FAST THINKING, STEADY CONTROL: COHERENT ACTION-CHUNK TRANSITIONS FOR REAL-TIME VLA INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

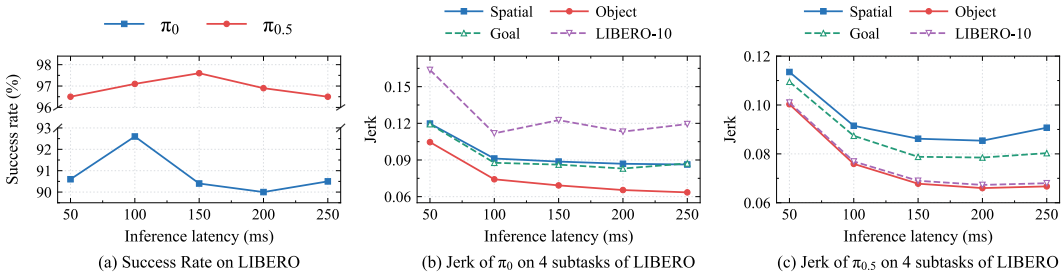
Vision-Language-Action (VLA) models have achieved impressive performance on diverse embodied tasks. To improve responsiveness to the environment, a growing body of work has focused on accelerating VLA inference. However, in real-world deployment, faster VLA inference does not necessarily translate into better control. Under continuous asynchronous inference, lower inference latency increases the frequency of chunk switching, which introduces inter-chunk discontinuities and latency-alignment errors, leading to discontinuous, jerky motion and even degraded task success. In this paper, we propose F2S-VLA, a training-free execution framework that translates fast asynchronous VLA inference into steady and coherent robot control. To mitigate inter-chunk discontinuity, we propose Noise-Decoupled Chunk Prediction, which generates two action chunks in a single forward pass with reference and newly sampled noise to separate observation-driven trajectory updates from noise-induced variations, and Anchor-Guided Trajectory Fusion, which uses the ongoing trajectory as an anchor to selectively fuse the new predictions, adaptively balancing responsiveness and continuity without task-specific thresholds. To eliminate latency-alignment errors, we introduce Latency-Aligned Action Resampling, which registers action chunks on a shared physical timeline and resamples them at the measured takeover time using the controller's native interpolation rule, improving temporal alignment granularity from tens of milliseconds to less than two milliseconds with negligible overhead. Experiments demonstrate that F2S-VLA outperforms RTC by 9.0% on RoboCasa365 benchmark and achieves a 93.9% average success rate across three long-horizon real-robot tasks. Our code is available on Anonymous Github.

1 INTRODUCTION

Vision-language-action (VLA) models have demonstrated strong performance across a wide range of embodied tasks (Black et al., 2024; Kim et al., 2024; Dai et al., 2025). By leveraging the broad semantic and visual knowledge acquired during vision-language pretraining and adapting it with robot demonstration data, VLAs can directly predict executable robot actions from visual observations and language instructions (Intelligence et al., 2025; Kim et al., 2025; Shi et al., 2026). However, effective deployment in the physical world requires low response latency, as robots must react promptly to changes in their environment (Yue et al., 2024; Song et al., 2025; Zheng et al., 2026b).

To achieve this, recent studies have explored both algorithmic and systems-level VLA inference optimization. Algorithmic approaches reduce computation through techniques such as quantization, pruning, and feature caching (Zhang et al., 2026; Zheng et al., 2026a; Xu et al., 2026c; Wang et al., 2025; Jiang et al., 2025; Li et al., 2025; Xu et al., 2026b). In parallel, optimized inference systems eliminate runtime overhead through computation-graph reuse and specialized GPU kernels (Su, 2026; Yang et al., 2026d), without modifying model architectures. These advances have brought VLA inference on an RTX 4090 into the tens-of-milliseconds regime. For example, Real-time VLA reports latencies of 32.9 ms, and 39.1 ms for full π_0 model with two, and three camera views, respectively (Ma et al., 2025). Meanwhile, asynchronous inference overlaps VLA prediction with the execution of the current action chunk, avoiding pauses between consecutive chunks while allowing the policy to continuously generate and update action trajectories (Zhao et al., 2025).

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.



(a) Success Rate on LIBERO (b) Jerk of π_0 on 4 subtasks of LIBERO (c) Jerk of $\pi_{0.5}$ on 4 subtasks of LIBERO

Figure 1: **(a)** Average success rates of π_0 and $\pi_{0.5}$ on LIBERO under different inference latencies. **(b–c)** Average trajectory jerk on LIBERO under different inference latencies, using π_0 and $\pi_{0.5}$. Jerk measures the temporal rate of acceleration change, with lower values indicating smoother motion.

Together, these optimizations substantially improve the responsiveness of VLAs during real-world deployment (Yang et al., 2026a; Niu et al., 2026).

However, in real-world deployment, faster inference does not necessarily translate into better control performance. As shown in Fig. 1 (a), task success rate (SR) can even decrease when inference latency becomes sufficiently low. Under continuous asynchronous inference, lower latency causes newly predicted action chunks to arrive more frequently and therefore increases the frequency of chunk switching (Zhao et al., 2025; Yang et al., 2026d). For example, at a 30 Hz action rate and 40 ms inference latency, each chunk executes only one or two actions before being replaced. As shown in Figure 1 (b)(c), this leads to discontinuous and jerky motion, where jerk measures the rate of acceleration change and reflects motion smoothness (Li et al., 2026; Lu et al., 2025). We find this degradation arises from two sources. First, temporal coherence is primarily learned within each action chunk, whereas consecutive chunks are generated by separate inference calls (Yuan et al., 2026; Nie et al., 2026). This results in inter-chunk discontinuities, which become more evident under fast inference even with continuity-enhancing methods such as RTC (Black et al., 2026). Second, inference latency generally does not correspond to an integer number of action steps. Consequently, conventional asynchronous execution, which compensates for inference latency by skipping a discrete number of actions in new chunk, introduces a temporal alignment error between the first retained action and its actual takeover time. This error occurs at chunk transitions and accumulates over time, exacerbating discontinuity, particularly in low-latency inference with frequent chunk updates.

