and-place checkpoints occur at 7,590 and 8,220 optimizer updates.

Frozen task-specific VLA policies generate deterministic proposal supervision from seven-token actions and intermediate representations at five selected VLA layers. Pick uses eight image-augmentation passes during supervision export and Pick-and-place four.

Each one-layer proposal network trains for 100 epochs with block size 7, batch size 32, gradient accumulation 2, learning rate $2 \times 10^{-5}$, weight decay 0.05, 1,000 warm-up steps, and seed 7. All VLA and proposal-network checkpoints are selected and frozen before physical evaluation.

Separate NVIDIA A800 GPUs are used only for VLA and proposal-network training, not timed deployment; the RTX 4090 service performs BF16 inference. The deployment client captures a $640 \times 480$ RGB frame and associated follower configuration with the same 90% center crop as policy fine-tuning.

All three physical deployments use the same Action Propagation configuration, with a refresh horizon of two consecutive propagation steps; Table C.21 reports the visual, scaling, and motion-certificate settings. The visual signature is the fused OpenVLA tensor pooled without channel reduction; its normalization floor is distinct from the external execution-certificate resolution thresholds. Thus, Alicia-D does not reuse the LIBERO $\epsilon_u$ motion-mode parameter.

Table C.21: Fixed Action Propagation configuration for the physical deployments. The external certificate checks that resolved motion does not oppose the sent command.

| Tasks | $\delta_{\text{low}}$ | $\delta_{\text{high}}$ | Visual signature | Continuous scaling | Motion certificate |
|---|---|---|---|---|---|
| Pick; Pick-and-place; Conveyor-Belt P&P | 0.075 | 0.15 | $6 \times 16 \times 16 \to 1{,}536$; $\epsilon = 10^{-6}$ | $\alpha_d = 1/d$ | joints: $2\pi/4096$ rad; gripper: 1 milli |

Each returned action contains six joint displacements and one absolute gripper command. The client adds each displacement once to the captured follower configuration and sends the resulting absolute

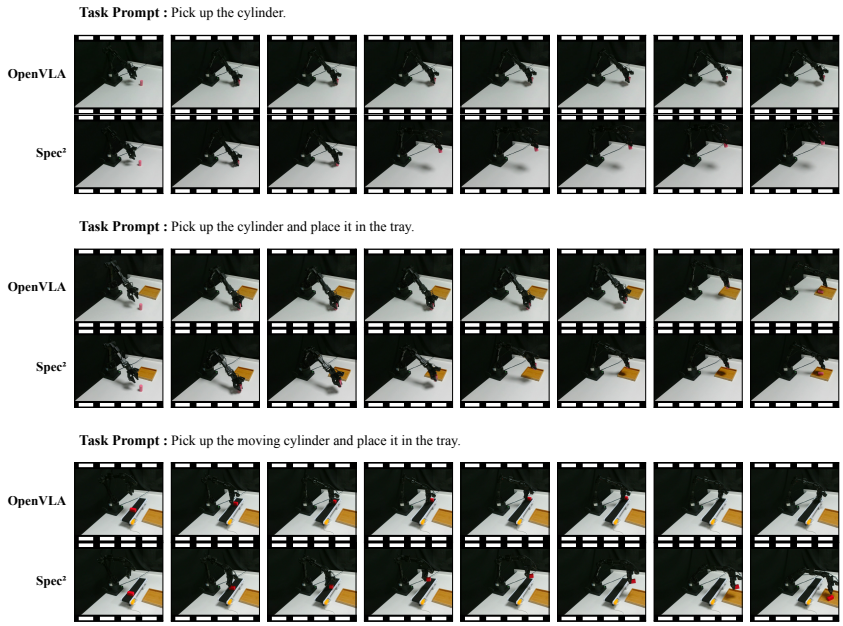



Figure C.1: Real-robot rollouts at eight video indices per method. Static tasks use matched indices; the conveyor-belt rows contrast a representative Spec$^2$ success with an OpenVLA failure. Frames are qualitative and excluded from metrics.

Table C.20: Detailed real-robot results over 50 trials per method and task. Model-path latency begins after image decoding and cropping.

| Task | Method | Successes | SR | Model-path latency | Speedup |
|---|---|---|---|---|---|
| Pick | OpenVLA | 47 / 50 | 94.0% | 180.48 ms | 1.00× |
| | Spec$^2$ | 45 / 50 | 90.0% | 60.16 ms | 3.00× |
| Pick-and-place | OpenVLA | 40 / 50 | 80.0% | 185.37 ms | 1.00× |
| | Spec$^2$ | 41 / 50 | 82.0% | 84.50 ms | 2.19× |
| Conveyor-Belt Pick-and-Place | OpenVLA | 23 / 50 | 46.0% | 174.90 ms | 1.00× |
| | Spec$^2$ | 36 / 50 | 72.0% | 66.30 ms | 2.64× |
| Three-task aggregate | OpenVLA | 110 / 150 | 73.3% | 180.25 ms | 1.00× |
| | Spec$^2$ | 122 / 150 | 81.3% | 70.32 ms | 2.56× |

joint target, with no residual accumulation, action amplification, gripper latching, or hidden task-phase controller.

