ASYNCVLA: ASYNCHRONOUS FLOW MATCHING FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models have recently emerged as a powerful paradigm for building generalist robots. However, traditional VLA models that generate actions through flow matching (FM) typically rely on rigid and uniform time schedules, i.e., synchronous FM (SFM). Without selective self-correction conditioned on retained action predictions, errors in SFM can cascade during long-horizon tasks. In this work, we propose asynchronous flow matching VLA (AsyncVLA), a novel framework that introduces temporal flexibility in asynchronous FM (AFM) and enables self-correction in action generation. AsyncVLA breaks from the vanilla SFM in VLA models by generating the action tokens in a non-uniform time schedule with action context awareness. Besides, our method introduces the confidence rater to extract confidence of the initially generated actions, enabling the model to selectively refine inaccurate action tokens before execution. Moreover, we propose a unified training procedure for SFM and AFM that endows a single model with both modes, improving KV-cache utilization. Extensive experiments on robotic manipulation benchmarks demonstrate that AsyncVLA is data-efficient and exhibits self-correction ability. AsyncVLA outperforms the evaluated baselines across both simulation and real-world evaluations.

1 INTRODUCTION

Training generalist robot policies that integrate perception, language, and low-level control remains a central challenge for embodied intelligence (Reed et al., 2022; Xu et al., 2024; Ma et al., 2024; Sapkota et al., 2025; Kim et al., 2024b; Li et al., 2025b; 2024b; Chi et al., 2023). Vision-language-action (VLA) models address this by leveraging large-scale vision-language corpora and robot demonstrations to ground broad semantics into executable control (Brohan et al., 2022; Huang et al., 2023; Bousmalis et al., 2023; Ahn et al., 2022; Fu et al., 2024; Yu et al., 2025; Zang et al., 2025; Liu et al., 2025). To improve success rates and efficiency, follow-up work advances architectures and action generation through better spatiotemporal modeling (Li et al., 2025c; Chen et al., 2025b; Lv et al., 2025; Wu et al., 2025; Shi et al., 2025), parameter-efficient adaptation (Wang et al., 2025), improved tokenization (Pertsch et al., 2025; Goyal et al., 2025), high-throughput execution (Kim et al., 2025; Shukor et al., 2025), and interleaving intermediate reasoning with action prediction (Wen et al., 2025b; Zhao et al., 2025; Zhong et al., 2025; Huang et al., 2025b; Wen et al., 2025a). Building on these advances, recent VLA models incorporate self-correction to improve reliability under uncertainty. Imitating fast and slow systems in the human brain (Li et al., 2025d; Jiang et al., 2025), SC-VLA (Li et al., 2024a) pairs a fast action head with a slow self-correction module to detect failures. Enhanced by reinforcement learning, RB-VLA (Li et al., 2025a) applies a dual-pathway loop for in-situ adaptation. Inspired by diffusion large language models (DLLMs) (Nie et al., 2025; Swerdlow et al., 2025; Cheng et al., 2025), discrete-diffusion VLA (Liang et al., 2025) brings masked-token denoising and secondary remasking into VLA for adaptive decoding. LLaDA-VLA (Wen et al., 2025c), dVLA (Wen et al., 2025a), and UD-VLA (Chen et al., 2025a) further employ multi-modal chain of thought (CoT) with prefix attention and KV-cache.

Flow-matching VLA policies (Black et al., 2025b;a; Zhai et al., 2025; Qu et al., 2025a; StarVLA Community, 2026; Zhang et al., 2024; Dong et al., 2025) motivate our synchronous FM (SFM) baseline, which shares a denoising-time schedule across action tokens. Joint attention supplies context among noisy actions, but this baseline does not select unreliable tokens for a second correction stage with other action values held fixed. Recent streaming and adaptive-flow methods (Li et al., 2026; Han

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.

et al., 2026) further motivate distinguishing within-chunk correction from other forms of inference adaptation.

Our insight is to introduce token-wise asynchrony into FM (Lipman et al., 2022; Gao et al., 2025; Zheng et al., 2025a): regenerate uncertain actions while retaining higher-confidence predictions as context. This separates initial chunk generation from selective correction rather than repeatedly denoising the full chunk.