To address these problems, we introduce F2S-VLA, a training-free execution framework that translates fast asynchronous VLA inference into steady and coherent robot control. To mitigate inter-chunk discontinuities without sacrificing responsiveness, we first propose **Noise-Decoupled Chunk Prediction**. At each planning step, F2S-VLA generates two action chunks under the latest observation: one reuses the noise associated with the previous prediction, while the other uses newly sampled noise. Comparing these predictions with the currently executed chunk operationally separates observation-driven trajectory updates from noise-induced trajectory variation. We further introduce **Anchor-Guided Trajectory Fusion**, which uses the ongoing trajectory as an anchor, blending it with the two new predictions to preserve observation-driven updates while attenuating noise-induced deviations that would disrupt continuity. Finally, to eliminate latency-alignment errors, we propose **Latency-Aligned Action Resampling**. Instead of approximating inference latency by directly skipping an integer number of actions (Shukor et al., 2025; Black et al., 2025), it first generate fine-grained trajectory by the low-level controller and resamples it at the takeover time, aligning the new chunk with the robot's actual execution timeline. Experiments demonstrate that F2S-VLA outperforms RTC by 9.0% on RoboCasa365 benchmark (Nasiriany et al., 2026) and achieves a 93.9% average success rate across three long-horizon real-robot tasks.

2 Background

Real-time VLA Inference. Recent work has substantially reduced VLA inference latency through systems and algorithmic optimizations (Yu et al., 2025; Guan et al., 2025; Yang et al., 2026c; Park et al., 2024). Real-Time VLA and FlashRT accelerate inference through CUDA graph reuse, optimized kernels, and low-precision computation (Ma et al., 2025; Su, 2026; Nguyen et al., 2026; Xu et al., 2026a). FASTER prioritizes near-term actions during flow sampling to reduce time to first action, while Realtime-VLA FLASH reduces full-policy evaluations through speculative inference

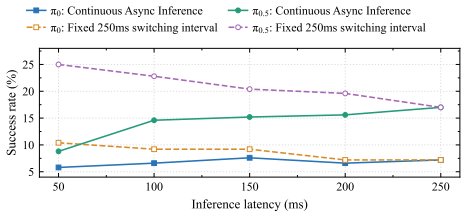


Figure 2: Success rate with continuous asynchronous inference and fixed-250ms chunk switching interval on RoboCasa365.

Table 1: Inference latency (ms) of different inference engines across 5 runs, evaluated on LIBERO with π_0.

Evaluation Times	1	2	3	4	5
Real-Time VLA	36.00	33.89	36.06	33.80	33.80
FlashRT	46.93	54.57	43.81	48.78	44.05
OpenPI JAX	67.85	78.91	66.23	69.08	78.85

(Lu et al., 2026; Niu et al., 2026). Jetson-PI combines inference-engine optimizations with adaptive scheduling for efficient onboard deployment (Yang et al., 2026d). Collectively, these advances enable action chunks to be generated at increasingly high frequencies. Our work studies the complementary problem: how to integrate rapidly arriving action chunks into ongoing execution without sacrificing cross-chunk coherence, thereby realizing full potential of VLA acceleration.

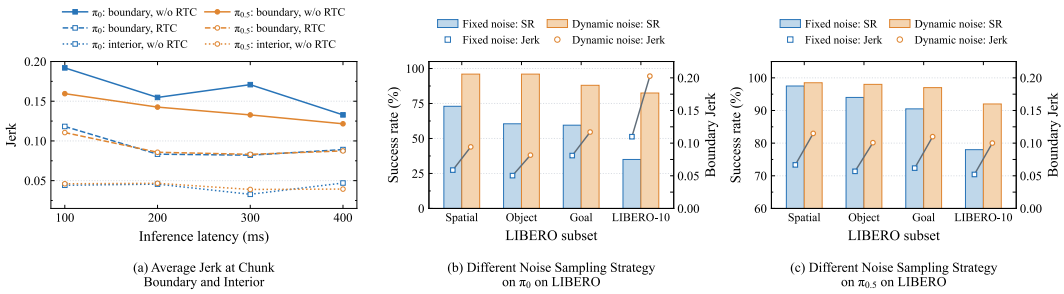
Asynchronous VLA Inference. Asynchronous VLA inference overlaps inference with action execution, but requires each newly arriving chunk to be smoothly integrated with the ongoing trajectory (Hirose et al., 2026; Wei et al., 2026). Action-prefix-conditioned methods, such as RTC, enforce inter-chunk continuity via previously committed actions (Black et al., 2026; 2025); however, low inference latency yields only a short committed prefix, providing insufficient constraints on the remaining actions. Future-state-aware methods instead predict the next chunk from the anticipated handoff state of the ongoing trajectory (Tang et al., 2025; Jiang et al., 2026), but successive chunks are still generated independently, leaving stochastic inconsistency at the chunk boundary unaddressed. In contrast, F2S-VLA directly decouples observation-driven updates from noise-induced variations and suppresses those that harm inter-chunk continuity via selective trajectory fusion, enabling coherent chunk transitions under real-time inference.

Adaptive Chunk Execution. Another line of work controls how many actions are executed from each chunk or when a new chunk takes over (Jing et al., 2025; Zhao et al., 2026). SGAC compares queued and newly predicted actions to decide whether to retain or replace the current chunk (So et al., 2026). Adaptive Action Chunking (AAC) uses action uncertainty to dynamically determine the execution horizon (Liang et al., 2026), while A^3 uses multi-sample trajectory consensus and self-speculative verification to select the reliable action prefix (Chen et al., 2026). However, they primarily formulate adaptive execution as a discrete gating problem—whether to adopt a new chunk or how many actions to commit—forcing a hard choice between the old and new chunks and leaving the boundary transition uncoordinated. In contrast, F2S-VLA fuses the trajectories to preserve necessary updates while attenuating incompatible stochastic deviations.

3 Why Lower Latency Does Not Guarantee Better Control

Is Faster VLA Inference Genuinely Beneficial? We first examine whether faster VLA inference genuinely improves robot performance when its effect is decoupled from chunk-switching frequency. We fix each chunk's execution duration at $250\,\mathrm{ms}$ (chunk-switching frequency fixed at $4\,\mathrm{Hz}$) and, for inference latency $L < 250\,\mathrm{ms}$, insert a scheduling delay of $(250 - L)\,\mathrm{ms}$ to maintain this switching interval. As shown in Figure 2, lower latency consistently improves performance under a fixed switching frequency. However, conventional asynchronous execution couples fast inference with frequent chunk transitions, limiting this benefit by increased discontinuity. We identify two primary sources of this degradation: inter-chunk discontinuity and latency-alignment error.

