FLOWLAP: EXECUTION-AWARE INFERENCE FOR FLOW-MATCHING VIDEO-ACTION WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-matching video-action models jointly generate visual futures and action chunks, but their rollout latency can introduce substantial transition waits at action-chunk boundaries. Real-Time Chunking (RTC) and related asynchronous methods can hide part of this latency, but they target action-only policies and cannot be transplanted to a video-action model, whose visual and action outputs share one joint forward pass; rehosted on the joint rollout with boundary context refresh, they still lack a handoff validity gate, so candidates that become stale or incompatible at execution time can be consumed. Other approaches, such as VLASH, Jetson-PI, LingBot-VA 2.0, and DexWorldModel, address temporal misalignment through model-specific adaptation or pretraining, limiting off-the-shelf deployment. We introduce FLOWLAP, an execution-aware inference framework for flow-matching video-action models with two components: a speculative joint rollout that overlaps inference with physical action execution while preserving the committed action prefix and gating candidates on timing and boundary continuity, and execution-state calibration that conditions the action branch on the state at which a candidate will execute. On the 50-task RoboTwin benchmark with 150 episodes per configuration, FLOWLAP improves success over an RTC baseline rehosted on the same joint rollout and boundary context handoff and differing only in the handoff validity gate, from 88.7% to 91.3% on FlowWAM and from 85.3% to 92.0% on LingBot-VA with calibration. Relative to matched reduced-work synchronous baselines, FLOWLAP reduces mean transition wait from 896.6 to 39.1 ms on FlowWAM and from 839.9 to 232.4 ms on LingBot-VA; execution-state calibration further reduces the latter to 71.8 ms. The p90 transition wait is 0 ms for FlowWAM and for calibrated LingBot-VA, while uncalibrated LingBot-VA retains a residual p90 of 1,280.8 ms.

1 INTRODUCTION

Flow-matching video-action models have emerged as a powerful paradigm for robot manipulation. Unlike conventional policies that map one observation directly to one action, they model a generative distribution over action sequences while predicting how the scene will evolve, allowing visual prediction and action generation to inform one another (Hu et al., 2025; Zhu et al., 2025; Cen et al., 2025; Jang et al., 2025). This generative view builds on denoising formulations for high-dimensional control (Ho et al., 2020), action-diffusion policies with temporally coherent chunks (Chi et al., 2023), and low-cost bimanual learning with short action sequences (Zhao et al., 2023). Flow matching provides an efficient continuous-time formulation for generative modeling (Lipman et al., 2023), and systems such as π_0, FlowWAM, and LingBot-VA apply this interface to robot control and bimanual manipulation (Black et al., 2025a; Chen et al., 2026; Li et al., 2026). Vision-language-action models transfer web-scale semantic knowledge to physical control and provide reusable visual-action interfaces (Brohan et al., 2023; Kim et al., 2024; Octo Model Team, 2024). A recurring closed-loop interface across predictive visual-policy, world-action, and video-action systems is to execute an action block and then re-anchor the next rollout with the real observation collected at its end; in autoregressive implementations this also updates persistent tokens or KV cache (Hu et al., 2025; Zhu et al., 2025; Cen et al., 2025; Chen et al., 2026; Li et al., 2026; Walke et al., 2023; Khazatsky et al., 2024).

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.

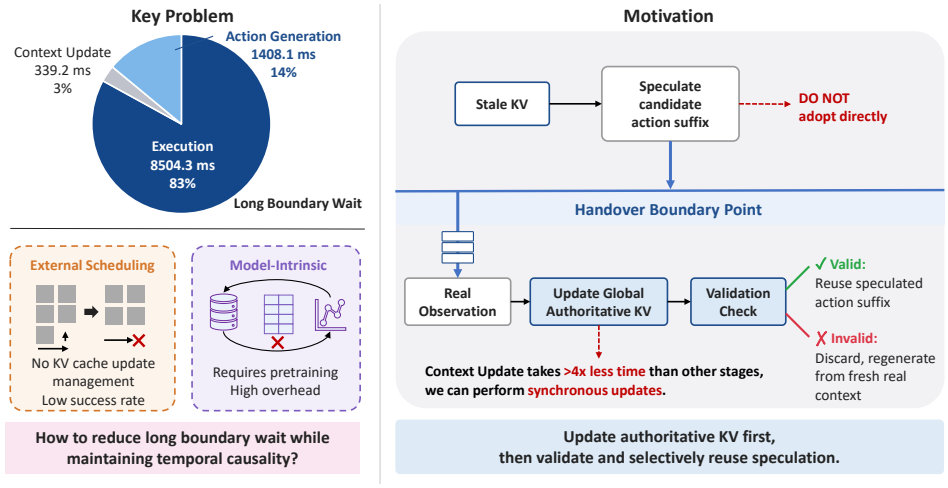

Figure 1: Overview of problem and motivation. Left: Latency breakdown and limitations of prior speculative KV methods. Right: Our handoff validation mechanism, which validates a speculative action suffix on timing and boundary continuity and refreshes authoritative KV from real observations for the subsequent rollout. Leveraging the more-than-fourfold latency gap between inference and context update reduces boundary wait.

This pipeline introduces a practical deployment bottleneck: the computation for a new rollout can exceed the remaining execution time of the current action chunk, introducing blocking boundary-transition waits of hundreds to over one thousand milliseconds during robot operation. A natural remedy is computation overlap—generating the next model output while the robot is physically moving (Xiao et al., 2020)—but it requires an explicit contract for model context at each handoff. Existing asynchronous inference approaches for robot world-models broadly fall into two categories: external scheduling approaches, and model-intrinsic approaches that build asynchronous logic inside model design or training. **(1) External Scheduling.** Black et al. (2025b) introduced Real-time Chunking (RTC), one such approach that preserves the already-executed action prefix and completes the remaining action suffix. RTC targets action-only policies, and a video-action model has no action-only pathway—its visual and action outputs share one joint forward pass—so RTC cannot be transplanted and must be rehosted on the joint rollout. Our RTC baseline rehosts it on the joint rollout with the same boundary context handoff as FlowLap but without a handoff validity gate; an early-ready candidate that is stale or incompatible at execution time is then still consumed, yielding absolute success drops of 4.6 and 4.7 percentage points for FlowWAM and LingBot-VA, respectively. **(2) Model-Intrinsic.** Alternative approaches address temporal misalignment at the cost of generality. Tang et al. (2025) conditions on the robot state at future execution time, and Yang et al. (2026) predicts a future environment representation from committed actions and schedules VLM versus action-expert invocations by confidence. Both target action-only vision-language-action (VLA) policies rather than joint video-action (VA) world models, so they need model-specific correction or scheduling modules and cannot be attached off-the-shelf. Other systems including LingBot-VA 2.0 and DexWorldModel likewise bake asynchronous prediction into specialized latent representations, tokenizers, or pretraining pipelines (Zhang et al., 2026; Deng et al., 2026); maintaining task success under asynchronous rollout thus requires expensive pretraining. The base LingBot-VA model also describes a native asynchronous prediction scheme, but its implementation is not part of the public release, so it cannot be run as a baseline; our comparisons therefore use released synchronous references and an RTC baseline with matched boundary context handoff.

Taken together, the existing asynchronous approaches fail to simultaneously achieve three key goals: (1) eliminating high boundary-transition latency via inference-execution computation overlap; (2) validating each speculative candidate at the handoff so that stale or incompatible actions are rejected, avoiding success degradation from consumed stale internal states; and (3) supporting off-the-shelf deployment purely at the inference layer across diverse pre-trained video-action world models, without modifying model backbones or requiring additional retraining. To tackle these issues, we present

FLOWLAP, a reusable execution-aware inference framework for flow-matching video-action models with two key contributions:

(1) Speculative Joint Rollout. To eliminate boundary-transition latency while validating candidates at each handoff, we introduce an inference-level scheduling protocol that rehosts external scheduling on the joint video-action rollout and overlaps generation with robot physical execution. It strictly preserves already-committed action prefixes and enforces that persistent context is only updated from real environment observations. A handoff validity gate discards stale, late or incompatible speculative candidates, with a guaranteed fallback to generation from fresh real context. This mechanism requires no modification to pre-trained model backbones. Experiments show that our method can reduce the boundary-transition latency from 896.6 ms to 39.1 ms for FlowWAM and from 839.9 ms to 232.4 ms for LingBot-VA compared against the matched efficient synchronous baseline.

(2) Execution-State Calibration. To mitigate temporal mismatch between speculative generation time and future action execution time under asynchronous rollout, we propose a lightweight calibration component. It estimates the anticipated robot state at candidate execution time and injects future-state conditions exclusively into the action branch via zero-initialized residual adapters. The visual branch remains untouched and conditioned only on real observations; the pre-trained model backbone is fully frozen, avoiding costly full-model retraining. Experiments show that our calibration further lifts task success from 86.0% to 92.0% and reduces mean transition wait from 232.4 ms to 71.8 ms.

We evaluate FLOWLAP on the 50-task RoboTwin benchmark (150 episodes per configuration) with FlowWAM and LingBot-VA, two independently implemented video-action models with real-context refresh at closed-loop handoffs. Averaged over FlowWAM FlowLap and calibrated LingBot-VA FlowLap, success improves over the RTC baseline with matched context handoff by 4.7 points (5.4% relative) and mean transition wait drops by 812.8 ms (93.5% relative) versus the efficient synchronous baseline, showing that an execution-aware, interface-level strategy generalizes across architectures without sacrificing task quality.

2 BACKGROUND

At control step t, a flow-matching video-action model maps a real context C_t to a predicted visual continuation and an action chunk:

$$(\widehat{V}_{t+1}, A_t) = M_\theta(C_t). \tag{1}$$

The model obtains this pair by numerically solving its video and action trajectories from noise or a warm-start state. The action chunk A_t is sent to the robot and remains owned by the executor until its scheduled portion has run.

When execution finishes, the new observation o_{t+1} and the executed action update the persistent context,

$$C_{t+1} = U(C_t, o_{t+1}, A_t). \tag{2}$$

This real-observation re-anchoring is common across predictive visual-policy, world-action, and video-action models (Hu et al., 2025; Zhu et al., 2025; Cen et al., 2025; Chen et al., 2026; Li et al., 2026). Contact, camera motion, and object state can differ from prediction, so the next rollout must use C_{t+1}; in autoregressive models this also advances persistent tokens or KV cache. A synchronous controller waits for $M_\theta(C_{t+1})$ before sending the next chunk.

Let t_{exec} be the end of the current action execution interval and t_{ready} the time at which the next action is ready for the executor. We measure observable chunk-transition wait as

$$W_t = \max\left(0, t_{\text{ready}} - t_{\text{exec}}\right), \tag{3}$$

and call a transition wait-free when $W_t = 0$. These are host-level control metrics: they measure the stall visible at the action boundary, not overlap between individual GPU denoising kernels.