In this paper, we propose asynchronous flow matching VLA (AsyncVLA), a novel VLA framework that employs the initial SFM and the subsequent asynchronous FM (AFM), enabling confidence-aware robot action generation with self-correction. Instead of a fixed and uniform time schedule in the denoising process, AsyncVLA adaptively schedules its AFM time steps, performing regeneration on action tokens with low confidence. Specifically, we propose a confidence rater that evaluates the confidence of each action token generated by SFM. AsyncVLA leverages these confidence signals to trigger asynchronous self-correction, enabling the model to selectively revisit and refine low-confidence parts of its action plan before execution. Moreover, the first-round generated actions with relatively high confidence provide context information that facilitates correcting actions with relatively low confidence.

2 Related Work

Vision-Language-Action Models VLA policies map visual observations and language to actions. OpenVLA, RT-1, and VLA-0 use autoregressive decoding (Kim et al., 2024a; Brohan et al., 2022; Goyal et al., 2025), whereas Octo uses diffusion decoding (Ghosh et al., 2024). CoT-VLA and FlowVLA generate visual subgoals before actions (Zhao et al., 2025; Zhong et al., 2025). OpenVLA-OFT supports parallel action decoding (Kim et al., 2025). Continuous-action models including π_0, $\pi_{0.5}$, WALL-OSS, and EO-1 use FM (Black et al., 2025b;a; Zhai et al., 2025; Qu et al., 2025a).

Asynchronous and Adaptive Inference RTC (Black et al., 2025c) overlaps execution with next-chunk generation and uses inpainting to preserve continuity. FlashVLA (Li et al., 2026) performs streaming denoising across chunks at staggered noise levels. AdaVLA (Han et al., 2026) adapts flow-solver progression using trajectory curvature and also prunes MLP computation. Attention-based adaptive chunking (Xu et al., 2026) selects the action prefix to execute. These mechanisms concern execution overlap, cross-chunk schedules, or solver/execution budgets. AsyncVLA focuses on within-chunk, confidence-selected regeneration conditioned on retained action values.

Self-Correction in VLA Models Correction mechanisms include human-assisted reflection (Sun et al., 2025), reflective planning (Feng et al., 2025), fast–slow policies (Li et al., 2025d; 2024a), and RL-based adaptation (Li et al., 2025a). Discrete-diffusion VLAs (Liang et al., 2025; Wen et al., 2025a;c; Chen et al., 2025a) build on DLLMs (Nie et al., 2025; Swerdlow et al., 2025; Cheng et al., 2025) for masked-token generation and remasking. More recent work addresses calibrated task-success confidence with VLAConf (Huang et al., 2026) and contrast-set-trained, conformally calibrated failure detection with SAFECAST (Rajaprakash et al., 2026). These confidence models monitor eventual success or failure; AsyncVLA instead uses demonstration-relative token scores to select continuous actions for regeneration, without treating those scores as calibrated task-success probabilities. Our empirical comparisons are restricted to the baselines explicitly reported in the tables.

3 Methodology

We introduce AsyncVLA, a VLA model enhanced by AFM. We start by introducing the self-correction mechanism of AFM in Sec. 3.1, followed by the confidence rater that determines the positions of masked action tokens in Sec. 3.2, and the overall training procedures in Sec. 3.3.

3.1 Asynchronous Flow Matching

We formulate the robot policy as a VLA model in a synergistic structure of VLM backbone and FM action head. The model can flexibly generate continuous action chunks whose length is denoted