Inter-chunk discontinuity. As shown in Figure 3 (a), even with RTC applied, discontinuity across chunk boundaries remains substantially larger than that within individual chunks. This is because actions within a chunk are predicted jointly, allowing their temporal coherence to be optimized during training, whereas consecutive chunks are produced by largely independent inference calls (Ma et al., 2025; Yuan et al., 2026). We find that such discontinuity has two main sources. First, environmental changes may require genuine policy updates, making deviation from the previous trajectory necessary for responsive control. Second, independently sampled action noise can induce spurious "mode hopping" even when no meaningful behavioral change is required (Chi et al., 2025; Liu et al.,



(a) Average Jerk at Chunk Boundary and Interior

(b) Different Noise Sampling Strategy on π_0 on LIBERO

(c) Different Noise Sampling Strategy on $\pi_{0.5}$ on LIBERO

Figure 3: (a) Jerk at chunk boundary and interior of π_0 and $\pi_{0.5}$ under different inference latencies. (b,c) Success rate and boundary jerk under fixed-noise and dynamic-noise sampling on LIBERO.

2026). As shown in Figure 3 (b)(c), reusing the same noise across all VLA predictions substantially reduces boundary jerk. However, completely eliminating noise-induced variation can suppress action diversity and reduce task success (Zhan et al., 2026; Liu et al., 2025). Therefore, an effective method should only attenuate stochastic variations inconsistent with environmental changes, thereby balancing inter-chunk continuity and action diversity.

Latency-alignment error. In conventional asynchronous execution, the first $d = \lceil (\tau - t_0)/\Delta_a \rfloor$ actions of a new chunk are discarded to compensate for robot motion during inference, where τ is new chunk takeover time , t_0 is the inference start time and Δ_a is the action interval. Since actions are defined only at discrete steps, it introduces a residual latency-alignment error, $\phi = (\tau - t_0) - d\Delta_a$, between the first retained action and the actual takeover time. This error occurs at each chunk transition and accumulates over time (Yuan et al., 2026; Xiao et al., 2020). Changing the action execution frequency cannot completely eliminate it because inference latency fluctuates across calls because of varying device operating conditions, as shown in Table 1. Moreover, deviating from the training-time action rate may introduce potential train-deployment mismatch (Sun et al., 2026; Arachchige et al., 2025). These observations motivate a latency-aware method that aligns each new chunk with its measured takeover time while preserving the policy's native action frequency.

Overview. Motivated by these observations, we propose F2S-VLA, a training-free framework that converts fast asynchronous VLA inference into coherent robot control. As illustrated in Figure 4, Noise-Decoupled Chunk Prediction generates two candidate chunks from the latest observation: $\mathbf{R}$ reuses the reference noise from the previous prediction, whereas $\mathbf{N}$ uses newly sampled noise, enabling observation-driven updates to be distinguished from noise-induced variations. Latency-Aligned Action Resampling then resamples $\mathbf{R}$ and $\mathbf{N}$ at the measured takeover time to align them with the current execution phase of the ongoing chunk $\mathbf{P}$. Finally, Anchor-Guided Trajectory Fusion uses $\mathbf{P}$ as an anchor to fuse the aligned predictions, preserving necessary updates while attenuating noise-induced discontinuities. Together, these components improve temporal alignment and inter-chunk continuity without retraining the VLA or changing its native action frequency.

4 Method

4.1 Noise-Decoupled Chunk Prediction

As discussed in Section 3, trajectory changes can arise from either environmental changes or noise sampling. We therefore seek to separate these two effects before trajectory fusion.

As illustrated in Figure 4 (c), at each inference step, the action expert generates two trajectories in a single forward pass under the same visual-language context: $\mathbf{R}$ reuses the reference noise from the previous prediction, whereas $\mathbf{N}$ uses newly sampled noise. A branch-isolated attention mask allows both branches to attend to the shared context while preventing cross-branch interaction. The new prediction can then be decomposed as

$$\mathbf{N} = \mathbf{P} + \underbrace{(\mathbf{R} - \mathbf{P})}_{\mathbf{u}} + \underbrace{(\mathbf{N} - \mathbf{R})}_{\mathbf{v}}. \tag{1}$$

Here, trajectory subtraction is performed action-wise at corresponding time, producing a sequence of action differences over the shared horizon. Accordingly, $\mathbf{u}$ captures the action change induced by the latest observation, whereas $\mathbf{v}$ captures the action variation introduced by noise sampling.

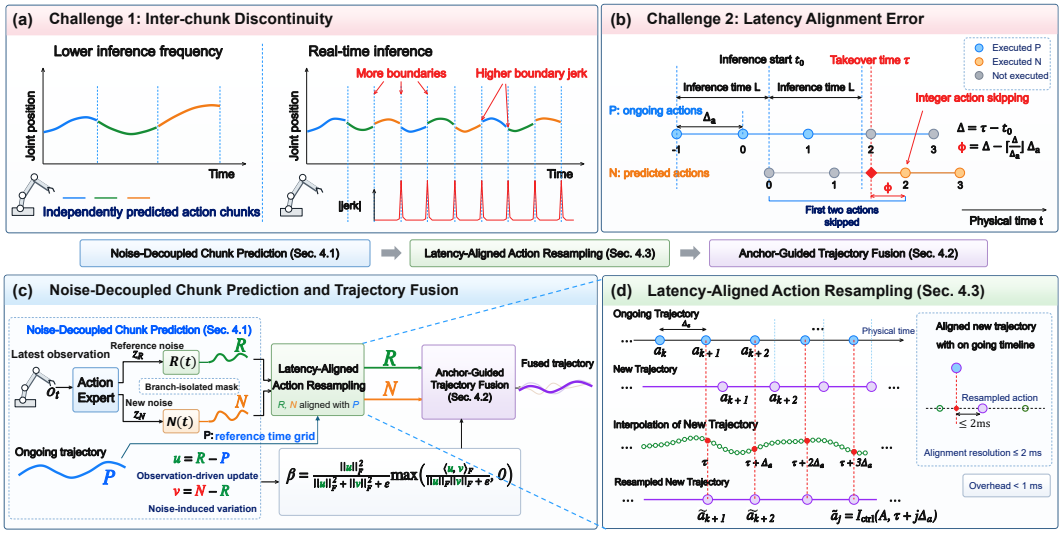


Figure 4: Overview of F2S-VLA.

Table 2: Average inference latency (ms) of π_0 under varying action token numbers.