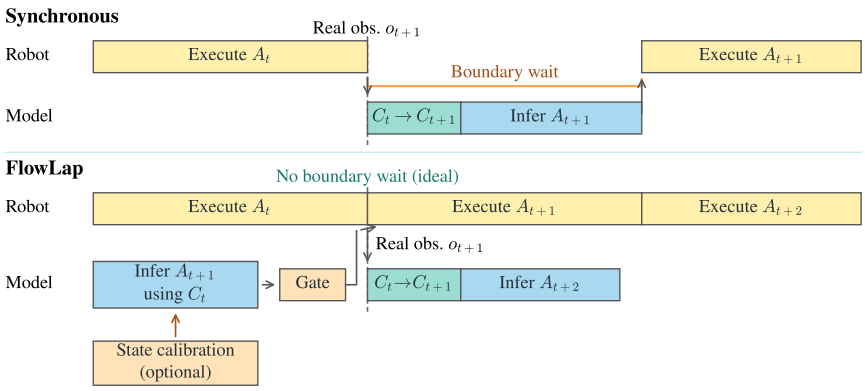

Figure 2: Overview of FLOWLAP. The synchronous pipeline suffers from boundary wait: robot execution stalls after o_{t+1} is captured, waiting for context update and action inference. Our FLOWLAP precomputes action A_{t+1} using C_t before the real observation arrives. A gate mechanism maintains temporal causality and can eliminate covered boundary wait when speculation finishes within the execution window. Optional state calibration further improves pre-inferred actions.

3 METHOD

3.1 MOTIVATION

To identify the source of transition wait, we profile the released LingBot-VA synchronous reference on frozen traces. Figure 2 contrasts serialized execution with an ideal overlap schedule in which the gate finishes before the boundary and the real-context refresh overlaps the following execution window.

The reference spends 1,408.1 ms in inference and 339.2 ms in context update around 8,504.3 ms of action execution. The two model-side phases therefore total 1,747.3 ms, while the measured mean transition wait is 1,741.9 ms. Let T_{inf}, T_{ctx}, and T_{exec} denote inference, context-update, and action-execution time, respectively. At the control-loop level, a synchronous controller exposes the model-side work directly at the boundary:

$$W_t^{\text{sync}} \approx T_{\text{inf}} + T_{\text{ctx}}. \tag{4}$$

Overlap changes the critical path rather than merely shortening its individual components. Once the next joint rollout is launched during execution, the residual boundary wait is approximately

$$W_t^{\text{overlap}} \approx \max\left(0, T_{\text{inf}} + T_{\text{ctx}} - T_{\text{exec}}\right), \tag{5}$$

when the candidate and context preparation complete within the current execution window; actual wait also includes scheduling overhead and fallback. Step-cut, denoising-step reduction, and context-cache optimizations instead reduce T_{inf} or T_{ctx} while retaining the same serial dependency, so they remove wait only by the amount of work removed. Overlap makes the action-execution interval useful computation time, eliminating the covered portion of the wait even when total model work is unchanged or slightly larger: the efficient synchronous baseline uses the same 4/4 and 2/5 step budgets as FlowLap, yet still waits 896.6 ms and 839.9 ms, while scheduling inside the execution window cuts this to 39.1 ms and 232.4 ms (71.8 ms with calibration). Overlap, not request-time reduction alone, thus drives transition responsiveness.

A structural constraint is that a flow-matching video-action model produces its visual and action outputs in a single joint forward pass, so it has no action-only pathway and no method defined for action-only policies can be applied without rehosting its scheduling on the joint rollout (Table 1). Black et al. (2025b) introduced RTC for action-only policies, where it parallelizes action generation through prefix protection and suffix completion; rehosted on a video-action model it also refreshes the boundary context from real observations, but it does not validate the speculative candidate, so a suffix that is ready within budget yet stale or incompatible with the committed prefix can still

Table 1: Comparison of asynchronous inference approaches.

Methods	Async. inference	Prefix protection	Joint visual-action output	Boundary context refresh	Handoff validity gate	Future-state condition
RTC (Black et al., 2025b)	✓	✓				
RTC + context handoff (ours)	✓	✓	†	✓		
VLASH (Tang et al., 2025)	✓					✓
Jetson-PI (Yang et al., 2026)	✓					✓
FLOWLAP (ours)	✓	✓	✓	✓	✓	✓

be consumed. VLA-targeted methods such as Tang et al. (2025) and Yang et al. (2026) inherit the same action-only assumption, and model-specific world-action systems such as Zhang et al. (2026) and Deng et al. (2026) embed asynchronous prediction in native representations, so preserving SR under overlap requires compatible interfaces and costly model-specific training. FlowLap rehosts scheduling on the native joint rollout, keeps real context authoritative, protects the executor-owned prefix, and admits a speculative suffix only after a handoff validity gate that checks timing, identity, shape, and boundary continuity.

Our RTC baseline rehosts RTC on the same joint rollout and applies the same boundary context handoff as FlowLap, but retains RTC's published protocol without the FlowLap handoff validity gate (†: the joint output is produced by the architecture, not chosen by the method); the empty RTC gate entry marks the component under test rather than a claim that RTC cannot be paired with a gate.

3.2 SPECULATIVE JOINT ROLLOUT

Overlapping generation and execution. Let $B_t = (b_t^0, \dots, b_t^{H-1})$ be the action chunk owned by the executor at step t, and let E be the portion executed before the next handoff. The executor first commits the prefix $P_t = B_t[0 : E)$; this prefix is no longer allowed to change. During the same execution interval, FlowLap launches the next joint rollout from the latest available real context:

$$\begin{aligned} (\widehat{V}_{t+1}, \widetilde{B}_{t+1}) &= M_\theta(C_t; P_t), \\ C_{t+1} &= U(C_t, o_{t+1}, P_t). \end{aligned} \tag{6}$$

The rollout is speculative because its action output may be consumed at the next handoff, but its visual continuation is never treated as an observation: only the real observation o_{t+1} and the committed prefix P_t advance the persistent real context, so inference can run ahead without letting a predicted context overwrite the state used for future requests. The resulting prefix-ownership and real-context invariants are formalized in Theorem A.4.

The runtime keeps at most one candidate in flight for a given handoff. Each pending request carries an episode generation, a logical target index, an expected temporal position, and the context version at which it was launched; these identifiers are part of the scheduling contract, preventing a result from a previous episode, an earlier handoff, or a stale context from being committed after the control state has advanced.

Action continuity. The in-flight rollout must not rewrite controls already being executed. During iterative generation, we preserve the committed prefix with a binary mask m:

$$x^{(j)} \leftarrow (1 - m) \odot x^{(j)} + m \odot x^{(j)}_{\text{commit}}, \tag{7}$$

where $m_i = 1$ for the controls owned by the executor. After the prefix has run, the candidate is shifted once to expose the unexecuted suffix:

$$\mathcal{S}_E(B) = (B_E, \dots, B_{H-1}) \in \mathcal{A}^{H-E}, \tag{8}$$

This preserves continuity across chunks while allowing the suffix to respond to the latest model prediction, so the candidate is a continuation of the committed trajectory rather than an independent action block. Because the suffix is obtained by deterministic shifting of the same joint video-action rollout, no independently resampled visual component is introduced (Theorem A.6).

Algorithm 1 Speculative Joint Rollout During Action Execution

Input: real context C_0, video-action model M_θ, execution horizon E
Initialize: $B_0 \leftarrow M_\theta(C_0)_{\text{action}}$
while control loop is running **do**
  Commit $P_t \leftarrow B_t[0:E)$; start $M_\theta(C_t; P_t)$ while executing P_t
  Attach episode, target-index, time-position, and context-version metadata
  While executing P_t, evaluate the Boolean commit decision Commit_t on the ready candidate
  Observe $o_{t+1} \leftarrow \textsc{Observe}()$; update $C_{t+1} \leftarrow U(C_t, o_{t+1}, P_t)$
  if $\text{Commit}_t = \text{true}$ **then**
    $B_{t+1} \leftarrow \mathcal{S}_E(\widetilde{B}_{t+1})$; refresh context in parallel
  else
    Discard candidate; $B_{t+1} \leftarrow [M_\theta(C_{t+1})]_{\text{action}}$
  end if
  Arm at most one candidate for the next handoff when no request is in flight
end while

Candidate selection. While the current chunk is still executing, the runtime makes a model-independent Boolean commit decision from information already available in the execution window, so the decision itself does not add to the boundary critical path. The candidate may be committed only if it is ready within the boundary budget and its action suffix passes the boundary-continuity check, which compares the candidate suffix against the committed boundary control per semantic action group; Appendix A.2 gives the residual and the per-backbone thresholds. Optional projection or uncertainty checks may repair a small mismatch or reject an unsafe suffix. We denote this decision by Commit_t, where "true" means that the candidate is admissible at the current handoff:

$$\text{Commit}_t = \begin{cases} \text{true}, & \text{candidate is ready and all validity checks pass,} \\ \text{false}, & \text{otherwise.} \end{cases} \tag{9}$$

The next action is selected as

$$B_{t+1} = \begin{cases} \mathcal{S}_E(\widetilde{B}_{t+1}), & \text{Commit}_t = \text{true}, \\ [M_\theta(C_{t+1})]_{\text{action}}, & \text{Commit}_t = \text{false}. \end{cases} \tag{10}$$

An unavailable, late, stale, or incompatible candidate is discarded rather than waited on, and the fallback is a fresh rollout from the updated real context; speculative inference never changes the context used by the fallback request. Theorem A.5 quantifies the computation hidden by this overlap.

The runtime submits at most one speculative request per handoff and falls back to a fresh-context rollout whenever the pending candidate is rejected; Algorithm 1 summarizes this protocol.

3.3 Execution-State Calibration

The speculative candidate is launched from one execution state but consumed after approximately d_t controls, so the action branch predicts for a state reached later while the visual branch must still be grounded in the real observation at the next handoff. Execution-state calibration addresses this temporal mismatch without replacing the real-context update.

Let s_t denote the state available when a candidate is launched, and let $A_t^{\text{delay}} = (a_t^0, \ldots, a_t^{K_t-1})$ be the committed controls expected to execute before that candidate is consumed. A state propagator Φ estimates the state at the candidate's execution time:

$$\begin{aligned} \widehat{s}_{t+K_t} &= \Phi(s_t, A_t^{\text{delay}}), \quad & e_t &= \varphi(s_t), \\ e_t^{\star+} &= \varphi(s^\star_{t+K_t}), & \widehat{e}_t^{+} &= \varphi(\widehat{s}_{t+K_t}), \\ \delta_t &= \widehat{e}_t^{+} - e_t. \end{aligned} \tag{11}$$

Here Φ propagates the action plan, φ maps state to normalized action features, and the starred quantity is analytical; at runtime only $\widehat{e}_t^{+}$ is available. We form the calibration condition

$$c_t = \left[e_t, \widehat{e}_t^{+}, \delta_t, \eta(K_t), m_t\right], \tag{12}$$

where $\eta(K_t)$ encodes delay and m_t masks invalid components. The condition is normalized with training statistics and set to zero when disabled.

Table 2: Evaluation configurations.