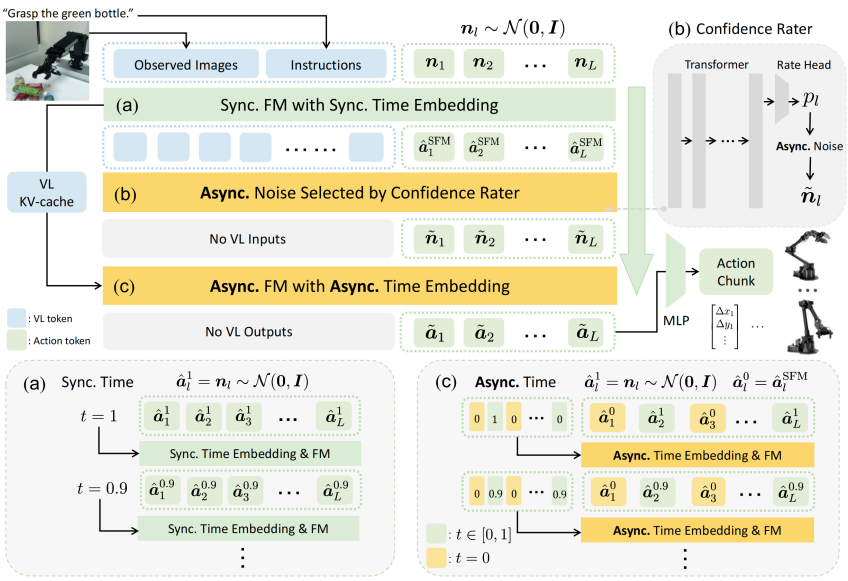


Figure 1: Overview of the AsyncVLA framework that comprises three components: (a) SFM applies a uniform time schedule t across all action tokens, generating them synchronously from noise ($t = 1$) to action ($t = 0$). (b) Confidence rater estimates the actions' token-level confidence and mask the low-confidence actions by selecting asynchronous noise for AFM. (c) AFM dynamically assigns individual FM time to each action token, allowing for selective and non-uniform regeneration based on the actions' confidence. SFM and AFM share a single unified model with the same parameters.

by L. The FM velocity for action generation can be written as $V_\theta\left(\boldsymbol{o}_t, \ell, \hat{\boldsymbol{a}}^\tau_{t:t+L}\right)$, where $\boldsymbol{o}_t = \left[\boldsymbol{I}_t^{(1)}, \ldots, \boldsymbol{I}_t^{(n)}, \boldsymbol{q}_t\right]$ consists of multi-view image observations and robot state at time t, the language context ℓ is the embodied task instructions, and $\hat{\boldsymbol{a}}^\tau_{t:t+L}$ is the partially denoised action chunk at FM time τ.

As shown in Figure 1, AsyncVLA consists of 3 sequential parts: SFM, the confidence rater, and AFM. SFM and AFM share the same model that is trained in a unified training procedure. For input, each token may correspond to a text token, an image patch token, a robot state token, or a partially denoised action token. For output, we employ an FM head to generate continuous action tokens.

Asynchronous Flow Matching Inference During inference of AFM, the model masks part of the action tokens generated by SFM, whose positions are denoted by the mask $\boldsymbol{m} \in \{0, 1\}^L$. The element of $\boldsymbol{m}$ is 1 if the corresponding action token is masked and is 0 otherwise. In AFM generation, the unmasked tokens remain unchanged, while the masked tokens are updated using the forward Euler rule as:

$$\hat{\boldsymbol{a}}^{\tau-\delta}_{t:t+L} = \hat{\boldsymbol{a}}^\tau_{t:t+L} - \delta V_\theta\left(\boldsymbol{o}_t, \ell, \hat{\boldsymbol{a}}^\tau_{t:t+L}\right) \odot \boldsymbol{m}, \tag{1}$$

where $\odot$ denotes the token-wise Hadamard product and δ denotes the time step size. For $\tau = 1$, we design the starting asynchronous noise $\hat{\boldsymbol{a}}^1_{t:t+L} = [\tilde{\boldsymbol{n}}_{t+1}, \cdots, \tilde{\boldsymbol{n}}_{t+L}]$ as:

$$\tilde{\boldsymbol{n}}_l = \begin{cases} \hat{\boldsymbol{a}}_l^{\mathrm{SFM}}, & \text{if } m_l = 0, \\ \boldsymbol{n}_l \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), & \text{if } m_l = 1, \end{cases} \tag{2}$$

where m_l is the l-th element of $\boldsymbol{m}$ with $l = t+1, \cdots, t+L$, and $\hat{\boldsymbol{a}}_l^{\mathrm{SFM}}$ is the action predicted by the previous SFM. In AFM, the model incorporates information from SFM-estimated action tokens even when the regenerated tokens remain in an early noisy stage, thus producing more accurate actions. Since SFM can be regarded as a fully-masked special case of AFM, we employ the same model for both SFM and AFM modes. Therefore, the VL KV-cache generated in SFM can be directly reutilized in AFM. In this way, we save the burden of repeatedly processing the VL tokens and thus significantly improve the model's inference efficiency for real-time control. The procedure of AFM inference is summarized in Algorithm 1.