	2 input images					3 input images				
# Action Tokens	20	40	60	80	100	20	40	60	80	100
Real-Time VLA	33.18	33.29	33.98	34.29	35.25	40.49	40.63	41.46	42.60	42.89
FlashRT	43.13	43.58	44.01	45.14	45.48	57.93	58.03	59.02	60.36	60.67
OpenPI JAX	64.62	66.79	67.39	67.40	68.19	75.18	75.99	76.79	77.37	78.00

We quantify the relative contribution of the observation-driven update using

$$\kappa = \frac{\|\mathbf{u}\|_F^2}{\|\mathbf{u}\|_F^2 + \|\mathbf{v}\|_F^2 + \epsilon}. \tag{2}$$

A large κ indicates the deviation from $\mathbf{P}$ is primarily supported by the latest observation; a small κ indicates the deviation is dominated by stochastic noise rather than meaningful behavioral changes.

Latency analysis. Introducing an additional noise branch doubles the number of action tokens processed by the action expert and thus increases its arithmetic operations. However, it introduces negligible end-to-end latency. We explain this behavior using the roofline model (Williams et al., 2009; Guo et al., 2024). The dominant computation in VLA inference is matrix multiplication. For a matrix multiplication between an activation of shape $M \times K$ and a weight matrix of shape $K \times N$, its execution time can be approximated as

$$T_{\text{GEMM}} \approx \max\left(\frac{2MKN}{C}, \frac{2(MK + KN + MN)}{B}\right), \tag{3}$$

where C and B denote the compute throughput and memory bandwidth, respectively. For VLA models, the action expert operates on short action sequences: even with two noise branches, M typically remains below 100, for which the bandwidth term in the roofline model typically dominates. Since $K, N \gg M$, transferring the weight matrix dominates the bandwidth cost, yielding $T_{\text{GEMM}} \approx 2KN/B$, which depends only weakly on M. The experiments in Table 2 further demonstrate this point. The additional branch therefore reuses the loaded weights and utilizes underused compute capacity, only increasing FLOPs without proportionally increasing inference latency.

4.2 Anchor-Guided Trajectory Fusion

From discrete switching to trajectory fusion. Based on the separation of observation-driven updates and noise-induced variations provided by Noise-Decoupled Chunk Prediction, a straightforward strategy for chunk transition is to switch between the previous trajectory $\mathbf{P}$ and the new trajec-

tory $\mathbf{N}$ according to whether the update score κ exceeds a threshold λ:

$$\mathbf{A}^{\text{switch}} = \begin{cases} \mathbf{N}, & \kappa \geq \lambda, \\ \mathbf{P}, & \kappa < \lambda. \end{cases} \tag{4}$$

However, this discrete strategy has two limitations. First, environmental changes and noise-induced variations may occur simultaneously, in which case neither retaining $\mathbf{P}$ nor fully adopting $\mathbf{N}$ is optimal. Second, the appropriate threshold can vary across tasks and environments, limiting generalization. We therefore propose a threshold-free trajectory fusion method that responds to environmental changes while keeping trajectory continuity.

Specifically, we consider both the relative magnitude and directional agreement of the observation-driven update $\mathbf{u}$ and the noise-induced variation $\mathbf{v}$. We compute their directional agreement as

$$\rho = \frac{\langle \mathbf{u}, \mathbf{v} \rangle_F}{\|\mathbf{u}\|_F \|\mathbf{v}\|_F + \epsilon}, \tag{5}$$

and define the fusion weight as

$$\beta = \kappa[\rho]_+, \qquad [\rho]_+ = \max(\rho, 0). \tag{6}$$

The final action trajectory is then

$$\mathbf{A}^{\text{fuse}} = \mathbf{P} + (1-\beta)^2 (\mathbf{R} - \mathbf{P}) + \beta^2 (\mathbf{N} - \mathbf{P}). \tag{7}$$

This formulation naturally handles three cases. **1)** When the environment remains largely unchanged, we have $\mathbf{u} \approx \mathbf{0}$ and hence $\kappa \approx \beta \approx 0$, yielding $\mathbf{A}^{\text{fuse}} \approx \mathbf{R} \approx \mathbf{P}$, which means the method preserves the ongoing trajectory rather than responding to sampling-induced deviations. **2)** When a clear environmental change produces consistent predictions from both noise branches, indicating that the newly sampled noise introduces negligible trajectory variation, we have $\beta \approx 1$ and $\mathbf{A}^{\text{fuse}} \approx \mathbf{N}$, so the fused trajectory follows their common update. **3)** When environmental changes and noise-induced variations coexist, β continuously interpolates among the anchor trajectory $\mathbf{P}$ and the predictions $\mathbf{R}$ and $\mathbf{N}$, according to their relative magnitude and directional agreement. In particular, a conflicting noise variation with $\rho \leq 0$ is rejected, whereas a noise-induced variation with $\rho > 0$ is incorporated in proportion to its agreement with the observation-driven update. The method therefore adapts the trajectory without requiring a task-specific switching threshold. Note that Eq. 7 is only for continuous action dimensions. For discrete gripper commands, we perform weighted categorical voting using the same fusion coefficients, thereby preserving valid discrete outputs.

Slow-timescale anchor refresh. At low inference latency, naively refreshing the reference after every inference call creates a moving-reference problem. Specifically, after each inference call, $\mathbf{P}$ and the reference noise will be replaced by the latest prediction $\mathbf{N}$ and its newly sampled noise, respectively. Consequently, sampling-induced offsets are repeatedly incorporated into the reference. Even if $\mathbf{N}$ receives only a small weight in each fusion step, the rapid reference changes can still prevent the method from maintaining inter-chunk continuity. To avoid this effect, we maintain the reference trajectory $\mathbf{P}$ and reference noise on a slower timescale ($500\,\text{ms}$ in our implementation). All predictions within the same interval are therefore evaluated against a stable reference. This does not collapse action diversity because: **1)** the anchor interval is much shorter than the duration of a task, allowing the policy to explore multiple action modes over an episode (Nasiriany et al., 2026; Liu et al., 2026). **2)** Each fusion step still evaluates the trajectory $\mathbf{N}$ generated from newly sampled noise and incorporates it whenever it is supported by observation-driven update.