Configuration	FlowWAM	LingBot-VA
Released synchronous reference	25 video / 50 action steps	3 video / 10 action steps
Efficient synchronous baseline	4 / 4 steps; reduced-work control	2 / 5 steps; reduced-work control
RTC	4 / 4 steps; context-handoff baseline	2 / 5 steps; context-handoff baseline
FLOWLAP (ours)	4 / 4 steps; speculative rollout	2 / 5 steps; speculative rollout
FLOWLAP + execution-state calibration	–	2 / 5 steps; optional calibration

The condition is injected only into action features through a zero-initialized residual adapter,

$$h_j^{\mathrm{act}} \leftarrow h_j^{\mathrm{act}} + \lambda\, r_\psi(c_t), \qquad 0 \le \lambda \le 1, \tag{13}$$

where $c_t = 0$ is an identity path. Training samples a launch point and delay, propagates the committed controls, masks unavailable targets, and predicts the delay-shifted action target:

$$\mathcal{L}_{\mathrm{state}} = \frac{\sum_i M_i w_i \left\| v^{\mathrm{act}}_{\theta,\psi}(z_i, A^{\mathrm{ctx}}, c_t) - v_i^\star \right\|_2^2}{\sum_i M_i + \varepsilon}. \tag{14}$$

where M_i identifies valid targets and w_i balances dimensions or positions. At inference, calibration is used only for valid propagated states within range; otherwise the condition is removed and the runtime follows the uncalibrated or fresh-context path in Sec. 3.2. The visual branch and persistent context remain real-observation conditioned. Theorem A.10 formalizes how propagation accuracy translates into action-target accuracy, and Corollary A.11 states when calibration tightens the error guarantee.

We freeze the pretrained backbone and optimize only the zero-initialized adapter r_ψ on delay-shifted targets, leaving the backbone and attention structure unchanged; Tang et al. (2025) instead fine-tunes a VLA on temporally offset state-action sequences.

4 EVALUATION

4.1 SETUP

Benchmark. Chen et al. (2025) present RoboTwin 2.0, a scalable simulation benchmark for bimanual manipulation. We use its standard clean-scene configuration over all 50 tasks, with three episodes per task (150 episodes per model and configuration). The suite includes pickup and placement, stacking and ranking, bimanual handover, articulated-object manipulation, and contact-rich tasks. Success is the simulator's native task-completion signal.

Models. Chen et al. (2026) introduce FLOWWAM, a dual-stream diffusion world-action model adopting optical flow as video-native action representation. Li et al. (2026) propose LINGBOT-VA, an autoregressive diffusion video-action world model that encodes visual and action tokens jointly and re-anchors closed-loop rollouts via real observations. Both models follow the boundary-refreshed real-context interface in Sec. 2, with persistent token/KV states for the autoregressive variant. We keep each model's native visual/action interfaces and integrate FLOWLAP at the request and handoff layer.

Configurations. Table 2 lists evaluation setups. The efficient synchronous baseline shares FlowLap's reduced inference budget but waits for rollout completion before dispatching subsequent action chunks, serving as a low-work control to isolate scheduling effects. We also rehost RTC (Black et al., 2025b) on the same boundary-refreshed flow-matching video-action interface with the same boundary context handoff, so it differs from FlowLap only in the handoff validity gate, isolating the effect of candidate validation. Execution-state calibration is only evaluated for LingBot-VA, as its checkpoint supports this module. We report success rate, mean and p90 transition wait as the primary task and responsiveness measures. The complete evaluation also records wait-free transitions, candidate consumption and fallback, and inference request count.

4.2 END-TO-END EXPERIMENTS

Table 3: End-to-end evaluation on 150 RoboTwin episodes. SR is success rate.

	FlowWAM			LingBot-VA		
Configuration	SR (%)	Mean wait (ms)	p90 wait (ms)	SR (%)	Mean wait (ms)	p90 wait (ms)
Reference	92.7	12,602.7	15,069.6	**92.7**	1,741.9	2,281.9
Efficient sync.	93.3	896.6	981.5	90.0	839.9	1,112.4
RTC	88.7	**29.8**	**0.0**	85.3	264.1	1,347.2
FLOWLAP	**91.3**	39.1	**0.0**	86.0	232.4	1,280.8
FLOWLAP + execution-state calibration	–	–	–	**92.0**	**71.8**	**0.0**

Table 4: Representative per-task success rate on the 50-task RoboTwin benchmark. Each entry is the success rate (%) over three episodes. The final row is the mean SR over the ten displayed tasks.

	FlowWAM				LingBot-VA				
Task	Ref.	Sync.	RTC	FLOWLAP (ours)	Ref.	Sync.	RTC	FLOWLAP (ours)	FLOWLAP + cal.
Blocks Ranking Size	100.0	100.0	33.3	100.0	100.0	66.7	100.0	100.0	100.0
Scan Object	66.7	100.0	100.0	100.0	33.3	100.0	33.3	100.0	66.7
Open Microwave	66.7	66.7	33.3	66.7	100.0	33.3	0.0	0.0	100.0
Place Can Basket	66.7	33.3	33.3	66.7	100.0	100.0	66.7	66.7	66.7
Stack Bowls Three	100.0	100.0	33.3	66.7	100.0	100.0	100.0	100.0	100.0
Handover Block	100.0	66.7	66.7	100.0	100.0	100.0	100.0	100.0	100.0
Pick Dual Bottles	100.0	100.0	100.0	100.0	100.0	100.0	66.7	100.0	100.0
Place Fan	100.0	100.0	100.0	100.0	100.0	100.0	66.7	100.0	100.0
Place Mouse Pad	100.0	100.0	100.0	100.0	100.0	100.0	33.3	66.7	66.7
Place Object Basket	66.7	100.0	66.7	66.7	100.0	100.0	66.7	100.0	100.0
Mean SR (%)	86.7	86.7	66.7	86.7	93.3	90.0	63.3	83.3	90.0

Tables 3 and 4 report the aggregate metrics and ten representative tasks.

Compared with synchronous baselines. Against the efficient synchronous baselines, FLOWLAP reduces mean transition wait by 857.5 ms (95.6%) on FlowWAM and 607.5 ms (72.3%) on uncalibrated LingBot-VA; execution-state calibration reduces LingBot-VA wait by a further 160.6 ms to 71.8 ms (91.4% overall). The p90 wait is 0.0 ms for FlowWAM and for calibrated LingBot-VA. Success rate drops by 2.0 percentage points on FlowWAM and by 4.0 percentage points for uncalibrated LingBot-VA; calibration raises LingBot-VA success by 6.0 points, ending 2.0 points above the efficient synchronous baseline. Both reference models already succeed on 92.7% of episodes, so headroom is small and a few episodes correspond to only one to four points; we claim no significance for individual small deltas and do not read the drops as a systematic cost of overlap.

The gain comes from moving the next rollout into the execution interval, so it finishes before the next action is needed while the speculative prefix leaves in-progress controls unchanged. It does not rely on reducing request work: FLOWLAP issues 2,349 requests versus 1,473 for the efficient synchronous control on FlowWAM and 2,647 versus 1,290 on LingBot-VA (2,290 with calibration), so total model-side work increases and the benefit is placement rather than less GPU work (Appendix A.6). The ten-task mean matches the efficient synchronous baseline for FlowWAM (86.7%) and calibrated LingBot-VA (90.0%), consistent with hidden rather than discarded computation.

Compared with RTC with context handoff. The RTC baseline rehosts RTC on the same boundary-refreshed flow-matching video-action interface: it runs the same joint rollout, protects the prefix, completes the suffix, and performs the same boundary context handoff from real observations as FlowLap. It differs only in the FlowLap handoff validity gate, so a candidate that is ready within budget yet stale or incompatible with the committed prefix can still be consumed. Relative to this baseline, FLOWLAP raises success by 2.6 points on FlowWAM and 0.7 points on uncalibrated LingBot-VA, with calibration raising LingBot-VA by 6.7 points. FLOWLAP keeps 0.0 ms p90 wait on FlowWAM at a 9.4 ms mean-wait cost; uncalibrated LingBot-VA reduces mean and p90 wait by 31.7 ms and 66.4 ms, and calibration reaches 71.8 ms mean and 0.0 ms p90. Because both methods share the joint rollout and boundary context handoff, these numbers isolate the contribution of the candidate-validation gate.

FlowLap instead keeps real context authoritative, updating persistent context only from the real handoff observation and admitting a suffix only if the gate accepts its identity, timing, shape, and continuity. Table 4 shows gains on five of ten FlowWAM and six of ten calibrated LingBot-VA tasks; the full 50-task tables in Appendix A.8 also contain regressions, so we read this as a favorable aggregate rather than a uniformly broad gain. Fallback counts drop from 69 to 5 on FlowWAM and from 426 to 158 on calibrated LingBot-VA.

4.3 Ablation Study

Table 5: Ablation studies on 150 RoboTwin episodes per configuration.

	FlowWAM				LingBot-VA			
Configuration	SR (%)	Mean (ms)	p50 (ms)	p90 (ms)	SR (%)	Mean (ms)	p50 (ms)	p90 (ms)
Pure overlap (no gate)	80.7	6.8	0.0	0.0	82.0	0.0	0.0	0.0
Half launch	77.3	657.5	947.7	1,188.9	83.3	540.1	197.9	1,294.2
FlowLap	91.3	39.1	0.0	0.0	86.0	232.4	0.0	1,280.8
Pure overlap + calibration	–	–	–	–	83.3	0.0	0.0	0.0
FlowLap + calibration	–	–	–	–	92.0	71.8	0.0	0.0

Speculative Joint Rollout without Handoff Gate. FlowWAM reaches 80.7% SR with mean/p50/p90 waits of 6.8/0.0/0.0 ms, while LingBot-VA reaches 82.0% with 0.0/0.0/0.0 ms. Removing the handoff gate reduces SR by 10.6 and 4.0 points, while lowering mean wait by 32.3 and 232.4 ms, respectively. Speculative rollout can therefore hide inference latency, but without validating phase, prefix, and suffix consistency at the handoff, early-ready candidates may be stale at execution time.

Speculative Joint Rollout with Half Launch. Half launch achieves 77.3% (116/150) and 83.3% (125/150) SR on FlowWAM and LingBot-VA, with mean/p50/p90 waits of 657.5/947.7/1,188.9 ms and 540.1/197.9/1,294.2 ms, respectively. Launching midway leaves less time to hide inference, producing residual waits and degraded success; launch timing controls the overlap window but cannot replace handoff validation.

Execution-State Calibration. On LingBot-VA, pure overlap with calibration improves SR from 82.0% to 83.3% while maintaining 0.0 ms mean, p50, and p90 wait; calibrated FlowLap further reaches 92.0% SR with 71.8 ms mean wait and 0.0 ms p50/p90. Calibration is evaluated across the complete LingBot-VA task suite under the same runtime protocol. It reduces temporal mismatch by conditioning the action branch on the anticipated execution state, but does not enforce context freshness or prefix/suffix consistency. These results demonstrate complementary roles: calibration improves the predicted action, while the handoff gate determines whether the speculative candidate is safe to commit.