Asynchronous Time Embedding in AFM To distinguish masked from unmasked action tokens, we propose the asynchronous time embedding module. Denote the dimension of the VLM's hidden states as d. At FM time τ, we apply the sinusoidal encoding function $\mathcal{S}(\cdot)$ to map $\tau\boldsymbol{m}$ to the asynchronous time-embedding matrix $\mathcal{S}(\tau\boldsymbol{m}) \in \mathbb{R}^{L\times d}$. We then concatenate $\mathcal{S}(\tau\boldsymbol{m})$ and the linearly projected noisy action $\mathcal{P}(\hat{\boldsymbol{a}}^{\tau}_{t:t+L}) \in \mathbb{R}^{L\times d}$ along the last dimension and yield $\hat{\boldsymbol{h}}^{\tau}_{t:t+L} \in \mathbb{R}^{L\times 2d}$. Finally, a multi-layer perceptron (MLP) is utilized to project $\hat{\boldsymbol{h}}^{\tau}_{t:t+L}$ to the asynchronous time-embedded action hidden state $\hat{\boldsymbol{x}}^{\tau}_{t:t+L} \in \mathbb{R}^{L\times d}$. With the same hidden dimension as the VLM, $\hat{\boldsymbol{x}}^{\tau}_{t:t+L} \in \mathbb{R}^{L\times d}$ is concatenated with VL hidden states and then sent into the VLM's transformer backbone. Following (Zhao et al., 2025), we apply full attention for action generation.

3.2 Confidence Rater

Since AsyncVLA lacks a dedicated output head for action-token logits, it is hard to directly estimate the model's confidence based on token probability. Thus, we individually design a confidence rater to estimate the confidence of the actions. The confidence rater takes the final-layer hidden states of VL tokens concatenated with the first-round actions generated by SFM as input, and evaluates the confidence of the l-th action token as $p_l \in (0, 1)$, $l = t + 1, \cdots, t + L$.

The confidence rater consists of several transformer layers and a final linear layer as its rate head. The action tokens are projected into the embedding space of VL tokens using a linear layer. The transformer layers apply full attention, such that the confidence can be calculated according to the VL information and the context actions before or after the evaluated action token. The rate head projects the hidden states to a scalar and employs Sigmoid function to generate p_l. Using p_l, we generate the l-th element of the mask as:

$$m_l = \mathbb{1}\{p_l < T\}, \quad l = t+1, \ldots, t+L, \tag{3}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function and $T \in (0, 1)$ is a predefined confidence threshold that controls the number of masked tokens. In Equation (3), a self-adaptive number of action tokens will be masked according to the actions' confidence, which offers better flexibility than other strategies such as Top-K selection. This rule regenerates lower-confidence tokens while retaining higher-confidence predictions as context.

3.3 Training Procedures

Unified Training for SFM and AFM In order to employ a single model to realize both SFM and AFM inference, we propose a unified training procedure that treats SFM as a fully-masked special case of AFM. The VLM backbone and FM head are jointly trained by minimizing the following end-to-end AFM velocity prediction loss on masked tokens:

$$\mathcal{L} = \mathbb{E}_{\tau}\left\{\left\|\left[V_{\theta}\left(\boldsymbol{o}_t, \ell, \hat{\boldsymbol{a}}^{\tau}_{t:t+L}\right) - \boldsymbol{u}_{t:t+L}\right] \odot \boldsymbol{m}\right\|^2\right\}, \tag{4}$$