4.3 Latency-Aligned Action Resampling

As discussed in Section 3, an action chunk is conventionally represented as a discrete sequence of actions rather than a time-continuous trajectory. Consequently, the continuous inference latency cannot generally be mapped to an exact action index. Skipping an integer number of actions discards the fractional part of the latency and introduces temporal misalignment between the predicted chunk and the robot's actual takeover time. However, in practical robot control systems, after VLA prediction, the robot controller usually interpolates between each two consecutive action waypoints, for example using linear or spline interpolation, to construct a denser trajectory for low-level execution

(Luo et al., 2024; Chi et al., 2025; Yang et al., 2026b). Based on this, we propose Latency-Aligned Action Resampling to correct the temporal misalignment.

To address this issue, we associate every predicted action with its intended physical execution time:

$$\mathcal{A} = \{(t_i, \mathbf{a}_i)\}_{i=0}^{H-1}, \qquad t_i = t_0 + i\Delta_{\mathrm{a}}. \tag{8}$$

This representation registers trajectories generated by different inference calls on a shared physical timeline rather than treating their action indices as directly interchangeable.

To correct the temporal misalignment between the predicted chunk and the robot's actual takeover time, as shown in Figure 4, we expand the predicted discrete waypoints $\mathcal{A}$ into a fine-grained trajectory using the controller's native interpolation rule (e.g., linear, cubic, or quintic-Hermite interpolation), and then sample the action at the target takeover time τ from this trajectory:

$$\widetilde{\mathbf{a}}(\tau) \;=\; \mathcal{I}_{\mathrm{ctrl}}(\mathcal{A})(\tau), \tag{9}$$

where $\mathcal{I}_{\mathrm{ctrl}}(\mathcal{A})$ denotes the fine-grained trajectory generated from the controller. Applying this operation at future control timestamps $\tau_j = \tau + j\Delta_{\mathrm{a}}$ produces a latency-aligned action chunk:

$$\widetilde{\mathbf{A}} = [\widetilde{\mathbf{a}}(\tau_0), \ldots, \widetilde{\mathbf{a}}(\tau_{H'-1})] = [\mathcal{I}_{\mathrm{ctrl}}(\mathcal{A})(\tau_0), \ldots, \mathcal{I}_{\mathrm{ctrl}}(\mathcal{A})(\tau_{H'-1})], \tag{10}$$

This operation reduces the granularity of temporal alignment from the original sparse action interval Δ_{a} to the finer controller interpolation interval Δ_{ctrl}, e.g., from tens of milliseconds to less than 2 ms. Since trajectory interpolation is already highly optimized in modern low-level controllers, the additional resampling introduces negligible overhead, taking less than $1\,\mathrm{ms}$ in our implementation Zhao et al. (2025); Lind (2020).

Latency-Aligned Action Resampling is performed before trajectory comparison and fusion. The reference trajectory $\mathbf{P}$ and newly predicted trajectories $\mathbf{R}$ and $\mathbf{N}$ may have different temporal origins because they are generated by different inference calls. Comparing them directly by action index would therefore introduce temporal offsets, corrupting the estimates of $\mathbf{u}$ and $\mathbf{v}$. So first align the newly predicted trajectories $\mathbf{R}$ and $\mathbf{N}$ with the ongoing trajectory $\mathbf{P}$ on the execution interval following the takeover time, and then perform Noise-Decoupled Chunk Prediction and Anchor-Guided Trajectory Fusion on the aligned trajectories. This enables the framework to compensate for the measured latency of each inference call without changing the policy's native action frequency.

5 Experiments

We conduct comprehensive experiments in both sim and real to validate the following questions: **(1)** How does the performance of F2S-VLA compare to other asynchronous inference methods? **(2)** How much does each component—Noise-Decoupled Chunk Prediction, Anchor-Guided Trajectory Fusion, and Latency-Aligned Action Resampling—contribute to control performance? **(3)** Can F2S-VLA adapt its fusion behavior to different levels of environmental change while maintaining stable and responsive control in long-term robot tasks?

5.1 Simulation Experiments

Experiment Setup. We evaluate F2S-VLA on RoboCasa365 (Nasiriany et al., 2026) benchmark based on $\pi_{0.5}$ model. Compared with LIBERO (Liu et al., 2023), RoboCasa365 features substantially longer-term, household tasks whose sustained execution demands coherent control across many chunk transitions. It also provides far greater scene and object diversity. We compare F2S-VLA with three baselines: (1) **Continuous RTC** immediately launches the next inference call once the previous call finishes, and smooth chunk transitions with RTC, which freezes the actions committed to execute and inpaints the rest (Black et al., 2026). (2) **Fixed-horizon RTC** uses the same RTC mechanism but executes each chunk for a fixed $250\,\mathrm{ms}$. When inference takes less than $250\,\mathrm{ms}$, an additional scheduling delay is introduced so that chunk switches at a fixed frequency of $4\,\mathrm{Hz}$. (3) **VLASH** estimates the robot state at takeover time to guide action prediction (Tang et al., 2025).

Main Results. The main results are presented in Table 3. On average, our method achieves a success rate of 31.3%, outperforming Continuous RTC, Fixed-horizon RTC, and VLASH by 19.0%, 9.0%, and 8.6%, respectively. Continuous RTC degrades substantially at low latency, achieving only 8.8% at $100\,\mathrm{ms}$, as more frequent chunk arrivals lead to frequent chunk transitions and high trajectory

Table 3: Task success rates under different inference latencies evaluated on RoboCasa365. The first four rows use fixed inference latencies, while the RTX 4090 + FlashRT and RTX 4090 + Real-Time VLA settings use real dynamically measured latency of advanced VLA inference engines.

Inference Setting	Continuous RTC	Fixed-horizon RTC	VLASH	F2S-VLA
100 ms	8.8%	25.0%	20.4%	**31.6%**
200 ms	14.6%	22.8%	25.2%	**31.2%**
300 ms	15.2%	20.4%	25.2%	**30.6%**
400 ms	15.6%	19.6%	20.4%	**30.8%**
RTX 4090 + FlashRT	8.4%	22.6%	22.0%	**32.0%**
RTX 4090 + Real-Time VLA	11.2%	23.2%	23.2%	**31.6%**
Average	12.3%	22.3%	22.7%	**31.3%**

Table 4: Success rates (%) of different asynchronous execution methods on real-robot tasks.