5 Conclusion

We presented FlowLap, an execution-aware inference framework for flow-matching video-action world models. On FlowWAM and LingBot-VA, which use boundary-refreshed closed-loop context interfaces, FlowLap reduces mean transition wait by 95.6% and 72.3% relative to efficient synchronous controls, while completing 137/150 and 129/150 episodes. Execution-state calibration on LingBot-VA further reaches 138/150 episodes and 71.8 ms mean wait. Relative to the RTC baseline with the same boundary context handoff, FlowLap improves success by 4 and 1 episodes, while the calibrated LingBot-VA configuration improves by 10 episodes. These results show that authoritative real-context refresh and gated speculative joint rollout trade extra computation for a favorable task-quality and responsiveness balance.

AI USE STATEMENT.

Generative AI tools (OpenAI Codex) were used for recommended-disclosure tasks including language editing, LaTeX maintenance, and creation and checking of the supplementary validation script. For the required-disclosure category, AI assistance included interpreting, organizing, and summarizing already completed benchmark outputs, as well as assisting with the theoretical analysis, proof drafting, and checking the scope of conditional mathematical claims. It was not used to generate synthetic data; design or provide feedback on methodology or experiments; implement FLOWLAP; translate text; clean or reformat datasets; perform qualitative or thematic analysis; or run the evaluations. The authors independently reviewed the assumptions, derivations, AI-assisted text and code, and checked every reported number against the frozen benchmark outputs. The authors take responsibility for the final manuscript, results, and all supplementary materials produced with AI assistance.

REPRODUCIBILITY STATEMENT.

The main text specifies the model interfaces, execution-aware handoff, calibration condition, evaluation protocol, and metrics. The anonymous supplementary materials provide current evaluation code snapshots for FlowWAM and LingBot-VA, the execution-state adapter checkpoint used by the calibrated evaluation, and the result summaries corresponding to the reported configurations. Base model weights, RoboTwin, and simulator assets remain external dependencies.

REFERENCES

K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *RSS*, 2025. DOI: 10.15607/rss.2025.xxi.010.

K. Black, M. Y. Galliker, and S. Levine. Real-time execution of action chunking flow policies. In *NeurIPS*, 2025. arXiv:2506.07339.

A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. arXiv preprint arXiv:2307.15818, 2023.

J. Cen, C. Yu, H. Yuan, Y. Jiang, S. Huang, J. Guo, X. Li, Y. Song, H. Luo, F. Wang, D. Zhao, and H. Chen. WorldVLA: Towards autoregressive action world model. arXiv preprint arXiv:2506.21539, 2025.

T. Chen, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. arXiv preprint arXiv:2506.18088, 2025.

Y. Chen, et al. FlowWAM: Optical flow as a unified action representation for world action models. arXiv preprint arXiv:2607.13017, 2026.

C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023. DOI: 10.15607/rss.2023.xix.026.

Y. Deng, G. Liu, and K. Jia. DexWorldModel: Causal latent world modeling towards automated learning of embodied tasks. arXiv preprint arXiv:2604.16484, 2026.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Y. Hu, Y. Guo, P. Wang, X. Chen, Y.-J. Wang, J. Zhang, K. Sreenath, C. Lu, and J. Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *ICML*, 2025. arXiv:2412.14803.

J. Jang, S. Ye, Z. Lin, et al. DreamGen: Unlocking generalization in robot learning through video world models. In *CoRL*, 2025. arXiv:2505.12705.

A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. Srirama, L. Chen, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *RSS*, 2024. DOI: 10.15607/rss.2024.xx.120.

M.-J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, et al. Openvla: An open-source vision-language-action model. arXiv preprint arXiv:2406.09246, 2024.

L. Li, Q. Zhang, Y. Luo, S. Yang, R. Wang, F. Han, M. Yu, Z. Gao, N. Xue, X. Zhu, Y. Shen, and Y. Xu. Causal world modeling for robot control. arXiv preprint arXiv:2601.21998, 2026.

Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *ICLR*, 2023.

Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, and C. Xu. Octo: An open-source generalist robot policy. arXiv preprint arXiv:2405.12213, 2024.

J. Tang, Y. Sun, Y. Zhao, S. Yang, Y. Lin, Z. Zhang, J. Hou, Y. Lu, Z. Liu, and S. Han. VLASH: Real-time VLAs via future-state-aware asynchronous inference. arXiv preprint arXiv:2512.01031, 2025.

H. R. Walke, K. Black, T. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. He, V. Myers, M.-J. Kim, and M. Du, et al. BridgeData v2: A dataset for robot learning at scale. In *CoRL*, pages 1723–1736, 2023. arXiv:2308.12952.

T. Xiao, E. Jang, D. Kalashnikov, S. Levine, J. Ibarz, K. Hausman, and A. Herzog. Thinking while moving: Deep reinforcement learning with concurrent control. arXiv preprint arXiv:2004.06089, 2020.

Z. Yang, Q. Wang, Y. Wang, X. Guo, B. Yu, S. Liu, J. Xu, H. Dong, and M. Li. Jetson-PI: Towards onboard real-time robot control via foresight-aligned asynchronous inference. In *CoRL*, 2026. arXiv:2607.12659.

Q. Zhang, et al. Native video-action pretraining for generalizable robot control. arXiv preprint arXiv:2607.08639, 2026.

T. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *RSS*, 2023.

C. Zhu, R. Yu, S. Feng, B. Burchfiel, P. Shah, and A. Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. In *RSS*, 2025.

A APPENDIX

A.1 ALGORITHMIC DETAILS

The main paper gives the general handoff and fallback protocol. This section records only model-specific implementation details that are needed to interpret the evaluated configurations.

FlowLap on FlowWAM. The FlowWAM action tensor has shape $[33, 14]$: one normalized current-qpos anchor followed by 32 executable joint-qpos rows. During each denoising step, the anchor is restored. All FlowWAM configurations share the same 25-control execution window; FlowLap additionally launches its speculative candidate with a 16-control execution horizon, so the 16 consumed controls are removed once after a candidate is consumed, and zero padding is not treated as an executable action row.

FlowLap on LingBot-VA. The reduced LingBot-VA configuration uses two video steps and five action steps. Its implementation applies the action projection and uncertainty checks described in Sec. 3.2 to position, rotation, acceleration, gripper, context, and temporal consistency. The uncalibrated configuration uses the same candidate submission and handoff protocol as the calibrated configuration.

RTC baseline with context handoff. The RTC baseline is rehosted on the boundary-refreshed flow-matching video-action interface. It runs the joint rollout, protects the prefix, completes the suffix, and performs the same boundary context handoff from real observations as FlowLap; it retains RTC's published action-chunk protocol and does not add the FlowLap handoff validity gate. The baseline and FlowLap therefore share the joint rollout and context handoff and differ in candidate validation, which isolates that component. The base LingBot-VA model also describes a native asynchronous prediction scheme, but that implementation is not part of the public release, so it cannot be run as a baseline and the RTC baseline with matched context handoff is used instead.

A.2 HANDOFF GATE

The gate applies one decision template across backbones; only the action partition and the numeric thresholds are instantiated per embodiment. Let the committed boundary control be b^- and let the shifted candidate suffix be $(\tilde{b}^0, \ldots, \tilde{b}^{n-1})$. Partition the action coordinates into semantic groups $g \in G$; our instantiations use translation, rotation, and gripper. For group g, the per-step continuity residual is

$$\rho_g = \left(\frac{1}{n} \sum_{i=0}^{n-1} \left\| \Pi_g(\tilde{b}^i) - \Pi_g(\hat{b}^{i-1}) \right\|^2 \right)^{1/2}, \qquad \hat{b}^{-1} = b^-, \tag{15}$$

where Π_g selects the group coordinates in a group-specific metric: Euclidean distance for translation and gripper, and the shortest-angle geodesic for rotation. A candidate is committed only if it passes the identity, timing, shape, and finiteness checks and $\rho_g \le \tau_g$ for every group. Before this measurement the projection step clips per-step and per-step-acceleration changes to (ℓ_g, a_g) so that the residual is measured on a physically plausible suffix rather than on the raw network output. Table 6 lists the instantiated groups and thresholds; the same template applies to an unseen embodiment once its group partition and limits are calibrated from executed baseline deltas.

Table 6: Instantiated handoff-gate groups and thresholds, calibrated from executed baseline action deltas. Translation uses metric units, rotation is in radians, and the gripper continuity metric is a sign-mismatch rate.

Group g	Step limit ℓ_g	Accel. limit a_g	Budget τ_g	Metric
Translation	0.08	0.08	0.30	Euclidean RMS
Rotation	0.30	0.35	0.60	Geodesic RMS (rad)
Gripper	0.05 / 0.15 / 0.25	–	0.50	Sign-mismatch rate

Scheduling checks are shared across groups: a candidate is rejected if it is not ready within the boundary budget, if its episode, target index, or temporal position disagrees with the scheduled

handoff, if its stale horizon exceeds eight controls, or if the uncertainty gate reports a latent mismatch above 0.35 or more than one consecutive no-progress interval. The three gripper limits denote deadband, hysteresis, and transition thresholds. All of these checks are evaluated from state already available during the current execution window, so they add no work after the boundary.

A.3 Speculative joint rollout

A.3.1 Setup and assumptions

Let $\mathcal{A}$ and $\mathcal{V}$ denote the action and visual-output spaces. For a real context C_t and an executor-owned prefix P_t, write the constrained joint rollout as

$$(\widetilde{V}_{t+1}, \widetilde{B}_{t+1}) \sim p_\theta(\cdot \mid C_t, P_t), \qquad [\widetilde{B}_{t+1}]_{0:E} = P_t. \tag{16}$$

The second condition is the mathematical form of the prefix mask in Eq. 7. Let $G_t \in \{0, 1\}$ denote the result of the handoff gate, where $G_t = 1$ means that the candidate is accepted. Here $[\cdot]_{0:E}$ denotes the half-open row range $0, \ldots, E-1$. The following assumptions describe the interface contract, rather than the internal architecture of M_θ.

Assumption A.1 (Executor ownership)**.** *Once P_t is sent to the executor, it is immutable until its scheduled execution completes. Any speculative computation may read P_t but cannot modify it.*

Assumption A.2 (Real-context update)**.** *The persistent context is updated by $C_{t+1} = U(C_t, o_{t+1}, P_t)$, where o_{t+1} is the real observation. The predicted visual output $\widetilde{V}_{t+1}$ is not an input to U.*

Assumption A.3 (Admissible handoff)**.** *The gate can accept a candidate only when its identity, temporal position, shape, finiteness, and shifted-prefix continuity are valid. When any check fails, $G_t = 0$ and the next action is obtained from a fresh rollout $[M_\theta(C_{t+1})]_{\text{action}}$.*

Theorem A.4 (Execution and context invariants)**.** *Under Assumptions A.1–A.3, for every handoff t the speculative joint rollout has the following properties:*

1. *the executor's already committed prefix is unchanged;*
2. *C_t depends only on the initial context, real observations, and executed controls, and is independent of speculative visual outputs;*
3. *an accepted action is the shifted suffix of one joint sample, while an inadmissible sample is replaced by a rollout from the current real context.*

Proof. We proceed by induction over handoffs. At $t = 0$, the initial context is real and the first prefix is sent to the executor once, so all three statements hold. Suppose they hold through handoff t.