where $\boldsymbol{u}_{t:t+L} = \boldsymbol{n}_{t:t+L} - \boldsymbol{a}_{t:t+L}$ denotes the ground-truth velocity with Gaussian noise $\boldsymbol{n}_{t:t+L} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $\hat{\boldsymbol{a}}^{\tau}_{t:t+L}$ denotes the intermediate asynchronous noisy action, computed as

$$\tilde{\boldsymbol{a}}_{t:t+L} = \boldsymbol{a}_{t:t+L} + \boldsymbol{\epsilon}_{t:t+L} \odot (\boldsymbol{1} - \boldsymbol{m}), \qquad \boldsymbol{\epsilon}_{t:t+L} \sim \mathcal{N}(\boldsymbol{0}, \sigma_c^2 \boldsymbol{I}), \tag{5}$$

$$\hat{\boldsymbol{a}}^{\tau}_{t:t+L} = \tilde{\boldsymbol{a}}_{t:t+L} - \tau\big(\tilde{\boldsymbol{a}}_{t:t+L} - \boldsymbol{n}_{t:t+L}\big) \odot \boldsymbol{m}. \tag{6}$$

where $\boldsymbol{\epsilon}_{t:t+L} \sim \mathcal{N}(\boldsymbol{0}, \sigma_c^2 \boldsymbol{I})$ is injected only on unmasked context tokens. That is, for positions with $m_i = 0$, the model conditions on slightly perturbed action tokens rather than perfectly clean ground-truth actions. This design mitigates the train-test mismatch between training, where unmasked tokens would otherwise be clean action tokens, and inference, where they are replaced by imperfect SFM predictions. It therefore reduces exposure bias and makes the model more robust to realistic context errors during AFM refinement.

In training, each element of $\boldsymbol{m}$ is identically and independently sampled from Bernoulli(y) with the pre-sampled probability $y \sim \mathcal{U}(0, 1)$. Following (Black et al., 2025b), we sample the FM time τ from the Beta distribution Beta$(1.5, 1)$, which emphasizes noisier time steps close to 1. Note that when the sampled $\boldsymbol{m}$ is an all-1 vector, the AFM loss in Equation (4) degenerates to the vanilla SFM loss. Such training samples guarantee the unified model's ability of both AFM and SFM inference. The

Table 1: LIBERO task performance results evaluated by success rates. OpenVLA-OFT (Con./Dis.) refers to OpenVLA-OFT with continuous or discrete action. All models are fine-tuned on the combined 4 task suites as a whole, instead of training 4 individual VLA models.

Model	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-Long	Avg.
WorldVLA (Cen et al., 2025)	87.6	96.2	83.4	60.0	81.8
TraceVLA (Zheng et al., 2025b)	84.6	85.2	75.1	54.1	74.8
SpatialVLA (Qu et al., 2025b)	88.2	89.9	78.6	55.5	78.1
π_0-FAST (Pertsch et al., 2025)	96.4	96.8	88.6	60.2	85.5
π_0 (Black et al., 2025b)	96.8	98.8	95.8	85.2	94.2
$\pi_{0.5}$ (Black et al., 2025a)	**98.8**	98.2	98.0	92.4	96.9
OpenVLA-OFT (Con.) (Kim et al., 2025)	96.9	98.1	95.5	91.1	95.4
OpenVLA-OFT (Dis.) (Kim et al., 2025)	96.2	98.2	95.6	92.0	95.5
GR00T-N1 (NVIDIA et al., 2025)	94.4	97.6	93.0	90.6	93.9
UD-VLA (Chen et al., 2025a)	94.1	95.7	91.2	89.6	92.7
Discrete-Diffusion VLA (Liang et al., 2025)	97.2	98.6	97.4	92.0	96.3
dVLA (Wen et al., 2025a)	97.4	97.9	98.2	92.2	96.4
AsyncVLA (Ours)	98.4	**99.2**	**98.6**	**93.4**	**97.4**

randomly sampled $\boldsymbol{m}$ also equivalently plays the role of data augmentation that improves training data efficiency. The unified training procedure for AFM and SFM is summarized in Algorithm 2.