The low-level controller uses a $30^\circ/\mathrm{s}$ joint-speed limit and accepts late responses, rejecting only nonfinite commands, joint-limit violations, or commands issued after the actuator reaches the $75^\circ\mathrm{C}$ thermal stop. The future endpoint is the policy target, not a measured end-to-end control frequency; model inference, network transport, and physical execution are measured separately.

### C.6.3 Evaluation and Timing

OpenVLA and Spec$^2$ are each evaluated in 50 physical trials per task (300 total) using the same predefined task-specific success criterion. Conveyor-Belt Pick-and-Place uses a belt speed of 3 m/min, matching data collection.

The server-side timer starts after image decoding and center cropping and includes model preprocessing, action generation, and, for Spec$^2$, Previous-Action Verification, Parallel Suffix Proposal

when invoked, and the visual- and motion-consistency computations. HTTP round-trip time and physical robot motion are excluded.

Across all tasks, OpenVLA and Spec$^2$ achieve 110/150 and 122/150 successes, with 180.25 and 70.32 ms task-mean latency ($2.56\times$, not the mean task-level speedup). In static-task logs, propagation covers 53% of steps, yielding a 47% VLA inference rate and 50.9% FLOPs. It invokes no VLA, while each remaining Spec$^2$ VLA step includes the proposal network. Separately, 15 stored Conveyor-Belt Pick-and-Place Spec$^2$ logs contain 1,644 control steps: 970 (59.00%) propagation steps and 674 (41.00%) VLA steps, with 652 initial and 318 continued propagation steps.

### C.7 VISUAL-GATE ROBUSTNESS UNDER CONTROLLED PERTURBATIONS

We stress-test the fixed LIBERO visual gate on 7,502 reference–current pairs from 80 episodes with photometric changes and $\pm 4\%$ image-space translations, the latter serving as proxies for small camera displacement. Persistent transformations affect both observations; transient ones affect only the current observation. Table C.22 shows 98.70%/98.37% agreement under persistent photometric/translation changes. No perturbation induces a false admission: transient changes instead suppress propagation and trigger fresh VLA evaluation.

Table C.22: Robustness of the deployed LIBERO visual gate. Persistent transformations affect both observations; transient transformations affect only the current one. Each category pools four transformations over the same 7,502 pairs. FA/FR denote false admission/rejection; $\Delta$Prop. is the propagation-rate change in percentage points.

| Regime | Perturbation | Agree. ↑ (%) | FA ↓ (%) | FR ↓ (%) | $\Delta$Prop. (pp) |
|---|---|---|---|---|---|
| Persistent | Photometric | 98.70 | 0.00 | 1.30 | $-1.30$ |
| Persistent | Translation proxy | 98.37 | 0.00 | 1.63 | $-1.63$ |
| Transient | Photometric | 58.62 | 0.00 | 41.38 | $-41.38$ |
| Transient | Translation proxy | 58.62 | 0.00 | 41.38 | $-41.38$ |

### C.8 SUPPLEMENTARY DECODER REPLAY

Table C.23: Original closed-loop component comparison for Parallel Suffix Proposal.

| Metric | Spec-VLA | Suffix proposal |
|---|---|---|
| Accepted proposals / round | 0.64 | 1.19 |
| Verified advance / round | 1.64 | 2.25 |
| Verification rounds | 202,399 | 147,435 |
| Latency (ms) | 178.5 | 157.6 |
| Speedup | $1.03\times$ | $1.16\times$ |
| SR (%) | 75.95 | 76.50 |

Figure 5 uses all 268 observations from the longest successful Goal episode in the saved 80-episode collection (task 5, episode 0; ties resolved by task and episode index). Decoders run separately at batch size one on the RTX 4090. Five timed calls follow ten warmups; synchronized GPU timing spans prepared model input to returned action and excludes preprocessing and simulator stepping. Panel (b) uses median timing; both panels show centered 11-step means/local 25th–75th percentiles with faint raw traces, whose bands indicate temporal variation rather than confidence intervals. AR has no proposal–verification rounds and appears only in panel (b).

### C.9 QUALITATIVE VISUAL-CHANGE OVERLAYS

Figures C.2 and C.3 are qualitative visualizations only; they are not used in gate, success, latency, or robustness measurements.

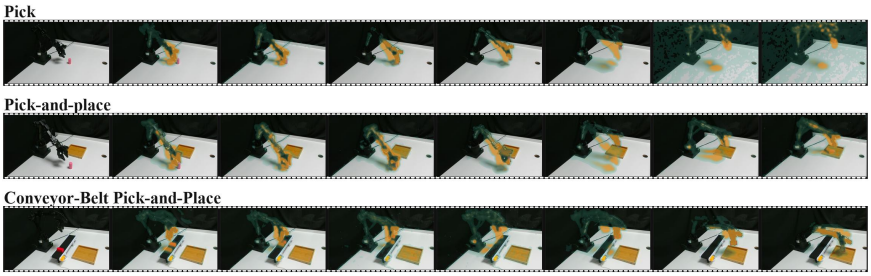



Figure C.2: Qualitative visual-change overlays for the real-robot tasks.




Figure C.3: Qualitative visual-change overlays for representative LIBERO simulation rollouts. Each row shows a sampled rollout sequence with temporal visual-change overlays.