Step 1: Prefix ownership. By Assumption A.1, the controls in P_t are already owned by the executor. Equation 16 and the mask in Eq. 7 force the first E rows of the in-flight action to equal P_t at every generation step. The shift operator therefore removes exactly those rows:

$$\mathcal{S}_E(\widetilde{B}_{t+1}) = (\widetilde{B}_{t+1}[E], \ldots, \widetilde{B}_{t+1}[H-1]).$$

No operation performed by the speculative request can alter the controls already being executed.

Step 2: Real-context causality. At the handoff, Assumption A.2 gives $C_{t+1} = U(C_t, o_{t+1}, P_t)$. Thus the new context is a function of the previous real history and the observation actually returned by the environment. Since $\widetilde{V}_{t+1}$ is excluded from this update, the induction hypothesis extends to C_{t+1}.

Step 3: Candidate selection. If $G_t = 1$, Eq. 10 selects $\mathcal{S}_E(\widetilde{B}_{t+1})$, which is the suffix of the same joint sample as $\widetilde{V}_{t+1}$. If $G_t = 0$, Assumption A.3 selects $[M_\theta(C_{t+1})]_{\text{action}}$, so the failed candidate cannot affect the next action or the next context. The three statements therefore hold at $t+1$, completing the induction. □

A.3.2 LATENCY COVERAGE

Let X_t be the execution window available after the speculative request is launched, and let R_t be the corresponding critical-path time until an admissible candidate can be selected. R_t includes rollout, metadata and validity checks, and any real-context work that must precede selection. Let F_t denote the wait incurred by the fresh-context fallback when the candidate is rejected.

Theorem A.5 (Execution-window coverage)**.** *If the candidate is admissible and the runtime introduces no additional serialization, speculative joint rollout has transition wait*

$$W_t^{\mathrm{SJ}} = [R_t - X_t]_+. \tag{17}$$

For the matched synchronous controller, $W_t^{\mathrm{sync}} = R_t$. Hence

$$W_t^{\mathrm{sync}} - W_t^{\mathrm{SJ}} = \min\{R_t, X_t\}, \tag{18}$$

and the candidate is wait-free whenever $R_t \le X_t$.

Proof. Launch the speculative request at time 0. It becomes available at time R_t, while execution finishes at time X_t. The next action can be selected at the later of these two times, so the residual wait is $\max(0, R_t - X_t) = [R_t - X_t]_+$, proving Eq. 17. The synchronous controller starts the same work only after execution and therefore exposes all R_t units of preparation time. Subtracting the two expressions gives

$$R_t - [R_t - X_t]_+ = \min\{R_t, X_t\},$$

which proves Eq. 18 and the wait-free condition.

When the gate rejects the candidate, the same reasoning is applied to the fresh request after the real-context update. In particular, the per-step wait obeys the runtime-level decomposition

$$W_t \le G_t[R_t - X_t]_+ + (1 - G_t)F_t, \tag{19}$$

so the latency guarantee is obtained on accepted candidates without requiring the controller to consume a stale or malformed result. □

A.3.3 JOINT-OUTPUT PRESERVATION

Define the deterministic post-processing map $\Pi_E : \mathcal{V} \times \mathcal{A}^H \to \mathcal{V} \times \mathcal{A}^{H-E}$ by $\Pi_E(V, B) = (V, \mathcal{S}_E(B))$. The joint rollout is consumed through this map, rather than by drawing a visual continuation and an action suffix from separate requests.

Theorem A.6 (Preservation of the joint rollout)**.** *For every bounded measurable test function g and fixed (C_t, P_t), the candidate before gating is the pushforward of the joint rollout. Moreover, if (V, B') denotes the analytical pair associated with an accepted action, then conditional on acceptance it satisfies*

$$\mathbb{E}\big[g(V, B') \mid C_t, P_t, G_t = 1\big] = \mathbb{E}_{(V,B)\sim p_\theta(\cdot|C_t,P_t,G_t=1)}\big[g(V, \mathcal{S}_E(B))\big]. \tag{20}$$

Thus gating may select a subset of joint samples, but it never introduces an independently sampled cross-modal component.

Proof. By Eq. 16, the in-flight result is a random variable (V, B) with conditional law $p_\theta(\cdot \mid C_t, P_t)$. Before gating, applying the deterministic map Π_E gives the pushforward $(\Pi_E)_{\#}p_\theta$. Conditional on $G_t = 1$, the accepted result is instead the pushforward of the restricted law $p_\theta(\cdot \mid C_t, P_t, G_t = 1)$. The definition of a pushforward measure gives

$$\int g \,\mathrm{d}((\Pi_E)_{\#}p_\theta(\cdot \mid G_t = 1)) = \int g(\Pi_E(V, B)) \,\mathrm{d}p_\theta(V, B \mid C_t, P_t, G_t = 1),$$

which is Eq. 20. The prefix constraint ensures that $\mathcal{S}_E$ removes executor-owned controls rather than replacing them; the visual and action outputs still originate from the same conditional sample. □

A.4 EXECUTION-STATE CALIBRATION

A.4.1 SETUP AND ASSUMPTIONS

Fix a real context C and a launch state s_t. Let $s^\star_{t+K_t}$ be the state reached by executing the committed controls and let $\widehat{s}_{t+K_t} = \Phi(s_t, A_t^{\text{delay}})$ be its propagated estimate. Write

$$e = \varphi(s_t), \qquad e^{\star+} = \varphi(s^\star_{t+K_t}), \qquad \widehat{e}^{+} = \varphi(\widehat{s}_{t+K_t}). \tag{21}$$

Let $a^*(C, e')$ be the ideal action target when the execution-time feature is e'. The calibrated action predictor is $f_\psi(C, e')$.

Assumption A.7 (Lipschitz action target)**.** *There is an $L_a \geq 0$ such that, for fixed C,*

$$\|a^*(C, e_1) - a^*(C, e_2)\| \leq L_a \|e_1 - e_2\|. \tag{22}$$

Assumption A.8 (Propagation accuracy)**.** *The valid calibrated operating range satisfies $\|\widehat{e}^{+} - e^{\star+}\| \leq \epsilon_\Phi$.*

Assumption A.9 (Predictor approximation)**.** *On the relevant launch and propagated features, the calibrated predictor satisfies $\|f_\psi(C, e') - a^*(C, e')\| \leq \epsilon_\theta$. The uncalibrated predictor obeys the same approximation bound at the launch feature e.*

Theorem A.10 (Execution-state calibration error)**.** *Under Assumptions A.7–A.9, the calibrated action obeys*

$$\|f_\psi(C, \widehat{e}^{+}) - a^*(C, e^{\star+})\| \leq \epsilon_\theta + L_a \epsilon_\Phi. \tag{23}$$

If an uncalibrated predictor uses the launch feature e with the same approximation error, then

$$\|f_\theta(C, e) - a^*(C, e^{\star+})\| \leq \epsilon_\theta + L_a \|e - e^{\star+}\|. \tag{24}$$

Proof. We first analyze the calibrated predictor. Add and subtract $a^*(C, \widehat{e}^{+})$:

$$\begin{aligned}\|f_\psi(C, \widehat{e}^{+}) - a^*(C, e^{\star+})\| \leq{}& \|f_\psi(C, \widehat{e}^{+}) - a^*(C, \widehat{e}^{+})\| \\ &+ \|a^*(C, \widehat{e}^{+}) - a^*(C, e^{\star+})\|.\end{aligned}$$

The first term is at most ϵ_θ by Assumption A.9. The second is at most $L_a\|\widehat{e}^{+} - e^{\star+}\|$ by Assumption A.7. Applying Assumption A.8 yields

$$\|f_\psi(C, \widehat{e}^{+}) - a^*(C, e^{\star+})\| \leq \epsilon_\theta + L_a \epsilon_\Phi,$$

which proves Eq. 23.

For the uncalibrated predictor, add and subtract $a^*(C, e)$ instead. The approximation term is again at most ϵ_θ, and the Lipschitz term is $L_a\|e - e^{\star+}\|$. This gives Eq. 24. □

Corollary A.11 (When calibration tightens the error guarantee)**.** *Let $D = \|e - e^{\star+}\|$. Under the assumptions of Theorem A.10, if $\epsilon_\Phi < D$, the calibrated upper bound is smaller by $L_a(D - \epsilon_\Phi)$. More generally, the same comparison holds in expectation whenever $\mathbb{E}[\|\widehat{e}^{+} - e^{\star+}\|] < \mathbb{E}[D]$.*

Proof. Subtract the right-hand side of Eq. 23 from that of Eq. 24. The approximation errors cancel, leaving $L_a(D - \epsilon_\Phi) > 0$. Taking expectations gives the second statement. □

The result identifies the mechanism measured by calibration: it replaces the launch-to-execution displacement in the action-target bound with the error of the execution-state predictor. The condition is injected only into action features, so this improvement is compatible with the real-context invariant of Theorem A.4; it does not require a predicted visual frame to update the persistent context.

A.5 CALIBRATION IMPLEMENTATION

The calibrated LingBot-VA configuration uses a 52-dimensional normalized execution-state condition, a 256-unit residual adapter, and a maximum delay and prediction horizon of 32 controls. The adapter was trained with zero-, 16-, and 32-control delays, sampled with weights 0.2, 0.5, and 0.3, respectively. The condition is derived from the committed action and current robot state, rather than from a predicted visual frame.

For a nonzero delay k, the runtime takes the current two-arm relative end-effector state and the state proxy obtained from the tail of the committed action. Each arm contributes normalized position, quaternion, and gripper features. The condition concatenates current-state features, future-state features, their normalized position/rotation/gripper deltas, and six temporal features: normalized delay, delay in seconds, execution-to-prediction horizon ratio, committed-prefix ratio, runtime phase, and suffix-motion RMS. A zero-delay request is represented by an exact zero vector. The adapter is therefore an identity path when calibration is disabled:

$$c_k = \begin{cases} \mathbf{0}, & k = 0, \\ [e, \widehat{e}^{+}, \Delta e, \tau(k), m], & 1 \le k \le 32. \end{cases} \tag{25}$$

The condition is broadcast to the action tokens through a zero-initialized residual path with runtime scale 0.25. It is applied to every LingBot-VA task in the calibrated configuration; the uncalibrated configuration sets the condition to zero. Within each model, all configurations share the same seed, task list, episode allocation, execution window, handoff gate, and fallback logic. The LingBot-VA uncalibrated and calibrated configurations also use the same overlap schedule, allowing up to eight consecutive pending candidates; they differ in the enabled execution-state condition and, for the calibrated run, in the deterministic model-RNG and attention-backend settings used for reproducibility. The resulting differences in transition and request counts (2,140 vs. 2,497 transitions and 2,290 vs. 2,647 requests) are therefore a natural consequence of the trajectories induced by the condition, not of a different evaluation protocol.