Training Confidence Rater After the VLA backbone is fully trained for SFM and AFM using Equation (4), we train the confidence rater with the frozen VLA backbone in an end-to-end manner. Since the actions generated by SFM do not provide direct signals that indicate the confidence of the model, we need to deliberately design the pseudo labels of the confidence rater. Define the sequence of token-wise mean-squared errors (MSEs) of the SFM-generated action chunk as $e_{t:t+L}$. Since the model should have higher confidence on tokens with smaller MSE and vice versa, we define the pseudo labels of the confidence rater as:

$$q_{t:t+L} = 1 - \alpha - \beta \frac{e_{t:t+L} - \min\{e_l\}}{\max\{e_l\} - \min\{e_l\} + \epsilon}, \tag{7}$$

where α and β are hyper-parameters that control the region of the pseudo labels in roughly $[1 - \alpha - \beta, 1 - \alpha]$, e.g., $[0.01, 0.99]$ if $\alpha = 0.01$ and $\beta = 0.98$, and ϵ is a small scalar that prevents the denominator being 0. In Equation (7), we alleviate the gradient vanishing problem caused by the final Sigmoid function, by preventing the labels from being extremely close to 0 or 1. To evaluate the confidence in a relative manner, $\max\{e_l\}$ and $\min\{e_l\}$ denote the maximum and minimum MSE in the action chunk, respectively. We thus assign the relatively accurate action tokens high confidence and utilize their context information to regenerate the action tokens with low confidence. When training the confidence rater, we set the loss function to the MSE between the confidence rater's output and $q_{t:t+L}$. We empirically find that the labeling method in Equation (7) outperforms labeling by the task success indicator (TSI), i.e., $q_{t:t+L} = 1$ if the task is successful and $q_{t:t+L} = 0$ if the task is failed. We will analyze the reason through the ablation experiments in Section 4.3.

4 Simulation Experiments

4.1 Experimental Setup

We adopt Qwen2.5-VL-3B-Instruct (Bai et al., 2025) as the VLM backbone and augment it with an FM action head along with a confidence rater. Our AsyncVLA is pretrained on the Open X-Embodiment dataset (O'Neill et al., 2023) and is subsequently finetuned for different evaluation tasks on the corresponding datasets, including LIBERO (Liu et al., 2023), Bridge-V2 (Walke et al., 2023), and Fractal (Brohan et al., 2022). In the data pre-processing stage, we mark the pause intervals in trajectories and exclude those action tokens from loss computation. The learning rate is set as 1×10^{-4} for both the language model backbone and the FM action head, and is set as 2×10^{-5} for the vision encoder. For unmasked context corruption in training, the noise scale is $\sigma_c = 0.05$. We use a 10-step denoising schedule for both SFM and AFM during inference. The backbone of the confidence rater is a 4-layer transformer with a linear layer as the rate head, which sums up to 308M parameters and takes up 7.56% of the total 4.08B parameters of the overall VLA model. In the mask generation, we set the confidence threshold as $T = 0.5$. When training the confidence rater, we set $\alpha = 0.01$, $\beta = 0.98$, and $\epsilon = 1 \times 10^{-6}$.