Task	Continuous RTC	Fixed-Horizon RTC	F2S-VLA
Block stacking	82.5%	87.5%	**95.0%**
Placing pens in the drawer	77.5%	95.0%	**100.0%**
Placing fruits in the basket	48.9%	66.6%	**86.6%**
Average	69.6%	83.0%	**93.9%**

discontinuity. Fixed-horizon RTC mitigates this issue by limiting the switching frequency, but delays the incorporation of newly predicted actions and therefore sacrifices responsiveness. In contrast, our method continuously incorporates newly arriving predictions while attenuating noise-induced discontinuities and correcting latency- alignment errors, thereby maintaining stable performance across different inference latencies

5.2 Real-world Experiments

In real-world experiments, we evaluate F2S-VLA on three manipulation tasks. In *Block Stacking*, the robot must stack four blocks consecutively without toppling them, requiring continuous orientation adjustment and stable control throughout the sequence. In *Placing Pens in a Drawer*, the robot must place two pens into a drawer, while continuously adjusting the pen's pose to align it with the narrow opening and maintaining stable control during both drawer opening and pen placement. In *Placing Fruits in a Basket*, the robot must transfer all fruits on the table into a basket in the presence of distractor objects, requiring stable control when placing the fruit. As shown in Table 4, F2S-VLA outperforms both Continuous RTC and Fixed-Horizon RTC on all three tasks, achieving an average success rates of 93.9%. These results demonstrate that F2S-VLA can maintain both real-time inference responsiveness and stable control during long-horizon real-world manipulation.

5.3 Ablation study and Analysis

Impact of proposed methods. Table 5 reports the impact of each proposed method on RoboCasa365 tasks. Removing the **slow-timescale anchor** causes the largest degradation. Because the reference trajectory are then refreshed at every inference call, allowing noise-induced deviations to accumulate into reference trajectory under rapid asynchronous updates. Without **trajectory fusion**, the controller must rely on less selective chunk updates, making it harder to adaptively preserve continuity while responding to new observations. Performance drops notably on PickPlaceDrawerToCounter (80% $\rightarrow$ 40%), StackBowlsCabinet (80% $\rightarrow$ 60%), and StoreLeftoversInBowl (60% $\rightarrow$ 10%). Removing **resampling** leaves the latency-alignment error uncorrected. This particularly affects tasks requiring precise timing and transitions, such as StackBowlsCabinet (80% $\rightarrow$ 30%), and StoreLeftoversInBowl (60% $\rightarrow$ 20%). These results confirm that all three components contribute to robust real-time control. For tasks such as CoffeeSetupMug and GarnishPancake, all variants achieve relatively low success rates, suggesting that the performance bottleneck is primarily the underlying VLA's task-solving capability rather than the proposed control mechanism.

Trajectory fusion analysis. Figure 5 shows an trajectory fusion analysis. When the environment changes substantially, e.g., the robot is grasping things, as shown in Figure 5 (a,c), the method increases β to assign more weight to N in trajectory fusion, enabling a faster response to the envi-

Table 5: Impact of our proposed methods on different RoboCasa365 tasks.

Task	Full method	w/o Slow-timescale Anchor	w/o Trajectory Fusion	w/o Resampling
ArrangeBreadBasket	40%	0%	30%	30%
CloseFridge	90%	60%	70%	80%
CoffeeSetupMug	20%	20%	10%	10%
GarnishPancake	20%	0%	0%	10%
KettleBoiling	40%	0%	20%	30%
NavigateKitchen	30%	0%	40%	10%
PickPlaceDrawerToCounter	80%	20%	40%	70%
StackBowlsCabinet	80%	30%	60%	30%
StoreLeftoversInBowl	60%	0%	10%	20%
CloseToasterOvenDoor	30%	30%	10%	40%
MakeIceLemonade	20%	0%	0%	10%
PackIdenticalLunches	10%	0%	0%	0%

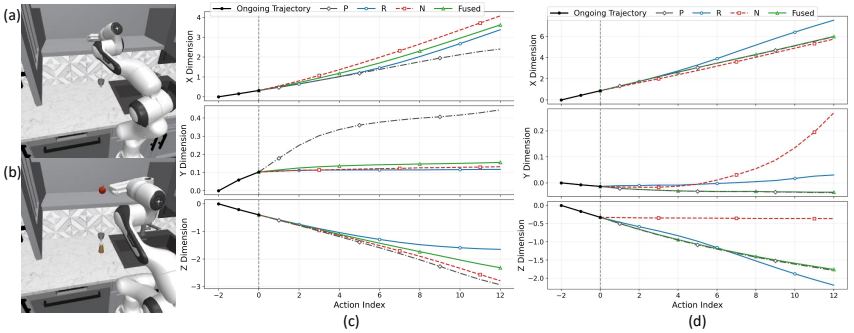


Figure 5: Trajectory fusion under different environmental changes. (a,c) show the observation and XYZ trajectories for a large β, while (b,d) show those for a small β.

ronment. When the environment remains relatively stable, as shown in Figure 5 (b,d), β is reduced so that the fused trajectory stays close to the ongoing trajectory and maintains temporal continuity. It shows F2S-VLA adaptively changes β to balance responsiveness and continuity.

Anchor Refresh Interval Ablation. Figure 6 evaluates the effect of the anchor refresh interval under 50 ms and 100 ms inference latency. Removing the anchor causes a substantial performance drop, as the reference trajectory and reference noise are then refreshed at every inference call, allowing noise-induced deviations to accumulate into the reference trajectory under rapid asynchronous updates. The strong performance across different anchor intervals demonstrates the robustness of our method. We also observe that the average fusion weight β increases with both inference latency and anchor refresh interval. As either increases, the anchor trajectory becomes less responsive to the current execution state on average over the entire task. F2S-VLA therefore assigns a larger weight to the freshly predicted trajectory $\mathbf{N}$, allowing the controller to recover responsiveness while retaining the continuity provided by the anchor. This adaptive change in β explains how the method balances temporal stability and real-time reactivity.

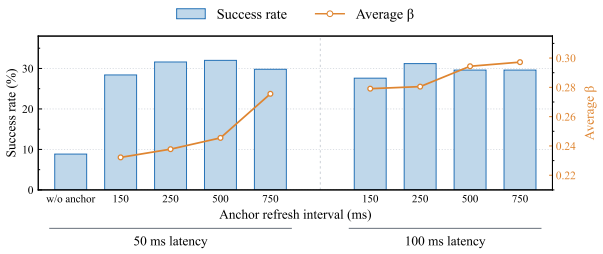


Figure 6: Ablation on the anchor refresh interval under different inference latencies. The "w/o anchor" setting removes the slow-timescale reference update.