A.6 Experimental protocol

All nine evaluation configurations use the RoboTwin 2.0 clean-scene configuration, all 50 tasks, three episodes per task, easy difficulty, and the same fixed random seed (0) for task and episode initialization, so every task is evaluated under identical stochastic conditions within and across configurations. All configurations use the shared fixed evaluation protocol, including the same task list, episode allocation, and handoff/fallback logic. Where recorded, the evaluation uses unseen instructions. The LingBot-VA RTC record predates the explicit seed field in the released logs but was executed under the same fixed protocol. Thus every configuration contains 150 episodes. The reference configurations retain the released inference budgets, while the efficient synchronous controls match the reduced budgets used by FlowLap but do not overlap inference with execution.

To bound test time, the ablation runs additionally impose a wall-clock timeout equal to twice the matched baseline's task runtime. This guard only stops a task that would otherwise run indefinitely after a possible model-side inference error, so it does not change task outcomes. A task terminated this way yields no execution records and is scored as $0/3$; the reported transition-wait statistics are pooled only over completed executions.

Table 7: Evaluation configurations. FlowLap and RTC are separate configurations with the listed video/action step budgets.

Model	Configuration	Video/action steps	Configuration details
FlowWAM	Reference	25 / 50	Released synchronous configuration
FlowWAM	Efficient sync.	4 / 4	Reduced-work synchronous control
FlowWAM	RTC	4 / 4	Context-handoff asynchronous baseline
FlowWAM	FlowLap	4 / 4	Speculative joint rollout; 16-control horizon in the 25-control window
LingBot-VA	Reference	3 / 10	Released synchronous configuration
LingBot-VA	Efficient sync.	2 / 5	Reduced-work synchronous control
LingBot-VA	RTC	2 / 5	Context-handoff asynchronous baseline
LingBot-VA	FlowLap	2 / 5	Speculative joint rollout; prefix-preserving handoff
LingBot-VA	FlowLap + cal.	2 / 5	FlowLap with execution-state calibration

For FlowWAM, every reduced configuration uses 33 action frames, 9 video frames, a 14-dimensional absolute joint-qpos action, and the same 25-control execution window; the configurations differ only in how that window is used, with FlowLap launching a 16-control speculative horizon. For LingBot-VA, the reduced configurations use two video steps and five action steps under the same window. The calibrated configuration uses the adapter at runtime scale 0.25 under the same runtime protocol as the uncalibrated configuration. Base checkpoints and simulator assets remain external dependencies.

Timing method and platform. All timestamps are host wall-clock readings from `time.perf_counter()`. The boundary time t_{exec} is taken at the end of a chunk's environment step and the ready time t_{ready} when the model response becomes available, so the reported wait is $W_t = \max(0, t_{\text{ready}} - t_{\text{exec}})$. Model-side phase times synchronize the CUDA device before reading the timer. Evaluations run on a single NVIDIA H100 GPU in BF16, with the simulator and policy server on the same host.

Scope and limitations. All results are simulation-only on the RoboTwin 2.0 clean, easy configuration. Each task uses three episodes under a single seed, so per-task success is quantized to $0/33.3/66.7/100\%$ and small aggregate differences are not tested for significance. The calibration study is restricted to LingBot-VA, and no real-robot or external-disturbance experiment is included; the reported latency gains therefore characterize boundary stalls rather than measured responsiveness to new events.

Compute cost. Overlap changes when work runs, not how much work the backbone performs. Summing per-request model time over a run, FlowLap performs about 1.75×10^3 s of model-side work versus 0.80×10^3 s for the efficient synchronous control on FlowWAM ($+118\%$), and 4.74×10^3 s versus 1.09×10^3 s on LingBot-VA ($+335\%$); calibrated LingBot-VA uses 3.67×10^3 s ($+236\%$). Per episode this is 11.6 s versus 5.3 s (FlowWAM) and 31.6 s versus 7.3 s (LingBot-VA). Peak allocated CUDA memory is essentially unchanged at about 23.7 GiB for FlowWAM and 31.6 GiB for LingBot-VA. The benefit is therefore a placement benefit—hiding a larger amount of computation inside the execution window—rather than a reduction in GPU work.

Metric extraction. For each transition, boundary wait is $\max(0, t_{\text{ready}} - t_{\text{exec}})$. The task-level tables report the arithmetic mean, empirical median, and empirical 90th percentile over the recorded transitions for each task. The first action chunk of each episode has no preceding transition and is excluded. The final row pools all transitions from all 50 tasks; success is reported as successful episodes out of 150.

A.7 Handoff gate statistics

Table 8 reports the fail-closed behavior of the handoff gate. For FlowLap we count the handoff decisions, how many speculative candidates were accepted and consumed, and how many were rejected or unavailable and replaced by a fresh-context rollout; the RTC baseline has no gate, so its "rejected" count is the set its native continuation declined. The rejection rate is the fraction of decisions at which the runtime declined a candidate that risked a timing, structural, or continuity violation; because the gate is fail-closed, this is the directly observable guard against accepting a bad candidate. Rejected candidates include late, stale, malformed, and continuity-violating suffixes, so the rate is an upper bound on the validity-only false-accept fraction. The no-gate ablation in the main text quantifies the cost of removing this guard: accepting every candidate lowers FlowWAM SR from 91.3% to 80.7% and LingBot-VA SR from 86.0% to 82.0%. FlowWAM rejects very few candidates, whereas LingBot-VA rejects 17.2% and calibration reduces that to 7.4%, consistent with the stronger boundary-continuity and stale-horizon signals of the autoregressive context rather than with a different decision rule.

Table 8: Candidate decisions over all recorded transitions. For FlowLap rows, "Rejected" counts candidates that were unavailable or failed the fail-closed gate and were replaced by a fresh-context rollout; for the RTC baseline, which has no gate, it counts candidates its native continuation declined.

Configuration	Decisions	Accepted	Rejected	Accept (%)	Reject (%)
FlowWAM RTC + context handoff	2,781	2,712	69	97.5	2.5
FlowWAM FlowLap	2,199	2,194	5	99.8	0.2
LingBot-VA RTC + context handoff	2,478	2,052	426	82.8	17.2
LingBot-VA FlowLap	2,497	2,068	429	82.8	17.2
LingBot-VA FlowLap + cal.	2,140	1,982	158	92.6	7.4

A.8 COMPLETE TASK-LEVEL OUTCOMES

The following tables contain all 50 tasks, rather than only the ten representative tasks shown in the main paper. Each configuration cell reports success out of three, followed by transition-wait Mean and p50 in milliseconds. Task-level p90 values are omitted here, while the primary results table reports p90 for the complete evaluation. The final row pools all recorded transitions from all 50 tasks. Table 11 reports the same breakdown for the ablation configurations; executions terminated by the runtime guard of Appendix A.6 are scored as $0/3$ and carry no wait entry.

Table 9: Complete FlowWAM task-level success and transition-wait statistics.

	Reference			Efficient sync.			RTC			FlowLap		
Task	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50
Adjust Bottle	3/3	12506.6	11576.9	3/3	825.9	797.7	3/3	70.4	0.0	3/3	0.0	0.0
Beat Block Hammer	3/3	12271.8	12341.3	3/3	952.8	954.7	3/3	0.0	0.0	3/3	0.0	0.0
Blocks Ranking RGB	3/3	11682.5	11792.9	3/3	763.4	747.5	3/3	5.7	0.0	3/3	50.4	0.0
Blocks Ranking Size	3/3	12073.4	12107.1	3/3	771.4	758.6	1/3	78.0	0.0	3/3	32.9	0.0
Click Alarm Clock	3/3	10652.2	10306.6	3/3	918.5	917.2	3/3	0.0	0.0	3/3	106.7	0.0
Click Bell	3/3	10865.1	10912.5	2/3	792.7	724.8	3/3	0.0	0.0	3/3	147.1	0.0
Dump Bin Big Bin	3/3	10613.1	10202.0	3/3	864.1	822.0	3/3	0.0	0.0	3/3	0.0	0.0
Grab Roller	3/3	11096.6	11294.6	3/3	1026.7	1044.8	3/3	0.0	0.0	3/3	35.3	0.0
Handover Block	3/3	11263.1	11197.2	2/3	771.8	746.7	2/3	0.0	0.0	3/3	0.0	0.0
Handover Mic	3/3	11283.9	11338.7	3/3	829.2	812.6	3/3	0.0	0.0	3/3	10.8	0.0
Hanging Mug	1/3	11297.8	11530.7	1/3	835.8	798.9	3/3	0.0	0.0	2/3	0.0	0.0
Lift Pot	3/3	10897.0	10785.1	3/3	1132.1	1187.3	3/3	0.0	0.0	3/3	0.0	0.0
Move Can Pot	3/3	12011.3	12020.8	3/3	836.4	911.4	3/3	0.0	0.0	3/3	0.0	0.0
Move Pillbottle Pad	3/3	11171.8	11098.1	3/3	888.7	881.6	3/3	153.3	0.0	3/3	45.9	0.0
Move Playingcard Away	3/3	11556.6	11712.5	3/3	920.0	940.0	3/3	14.0	0.0	3/3	47.5	0.0
Move Stapler Pad	3/3	16792.2	12575.4	2/3	3393.1	877.7	3/3	0.0	0.0	2/3	0.0	0.0
Open Laptop	3/3	7146.4	7009.4	3/3	984.0	984.2	3/3	0.0	0.0	3/3	0.0	0.0
Open Microwave	2/3	11281.5	12276.7	2/3	775.3	758.3	1/3	50.2	0.0	2/3	216.5	355.8
Pick Diverse Bottles	3/3	10860.5	10801.2	3/3	923.4	947.2	3/3	0.0	0.0	2/3	0.0	0.0
Pick Dual Bottles	3/3	12152.1	12038.1	3/3	959.0	994.2	3/3	0.0	0.0	3/3	0.0	0.0
Place A2B Left	2/3	12769.9	12730.6	3/3	860.9	895.8	3/3	11.3	0.0	3/3	25.5	0.0
Place A2B Right	3/3	13233.2	13413.2	3/3	828.1	774.0	2/3	0.0	0.0	3/3	34.3	0.0
Place Bread Basket	3/3	13228.9	13232.6	3/3	920.8	943.8	2/3	0.0	0.0	3/3	18.5	0.0
Place Bread Skillet	3/3	12693.0	12690.9	3/3	933.8	953.8	3/3	0.0	0.0	3/3	0.0	0.0
Place Burger Fries	3/3	13403.4	13651.3	3/3	877.1	852.6	3/3	0.0	0.0	3/3	0.0	0.0
Place Can Basket	2/3	12317.9	13053.3	1/3	792.4	787.5	1/3	0.0	0.0	2/3	0.0	0.0
Place Cans Plasticbox	3/3	10867.0	11005.5	3/3	891.2	850.6	3/3	0.0	0.0	3/3	0.0	0.0
Place Container Plate	3/3	11343.2	10747.0	3/3	887.2	936.5	3/3	0.0	0.0	3/3	0.0	0.0
Place Dual Shoes	3/3	10884.6	11074.9	2/3	911.2	899.5	3/3	0.0	0.0	2/3	0.0	0.0
Place Empty Cup	3/3	13371.0	13326.7	3/3	829.2	841.6	3/3	0.0	0.0	3/3	0.0	0.0
Place Fan	3/3	13530.9	13109.2	3/3	816.1	768.8	3/3	40.7	0.0	3/3	16.5	0.0
Place Mouse Pad	3/3	13989.3	14328.8	3/3	832.7	856.4	3/3	0.0	0.0	3/3	16.0	0.0
Place Object Basket	2/3	12986.9	12845.8	3/3	869.3	872.4	2/3	0.0	0.0	2/3	0.0	0.0
Place Object Scale	3/3	13251.8	13213.5	3/3	904.3	930.8	3/3	0.0	0.0	3/3	14.3	0.0
Place Object Stand	3/3	14098.7	14047.3	3/3	880.3	914.6	3/3	0.0	0.0	3/3	15.7	0.0
Place Phone Stand	3/3	14610.6	14805.7	3/3	926.0	941.7	3/3	0.0	0.0	3/3	17.9	0.0
Place Shoe	3/3	13611.2	13352.4	3/3	889.8	924.0	3/3	0.0	0.0	3/3	0.0	0.0
Press Stapler	3/3	14584.9	14630.9	3/3	888.0	882.5	3/3	0.0	0.0	3/3	0.0	0.0
Put Bottles Dustbin	3/3	13331.6	13648.3	3/3	826.3	804.2	2/3	39.2	0.0	1/3	57.2	0.0
Put Object Cabinet	2/3	12577.4	13212.3	3/3	945.6	945.4	2/3	6.9	0.0	2/3	0.0	0.0
Rotate QR Code	2/3	13993.8	13874.9	3/3	839.8	892.3	3/3	0.0	0.0	3/3	40.4	0.0
Scan Object	2/3	13073.5	12988.6	3/3	941.8	974.5	3/3	0.0	0.0	3/3	18.2	0.0
Shake Bottle	3/3	13687.8	13692.8	3/3	927.8	914.9	3/3	217.5	0.0	3/3	140.4	0.0
Shake Bottle Horizontally	3/3	13131.9	13496.3	3/3	931.0	924.5	3/3	0.0	0.0	3/3	76.5	0.0
Stack Blocks Three	3/3	14565.5	14409.5	3/3	767.7	747.7	2/3	61.8	0.0	3/3	65.3	0.0
Stack Blocks Two	3/3	14160.5	14243.8	3/3	770.4	745.6	3/3	0.0	0.0	3/3	25.7	0.0
Stack Bowls Three	3/3	14168.7	14875.3	3/3	852.8	827.4	1/3	7.1	0.0	2/3	33.8	0.0
Stack Bowls Two	3/3	13930.3	14076.6	3/3	821.7	785.6	3/3	65.2	0.0	3/3	24.7	0.0
Stamp Seal	2/3	12936.0	13724.6	3/3	821.1	853.8	3/3	274.4	0.0	3/3	14.0	0.0
Turn Switch	2/3	14632.1	15437.9	2/3	836.9	806.0	1/3	204.1	0.0	1/3	155.6	0.0
All tasks	**139/150**	12602.7	12786.4	**140/150**	896.6	816.8	**133/150**	29.8	0.0	**137/150**	39.1	0.0