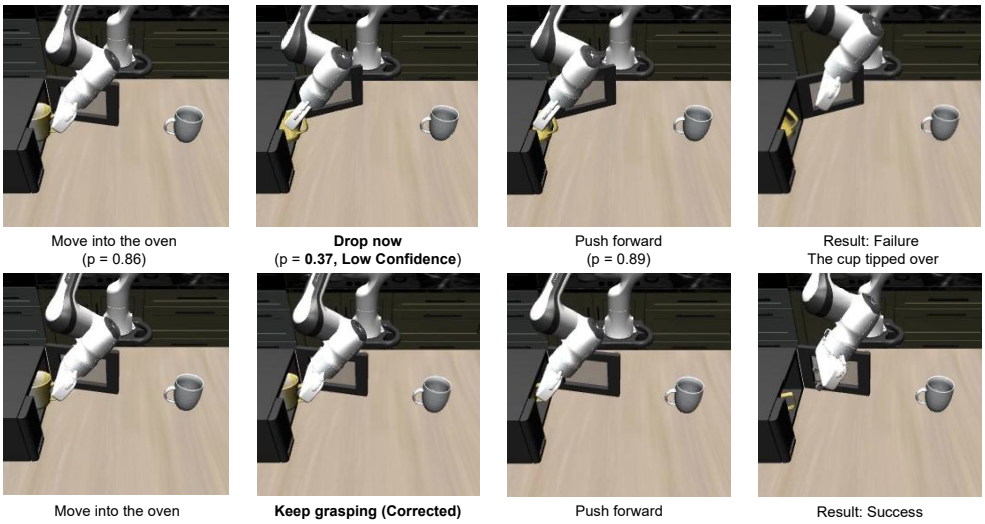


Figure 2: Illustration of self-correction ability in AsyncVLA on the LIBERO-Long task suite. The top row shows the first-round actions generated by SFM, and the bottom row shows the second-round actions regenerated by the following AFM.

Table 2: Comparison of different VLA models on the WidowX Robot benchmark conducted in SimplerEnv (Li et al., 2024d).

Model	Put Spoon on Towel	Put Carrot on Plate	Stack Cubes	Put Eggplant in Basket	Avg.
OpenVLA (Kim et al., 2024a)	0	0	0	4.1	1.0
SpatialVLA (Qu et al., 2025b)	20.8	20.8	25.0	70.8	34.4
Magma (Yang et al., 2025)	37.5	29.2	20.8	**91.7**	44.8
π_0-FAST (Pertsch et al., 2025)	29.1	21.9	10.8	66.6	32.1
Octo-Base (Ghosh et al., 2024)	12.5	8.3	0	43.1	16.0
Octo-Small (Ghosh et al., 2024)	47.2	9.7	4.2	56.9	29.5
RoboVLM (Li et al., 2024c)	45.8	20.8	4.2	79.2	37.5
$\pi_{0.5}$ (Black et al., 2025a)	49.3	64.7	44.7	69.7	57.1
ThinkAct (Huang et al., 2025a)	58.3	37.5	8.7	70.8	43.8
Discrete-Diffusion VLA (Liang et al., 2025)	37.5	29.2	20.8	–	–
UD-VLA (Chen et al., 2025a)	58.3	62.5	54.1	75.0	62.5
AsyncVLA (Ours)	**70.8**	**66.7**	**58.3**	87.5	**70.8**

4.2 Evaluation Results

LIBERO Benchmark We finetune and evaluate AsyncVLA and baselines on the LIBERO benchmark (Liu et al., 2023). Results are presented in Table 1, where we evaluate the models over 500 trials per task suite (10 tasks $\times$ 50 episodes). AsyncVLA achieves the highest average success rate and the highest success rates on three of the four task suites; $\pi_{0.5}$ performs best on LIBERO-Spatial. By analyzing rollout videos of successful cases and comparing them with the trajectory generated by SFM in the same task, we find that AsyncVLA demonstrates the ability of self-correction, particularly in challenging tasks where first-round actions contain errors.

Self-Correction Ability We illustrate AsyncVLA's self-correction ability on LIBERO-Long task in Figure 2. The top row shows the actions generated by SFM, and the bottom row shows the actions regenerated by the following AFM. Since the confidence rater gives a low confidence on the "drop now" action, this action token is remasked while others are not. AsyncVLA takes the other high-confidence action tokens into account as context and finds that before the "push forward" action, the correct action should be "keep grasping". After AFM's regeneration, the action with low confidence is corrected, and the task is successfully completed. It demonstrates that AFM with self-correction enhances AsyncVLA's robustness to perturbations from first-round generated erroneous actions with large deviation.

WidowX Robot Benchmark We evaluate AsyncVLA and baselines on the WidowX Robot benchmark after further finetuning on the Bridge-V2 dataset (Walke et al., 2023). We test across four generalization categories with environmental variations, and report the results in Table 2. AsyncVLA

Table 3: Comparison on the Google Robot benchmark under both visual matching (M) and variant aggregation (A) settings. The task "O/C Drawer" is short for "open or close the (top/middle/bottom) drawer". Evaluation is conducted in SimplerEnv (Li et al., 2024d).