6 Conclusion

We presented F2S-VLA, a training-free framework that turns fast asynchronous VLA inference into steady, coherent robot control. It addresses inter-chunk discontinuity and latency-alignment error by decoupling observation-driven updates from noise-induced variations, selectively fusing predictions into the ongoing trajectory, and resampling chunks on the execution timeline. F2S-VLA outperforms RTC by 9.0% on RoboCasa365 and achieves a 93.9% average success rate on real-robot tasks.

AI USE STATEMENT

We used generative AI tools solely to assist with English translation and grammar correction.

ETHICS STATEMENT

This work involves no human subjects or sensitive personal data and complies with the ICLR Code of Ethics. Real-robot experiments were conducted in a controlled laboratory environment with standard safety safeguards.

REPRODUCIBILITY STATEMENT

The method, experimental settings, evaluation metrics, and implementation details are described in the main paper. Code is provided on the Anonymous Github.

REFERENCES

Nadun Ranawaka Arachchige, Zhenyang Chen, Wonsuhk Jung, Woo Chul Shin, Rohan Bansal, Pierre Barroso, Yu Hang He, Yingyang Celine Lin, Benjamin Joffe, Shreyas Kousik, et al. Sail: Faster-than-demonstration execution of imitation learning policies. *arXiv preprint arXiv:2506.11948*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Allen Z Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking. *arXiv preprint arXiv:2512.05964*, 2025.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems*, 38:33383–33407, 2026.

Feng Chen, Xianghui Wang, Yuxuan Chen, Boying Li, Yefei He, Zeyu Zhang, and Yicheng Wu. Dynamic execution commitment of vision-language-action models. *arXiv preprint arXiv:2605.11567*, 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Yuntao Dai, Hang Gu, Teng Wang, Qianyu Cheng, Yifei Zheng, Zhiyong Qiu, Lei Gong, Wenqi Lou, and Xuehai Zhou. Actionflow: A pipelined action acceleration for vision language models on edge. *arXiv preprint arXiv:2512.20276*, 2025.

Weifan Guan, Qinghao Hu, Aosheng Li, and Jian Cheng. Efficient vision-language-action models for embodied manipulation: A systematic survey. *arXiv preprint arXiv:2510.17111*, 2025.

Qingyu Guo, Jiayong Wan, Songqiang Xu, Meng Li, and Yuan Wang. Hg-pipe: Vision transformer acceleration with hybrid-grained pipeline. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2024.

Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. Asyncvla: An asynchronous vla for fast and robust navigation on the edge. *arXiv preprint arXiv:2602.13476*, 2026.

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

Hai Jiang, Yixian Zou, Binbin Liang, Boqian Liu, Fanman Meng, and Shuaicheng Liu. Futurertc: Real-time robot execution with anticipatory-conditioned action chunking. *arXiv preprint arXiv:2607.24008*, 2026.

Titong Jiang, Xuefeng Jiang, Yuan Ma, Xin Wen, Bailin Li, Kun Zhan, Peng Jia, Yahui Liu, Sheng Sun, and Xianpeng Lang. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning. *arXiv preprint arXiv:2509.12594*, 2025.

Dong Jing, Gang Wang, Jiaqi Liu, Weiliang Tang, Zelong Sun, Yunchao Yao, Zhenyu Wei, Yunhui Liu, Zhiwu Lu, and Mingyu Ding. Mixture of horizons in action chunking. *arXiv preprint arXiv:2511.19433*, 2025.

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

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

Jiashun Li, Xiaoyu Shi, Hong Xie, Mingsheng Shang, and Yun Lu. Smoothvla: Aligning vision-language-action models with physical constraints via intrinsic smoothness optimization. *arXiv preprint arXiv:2603.13925*, 2026.

Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shutao Xia, Zhi Wang, and Wenwu Zhu. Sp-vla: A joint model scheduling and token pruning approach for vla model acceleration. *arXiv preprint arXiv:2506.12723*, 2025.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models. *arXiv preprint arXiv:2604.04161*, 2026.

Morten Lind. Real-time quintic hermite interpolation for robot trajectory execution. *PeerJ Computer Science*, 6:e304, 2020.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Yuejiang Liu, Jubayer Hamid, Annie Xie, Yoonho Lee, Max Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. In *International Conference on Learning Representations*, volume 2025, pp. 4594–4627, 2025.

Yufeng Liu, Hang Yu, Juntu Zhao, Bocheng Li, Di Zhang, Mingzhu Li, Wenxuan Wu, Yingdong Hu, Junyuan Xie, Junliang Guo, et al. Learning native continuation for action chunking flow policies. *arXiv preprint arXiv:2602.12978*, 2026.

Jiaxin Lu, Chun-Hao Paul Huang, Uttaran Bhattacharya, Qixing Huang, and Yi Zhou. Humoto: A 4d dataset of mocap human object interactions. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10886–10897. IEEE, 2025.

Yuxiang Lu, Zhe Liu, Xianzhe Fan, Zhenya Yang, Jinghua Hou, Junyi Li, Kaixin Ding, and Hengshuang Zhao. Faster: Rethinking real-time flow vlas. *arXiv preprint arXiv:2603.19199*, 2026.

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16961–16969. IEEE, 2024.

Yunchao Ma, Yizhuang Zhou, Yunhuan Yang, Tiancai Wang, and Haoqiang Fan. Running vlas at real-time speed. *arXiv preprint arXiv:2510.26742*, 2025.

Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 2026.

Khanh D Nguyen, Hung T Ho, Chinh T Nguyen, Thanh Q Duong, Linh D Le, Duy MH Nguyen, Vien A Ngo, and An T Le. vla. cpp: A unified inference runtime for vision-language-action models. *arXiv preprint arXiv:2606.08094*, 2026.

Junnan Nie, Jiayi Li, Jiachen Zhang, Junyi Lao, Chenghao Liu, Tianle Zhang, Liang Lin, and Songfang Huang. Pace: Phase-aware chunk execution for robot policies with action chunking. *arXiv preprint arXiv:2606.00537*, 2026.

Jiahui Niu, Kefan Gu, Yucheng Zhao, Shengwen Liang, Tiancai Wang, Xing Hu, Ying Wang, and Huawei Li. Realtime-vla flash: Speculative inference framework for diffusion-based vlas. *arXiv preprint arXiv:2605.13778*, 2026.