Table 10: Complete LingBot-VA task-level success and transition-wait statistics.

	Reference			Efficient sync.			RTC			FlowLap			FlowLap+ cal.		
Task	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50
Adjust Bottle	3/3	1950.9	2081.6	3/3	928.8	934.5	3/3	226.0	0.0	3/3	262.4	0.0	3/3	0.0	0.0
Beat Block Hammer	3/3	1910.9	2072.1	3/3	959.4	939.2	3/3	151.4	0.0	3/3	204.9	0.0	3/3	0.0	0.0
Blocks Ranking RGB	3/3	1827.9	2072.5	3/3	623.6	544.6	3/3	172.9	0.0	3/3	152.9	0.0	2/3	82.0	0.0
Blocks Ranking Size	3/3	1480.4	1259.7	2/3	729.4	680.2	3/3	274.1	0.0	3/3	199.0	0.0	3/3	75.0	0.0
Click Alarm Clock	3/3	2228.5	2350.4	3/3	1080.0	1124.7	3/3	0.0	0.0	3/3	0.0	0.0	3/3	0.0	0.0
Click Bell	3/3	2121.8	2087.3	3/3	936.5	930.4	3/3	0.0	0.0	3/3	0.0	0.0	3/3	0.0	0.0
Dump Bin Big Bin	3/3	1497.2	1261.8	3/3	669.8	583.6	3/3	238.7	0.0	3/3	272.4	0.0	3/3	30.8	0.0
Grab Roller	3/3	2186.6	2095.5	3/3	1101.9	1099.9	3/3	240.9	0.0	3/3	248.3	0.0	3/3	0.0	0.0
Handover Block	3/3	1411.5	1245.6	3/3	861.2	932.4	3/3	291.5	0.0	3/3	256.2	0.0	3/3	65.3	0.0
Handover Mic	3/3	1605.6	1335.7	2/3	712.9	565.2	3/3	206.7	0.0	3/3	149.2	0.0	3/3	84.0	0.0
Hanging Mug	1/3	1541.6	1356.3	0/3	754.0	569.0	0/3	341.7	0.0	0/3	340.5	0.0	1/3	139.7	0.0
Lift Pot	3/3	1737.2	1608.2	3/3	965.0	944.0	3/3	316.6	0.0	3/3	254.6	0.0	3/3	0.0	0.0

Table 10: Complete LingBot-VA task-level statistics (continued).

	Reference			Efficient sync.			RTC			FlowLap			FlowLap+ cal.		
Task	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50
Move Can Pot	3/3	2226.5	2390.9	3/3	661.3	545.3	3/3	123.0	0.0	2/3	138.3	0.0	2/3	42.9	0.0
Move Pillbottle Pad	3/3	1996.6	2077.0	3/3	944.2	946.5	3/3	198.4	0.0	3/3	189.2	0.0	3/3	33.3	0.0
Move Playingcard Away	3/3	2074.5	2086.8	3/3	804.1	877.2	3/3	220.5	0.0	3/3	185.4	0.0	3/3	46.6	0.0
Move Stapler Pad	1/3	2599.6	2141.1	2/3	1105.6	1201.8	1/3	284.5	0.0	2/3	221.3	0.0	1/3	90.3	0.0
Open Laptop	3/3	1556.6	1281.6	3/3	905.6	914.0	3/3	142.2	0.0	3/3	160.2	0.0	3/3	98.9	0.0
Open Microwave	3/3	1487.7	1286.6	1/3	765.4	735.7	0/3	321.1	0.0	0/3	297.5	0.0	3/3	92.3	0.0
Pick Diverse Bottles	3/3	2061.5	2084.5	3/3	1003.9	1012.3	3/3	289.9	0.0	3/3	274.9	0.0	3/3	0.0	0.0
Pick Dual Bottles	3/3	1657.5	1530.0	3/3	971.2	955.8	2/3	232.6	0.0	3/3	277.7	0.0	3/3	0.0	0.0
Place A2B Left	3/3	2033.4	2073.2	2/3	729.8	579.9	3/3	132.8	0.0	3/3	134.2	0.0	3/3	41.6	0.0
Place A2B Right	3/3	1709.5	1924.4	3/3	932.6	941.0	3/3	155.8	0.0	3/3	142.4	0.0	2/3	32.1	0.0
Place Bread Basket	3/3	1520.2	1270.9	2/3	889.0	936.3	3/3	204.6	0.0	3/3	184.2	0.0	3/3	52.9	0.0
Place Bread Skillet	3/3	1828.9	2011.5	2/3	850.2	921.3	2/3	233.6	0.0	1/3	182.5	0.0	3/3	0.0	0.0
Place Burger Fries	3/3	1961.6	2094.6	3/3	831.9	915.5	3/3	269.5	0.0	3/3	221.8	0.0	3/3	52.2	0.0
Place Can Basket	3/3	1389.7	1251.6	3/3	946.6	1044.2	2/3	224.5	0.0	2/3	144.8	0.0	2/3	71.9	0.0
Place Cans Plasticbox	3/3	1792.5	1813.6	3/3	839.2	932.2	3/3	222.6	0.0	3/3	190.4	0.0	3/3	75.9	0.0
Place Container Plate	3/3	1720.3	1940.1	3/3	786.2	871.5	3/3	212.4	0.0	3/3	212.3	0.0	3/3	0.0	0.0
Place Dual Shoes	3/3	1893.1	2057.7	2/3	804.4	757.1	3/3	140.4	0.0	3/3	144.3	0.0	3/3	72.7	0.0
Place Empty Cup	3/3	1570.5	1249.1	3/3	1000.8	1012.5	3/3	151.8	0.0	3/3	112.4	0.0	3/3	123.9	0.0
Place Fan	3/3	1503.9	1294.9	3/3	862.0	931.4	2/3	272.7	0.0	3/3	234.8	0.0	3/3	89.4	0.0
Place Mouse Pad	3/3	1734.7	1993.1	3/3	1176.1	953.4	1/3	260.3	0.0	2/3	234.7	0.0	2/3	104.1	0.0
Place Object Basket	3/3	1597.1	1357.1	3/3	872.7	915.7	2/3	238.6	0.0	3/3	169.9	0.0	3/3	64.8	0.0
Place Object Scale	3/3	1909.5	2057.4	3/3	1016.0	1075.7	2/3	163.7	0.0	1/3	137.7	0.0	3/3	51.7	0.0
Place Object Stand	3/3	2127.2	2109.9	3/3	898.2	912.7	3/3	202.6	0.0	3/3	187.4	0.0	3/3	0.0	0.0
Place Phone Stand	3/3	1498.7	1253.7	3/3	983.8	1018.2	3/3	230.5	0.0	2/3	179.3	0.0	3/3	0.0	0.0
Place Shoe	3/3	1503.3	1275.6	3/3	959.8	943.2	3/3	262.2	0.0	3/3	170.3	0.0	3/3	58.0	0.0
Press Stapler	2/3	2110.3	2073.4	2/3	767.6	781.8	2/3	184.9	0.0	3/3	191.2	0.0	3/3	64.4	0.0
Put Bottles Dustbin	2/3	1601.6	1413.9	3/3	853.8	910.1	1/3	340.5	0.0	1/3	305.9	0.0	2/3	84.6	0.0
Put Object Cabinet	2/3	1533.6	1331.9	3/3	835.6	911.8	2/3	328.1	0.0	1/3	190.1	0.0	2/3	66.4	0.0
Rotate QR Code	3/3	1636.7	1495.7	2/3	870.6	925.2	3/3	194.0	0.0	3/3	200.9	0.0	3/3	96.9	0.0
Scan Object	1/3	1384.4	1301.3	3/3	904.2	919.1	1/3	274.5	0.0	3/3	186.8	0.0	2/3	95.4	0.0
Shake Bottle	3/3	2056.5	2092.1	3/3	948.5	940.6	3/3	275.9	0.0	3/3	282.3	0.0	3/3	0.0	0.0
Shake Bottle Horizontally	3/3	2091.6	2086.2	3/3	927.9	928.0	3/3	252.6	0.0	3/3	241.2	0.0	3/3	0.0	0.0
Stack Blocks Three	3/3	1757.0	1398.8	3/3	696.7	556.3	3/3	301.9	0.0	3/3	240.7	0.0	3/3	103.5	0.0
Stack Blocks Two	3/3	1695.4	1805.3	3/3	861.4	925.5	3/3	234.8	0.0	3/3	136.2	0.0	3/3	80.6	0.0
Stack Bowls Three	3/3	2574.5	2434.2	3/3	946.5	931.2	3/3	456.3	0.0	3/3	454.1	0.0	3/3	143.9	0.0
Stack Bowls Two	2/3	1661.0	1619.2	3/3	830.6	905.5	3/3	160.1	0.0	3/3	148.2	0.0	3/3	77.0	0.0
Stamp Seal	3/3	1541.7	1327.6	3/3	730.5	615.7	3/3	112.8	0.0	3/3	135.6	0.0	3/3	0.0	0.0
Turn Switch	2/3	1649.4	1391.1	2/3	1226.6	943.8	3/3	185.5	0.0	1/3	225.8	0.0	3/3	53.7	0.0
All tasks	139/150	1741.9	1690.6	135/150	839.9	915.7	128/150	264.1	0.0	129/150	232.4	0.0	138/150	71.8	0.0

Table 11: Complete task-level success and transition-wait statistics for the ablation configurations.

	FlowWAM						LingBot-VA								
	Pure overlap			Half launch			Pure overlap			Half launch			Pure + cal.		
Task	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50
Adjust Bottle	3/3	0.0	0.0	2/3	387.7	231.7	3/3	0.0	0.0	3/3	425.9	0.0	3/3	0.0	0.0
Beat Block Hammer	3/3	0.0	0.0	3/3	510.6	270.1	3/3	0.0	0.0	3/3	524.6	392.0	3/3	0.0	0.0
Blocks Ranking RGB	3/3	50.5	0.0	3/3	819.8	1028.8	1/3	0.0	0.0	2/3	660.9	140.4	1/3	0.0	0.0
Blocks Ranking Size	1/3	5.5	0.0	2/3	919.4	1112.6	1/3	0.0	0.0	1/3	672.1	748.2	1/3	0.0	0.0
Click Alarm Clock	3/3	115.2	0.0	3/3	844.9	1022.8	3/3	0.0	0.0	3/3	707.3	72.2	3/3	0.0	0.0
Click Bell	2/3	0.0	0.0	3/3	762.5	1032.9	3/3	0.0	0.0	3/3	651.1	55.0	3/3	0.0	0.0
Dump Bin Big Bin	2/3	0.0	0.0	3/3	440.9	291.3	3/3	0.0	0.0	3/3	400.6	0.0	2/3	0.0	0.0
Grab Roller	3/3	26.3	0.0	3/3	855.6	1199.8	3/3	0.0	0.0	3/3	579.8	0.0	3/3	0.0	0.0
Handover Block	1/3	0.0	0.0	3/3	769.9	1035.1	3/3	0.0	0.0	2/3	471.3	38.6	3/3	0.0	0.0
Handover Mic	3/3	0.0	0.0	3/3	488.0	281.3	3/3	0.0	0.0	3/3	653.9	16.4	2/3	0.0	0.0
Hanging Mug	2/3	0.0	0.0	1/3	467.0	0.0	1/3	0.0	0.0	0/3	426.9	388.3	0/3	0.0	0.0
Lift Pot	3/3	0.0	0.0	3/3	689.1	635.0	3/3	0.0	0.0	3/3	435.0	387.9	2/3	0.0	0.0
Move Can Pot	3/3	0.0	0.0	3/3	762.2	1025.1	2/3	0.0	0.0	2/3	667.6	633.2	2/3	0.0	0.0
Move Pillbottle Pad	3/3	0.0	0.0	3/3	876.7	1091.0	3/3	0.0	0.0	2/3	412.0	0.0	3/3	0.0	0.0
Move Playingcard Away	3/3	46.5	0.0	3/3	855.4	1087.1	2/3	0.0	0.0	3/3	705.2	633.7	3/3	0.0	0.0
Move Stapler Pad	3/3	0.0	0.0	0/3	–	–	1/3	0.0	0.0	1/3	420.8	0.0	0/3	0.0	0.0
Open Laptop	3/3	0.0	0.0	0/3	–	–	3/3	0.0	0.0	3/3	623.0	0.0	3/3	0.0	0.0
Open Microwave	1/3	0.0	0.0	1/3	959.0	1076.8	0/3	0.0	0.0	1/3	478.6	151.0	2/3	0.0	0.0
Pick Diverse Bottles	3/3	0.0	0.0	3/3	497.9	316.6	3/3	0.0	0.0	3/3	458.2	396.2	3/3	0.0	0.0
Pick Dual Bottles	3/3	0.0	0.0	3/3	694.9	690.6	3/3	0.0	0.0	3/3	444.3	388.7	3/3	0.0	0.0
Place A2B Left	3/3	10.9	0.0	3/3	828.5	998.0	2/3	0.0	0.0	1/3	663.1	631.5	3/3	0.0	0.0
Place A2B Right	2/3	0.0	0.0	2/3	750.5	962.2	2/3	0.0	0.0	3/3	657.4	0.0	3/3	0.0	0.0
Place Bread Basket	3/3	0.0	0.0	3/3	663.6	1007.6	3/3	0.0	0.0	3/3	404.2	0.0	3/3	0.0	0.0
Place Bread Skillet	2/3	0.0	0.0	2/3	420.0	0.0	1/3	0.0	0.0	3/3	669.9	0.0	2/3	0.0	0.0
Place Burger Fries	3/3	0.0	0.0	3/3	338.7	0.0	3/3	0.0	0.0	3/3	399.8	0.0	3/3	0.0	0.0
Place Can Basket	1/3	0.0	0.0	1/3	53.7	0.0	1/3	0.0	0.0	3/3	641.4	0.0	2/3	0.0	0.0
Place Cans Plasticbox	3/3	0.0	0.0	3/3	433.4	0.0	3/3	0.0	0.0	2/3	661.5	0.0	3/3	0.0	0.0
Place Container Plate	3/3	0.0	0.0	3/3	801.2	1000.6	3/3	0.0	0.0	3/3	400.3	0.0	3/3	0.0	0.0
Place Dual Shoes	2/3	0.0	0.0	3/3	329.2	0.0	3/3	0.0	0.0	3/3	646.1	0.0	2/3	0.0	0.0
Place Empty Cup	3/3	17.1	0.0	3/3	723.7	1007.0	3/3	0.0	0.0	3/3	669.1	0.0	3/3	0.0	0.0
Place Fan	3/3	0.0	0.0	3/3	308.8	0.0	2/3	0.0	0.0	3/3	405.2	0.0	2/3	0.0	0.0
Place Mouse Pad	0/3	–	–	0/3	–	–	2/3	0.0	0.0	2/3	411.3	0.0	2/3	0.0	0.0
Place Object Basket	1/3	0.0	0.0	2/3	238.5	0.0	1/3	0.0	0.0	3/3	665.2	633.4	3/3	0.0	0.0
Place Object Scale	2/3	0.0	0.0	0/3	–	–	3/3	0.0	0.0	2/3	662.9	610.2	3/3	0.0	0.0
Place Object Stand	3/3	0.0	0.0	2/3	879.2	1025.9	3/3	0.0	0.0	3/3	413.4	0.0	3/3	0.0	0.0
Place Phone Stand	3/3	0.0	0.0	3/3	748.9	970.8	3/3	0.0	0.0	3/3	609.1	0.0	3/3	0.0	0.0
Place Shoe	3/3	0.0	0.0	2/3	487.8	258.1	3/3	0.0	0.0	3/3	436.8	0.0	3/3	0.0	0.0

Table 11: Complete ablation task-level statistics (continued).

	FlowWAM						LingBot-VA								
	Pure overlap			Half launch			Pure overlap			Half launch			Pure + cal.		
Task	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50	SR	Mean	p50
Press Stapler	3/3	0.0	0.0	3/3	442.1	132.2	3/3	0.0	0.0	2/3	750.2	196.5	2/3	0.0	0.0
Put Bottles Dustbin	1/3	5.7	0.0	1/3	492.8	273.2	2/3	0.0	0.0	2/3	438.2	262.4	1/3	0.0	0.0
Put Object Cabinet	2/3	0.0	0.0	0/3	–	–	2/3	0.0	0.0	2/3	407.7	0.0	2/3	0.0	0.0
Rotate QR Code	3/3	0.0	0.0	3/3	912.8	1086.1	3/3	0.0	0.0	3/3	448.0	389.1	3/3	0.0	0.0
Scan Object	3/3	0.0	0.0	3/3	686.9	1015.6	2/3	0.0	0.0	2/3	405.1	0.0	3/3	0.0	0.0
Shake Bottle	3/3	19.3	0.0	3/3	683.8	875.3	3/3	0.0	0.0	3/3	450.3	17.4	3/3	0.0	0.0
Shake Bottle Horizontally	3/3	0.0	0.0	3/3	855.7	1044.7	3/3	0.0	0.0	3/3	455.1	25.4	3/3	0.0	0.0
Stack Blocks Three	2/3	29.2	0.0	3/3	871.9	1043.3	2/3	0.0	0.0	2/3	426.0	15.6	3/3	0.0	0.0
Stack Blocks Two	3/3	0.0	0.0	3/3	905.5	1046.9	3/3	0.0	0.0	3/3	640.0	0.0	3/3	0.0	0.0
Stack Bowls Three	1/3	5.9	0.0	1/3	817.9	1033.0	3/3	0.0	0.0	2/3	733.6	378.4	2/3	0.0	0.0
Stack Bowls Two	2/3	3.7	0.0	3/3	883.4	1028.7	3/3	0.0	0.0	3/3	670.8	118.5	3/3	0.0	0.0
Stamp Seal	2/3	83.6	0.0	3/3	766.6	1008.5	3/3	0.0	0.0	3/3	687.2	634.6	3/3	0.0	0.0
Turn Switch	2/3	5.8	0.0	1/3	762.4	835.5	3/3	0.0	0.0	2/3	438.8	0.0	3/3	0.0	0.0
All tasks	**121/150**	6.8	0.0	**116/150**	657.5	947.7	**123/150**	0.0	0.0	**125/150**	540.1	197.9	**125/150**	0.0	0.0