Seongmin Park, Hyungmin Kim, Wonseok Jeon, Juyoung Yang, Byeongwook Jeon, Yoonseon Oh, and Jungwook Choi. Quantization-aware imitation-learning for resource-efficient robotic control. *arXiv preprint arXiv:2412.01034*, 2024.

Yiran Shi, Dongqi Guo, Tianchen Zhao, Feng Gao, Liangzhi Shi, Chao Yu, ZhiJian Mo, Qihua Xiao, XiaoShuai Peng, Qingmin Liao, et al. Streamingvla: Streaming vision-language-action model with action flow matching and adaptive early observation. *arXiv preprint arXiv:2603.28565*, 2026.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Junhyuk So, Chiwoong Lee, Shinyoung Lee, Jungseul Ok, and Eunhyeok Park. Improving generative behavior cloning via self-guidance and adaptive chunking. *Advances in Neural Information Processing Systems*, 38:84432–84456, 2026.

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

Liang Su. Execution-state capsules: Graph-bound execution-state checkpoint and restore for low-latency, small-batch, on-device physical-ai serving. *arXiv preprint arXiv:2606.20537*, 2026.

Yuteng Sun, Haoran Wang, Ruofei Bai, Zhengguo Li, Jun Li, Meng Yee Michael Chuah, and Wei Yun Yau. Tidal: Temporally interleaved diffusion and action loop for high-frequency vla control. *arXiv preprint arXiv:2601.14945*, 2026.

Jiaming Tang, Yufei Sun, Yilong Zhao, Shang Yang, Yujun Lin, Zhuoyang Zhang, James Hou, Yao Lu, Zhijian Liu, and Song Han. Vlash: Real-time vlas via future-state-aware asynchronous inference. *arXiv preprint arXiv:2512.01031*, 2025.

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

Haoyu Wei, Xiuwei Xu, Ziyang Cheng, Hang Yin, Angyuan Ma, Bingyao Yu, Jie Zhou, and Jiwen Lu. F2f-ap: Flow-to-future asynchronous policy for real-time dynamic manipulation. *arXiv preprint arXiv:2604.02408*, 2026.

Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

Ted Xiao, Eric Jang, Dmitry Kalashnikov, Sergey Levine, Julian Ibarz, Karol Hausman, and Alexander Herzog. Thinking while moving: Deep reinforcement learning with concurrent control. *arXiv preprint arXiv:2004.06089*, 2020.

Ling Xu, Borui Li, Hao Wu, Chuyu Han, Xiangyu Li, Mohan Hua, Shiqi Jiang, Ting Cao, Chuanyou Li, Sheng Zhong, et al. Embodied. cpp: A portable inference runtime of embodied ai models on heterogeneous robots. *arXiv preprint arXiv:2607.02501*, 2026a.

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

Yuhao Xu, Yantai Yang, Zhenyang Fan, Yufan Liu, Yuming Li, Bing Li, and Zhipeng Zhang. Qvla: Not all channels are equal in vision-language-action model's quantization. *arXiv preprint arXiv:2602.03782*, 2026c.

Chen Yang, Yucheng Hu, Yunchao Ma, Yunhuan Yang, Jing Tan, and Haoqiang Fan. Realtime-vla v2: Learning to run vlas fast, smooth, and accurate. *arXiv preprint arXiv:2603.26360*, 2026a.

Fan Yang, Peiguang Jing, Kaihua Qu, Ningyuan Zhao, and Yuting Su. Abpolicy: Asynchronous b-spline flow policy for real-time and smooth robotic manipulation. *arXiv preprint arXiv:2602.23901*, 2026b.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. Efficientvla: Training-free acceleration and compression for vision-language-action models. *Advances in Neural Information Processing Systems*, 38:40891–40914, 2026c.

Zebin Yang, Qi Wang, Yunhe Wang, Xiurui Guo, Bo Yu, Shaoshan Liu, Jiafeng Xu, Hao Dong, and Meng Li. Jetson-pi: Towards onboard real-time robot control via foresight-aligned asynchronous inference. *arXiv preprint arXiv:2607.12659*, 2026d.

Zhaoshu Yu, Bo Wang, Pengpeng Zeng, Haonan Zhang, Ji Zhang, Zheng Wang, Lianli Gao, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A survey on efficient vision-language-action models. *arXiv preprint arXiv:2510.24795*, 2025.

Puzhen Yuan, Angyuan Ma, Yunchao Yao, Huaxiu Yao, Masayoshi Tomizuka, and Mingyu Ding. Remac: Self-reflective and self-evolving multi-agent collaboration for long-horizon robot manipulation. *Pattern Recognition*, pp. 114831, 2026.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. *Advances in Neural Information Processing Systems*, 37:56619–56643, 2024.

Dijia Zhan, Xuemiao Xu, Jinyi Li, and Jie Tang. Seam: Smooth execution of action-chunked motion for vision-language-action policies. *arXiv preprint arXiv:2607.04609*, 2026.

Jingxuan Zhang, Yunta Hsieh, Zhongwei Wan, Haokun Lin, Xin Wang, Ziqi Wang, Yingtie Lei, and Mi Zhang. Quantvla: Scale-calibrated post-training quantization for vision-language-action models. *arXiv preprint arXiv:2602.20309*, 2026.

Yongsheng Zhao, Lei Zhao, Baoping Cheng, Gongxin Yao, Xuanzhang Wen, and Han Gao. Vla-rail: A real-time asynchronous inference linker for vla models and robots. *arXiv preprint arXiv:2512.24673*, 2025.

Yuchi Zhao, Miroslav Bogdanovic, Arjun Sohal, Liyu Tao, Kourosh Darvish, Alán Aspuru-Guzik, Florian Shkurti, and Animesh Garg. Dynamic execution horizon prediction for chunk-based robot policies. *arXiv preprint arXiv:2606.11408*, 2026.

Zihao Zheng, Hangyu Cao, Sicheng Tian, Jiayu Chen, Maoliang Li, Xinhao Sun, Hailong Zou, Zhaobo Zhang, Xuanzhe Liu, Donggang Cao, et al. Dyq-vla: Temporal-dynamic-aware quantization for embodied vision-language-action models. *arXiv preprint arXiv:2603.07904*, 2026a.

Zihao Zheng, Zhihao Mao, Maoliang Li, Jiayu Chen, Xinhao Sun, Zhaobo Zhang, Donggang Cao, Hong Mei, and Xiang Chen. Kerv: Kinematic-rectified speculative decoding for embodied vla models. *arXiv preprint arXiv:2603.01581*, 2026b.