Model	Pick Coke		Move Near		O/C Drawer		Put in Drawer		Avg.	
	M	A	M	A	M	A	M	A	M	A
TraceVLA (Zheng et al., 2025b)	28.0	60.0	53.7	56.4	57.0	31.0	0.0	0.0	34.7	36.9
SpatialVLA (Qu et al., 2025b)	86.0	88.0	77.9	72.7	57.4	41.8	0.0	6.3	55.3	52.2
Magma (Yang et al., 2025)	75.0	68.6	53.0	78.5	58.9	59.0	8.3	24.0	48.8	57.5
π_0-FAST (Pertsch et al., 2025)	75.3	77.6	67.5	68.2	42.9	31.3	0.0	0.0	46.4	44.3
RT-2-X (Brohan et al., 2023)	78.7	82.3	77.9	79.2	25.0	35.3	7.4	20.6	47.3	54.4
Octo-Base (Ghosh et al., 2024)	17.0	0.6	4.2	3.1	22.7	1.1	0.0	0.0	11.0	1.2
RoboVLM (Li et al., 2024c)	77.3	75.6	61.7	60.0	43.5	10.6	24.1	0.0	51.7	36.6
π_0 (Black et al., 2025b)	97.9	90.1	78.7	80.7	62.3	27.6	46.6	20.5	71.4	54.7
ThinkAct (Huang et al., 2025a)	92.0	84.0	72.4	63.8	50.0	47.6	–	–	–	–
MolmoAct (Lee et al., 2025)	77.7	76.1	77.1	61.3	60.0	**78.8**	–	–	–	–
Discrete-Diffusion VLA (Liang et al., 2025)	85.4	82.5	67.5	64.6	60.6	23.6	–	–	–	–
AsyncVLA (Ours)	**98.0**	**91.6**	**82.3**	**81.7**	**70.5**	58.0	**50.4**	**26.0**	**75.3**	**64.3**

Table 4: Ablation study on the WidowX Robot benchmark in SimplerEnv (Li et al., 2024d). We use a 10-step denoising schedule in both SFM and AFM unless otherwise specified.

Model	Put Spoon on Towel	Put Carrot on Plate	Stack Cubes	Put Eggplant in Basket	Avg.
w/o Unified Training	4.2	8.3	0.0	16.7	7.3
w/o AFM (10 steps)	58.3	54.2	33.3	45.8	47.9
w/o AFM (20 steps)	58.3	54.2	37.5	54.2	51.1
w/o Confidence rater	66.7	54.2	54.2	75.0	62.5
w/ TSI labeling	66.7	58.3	54.2	79.2	64.6
w/ Delta refinement	66.7	58.3	45.8	75.0	61.5
w/ Direct refinement	66.7	54.2	50.0	79.2	62.5
AsyncVLA (Ours)	**70.8**	**66.7**	**58.3**	**87.5**	**70.8**

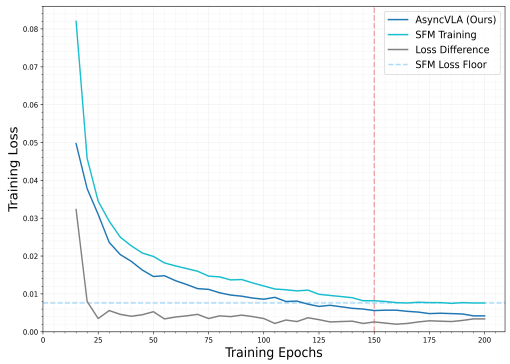
achieves the best overall performance, obtaining the highest average success rate (70.8%). Notably, it outperforms all baselines on three out of four tasks, including "put spoon on towel", "put carrot on plate", and "stack cubes", demonstrating strong generalization across diverse manipulation skills. While Magma achieves the best performance on "put eggplant in basket", AsyncVLA remains competitive on this task and maintains consistently strong performance across all categories.

Google Robot Benchmark We evaluate AsyncVLA and the baselines on the Google Robot benchmark after further finetuning on the Fractal dataset (Brohan et al., 2022). As shown in Table 3, AsyncVLA consistently achieves the strongest overall performance, obtaining the highest average success rates under both visual matching (M) and variant aggregation (A) settings. Specifically, AsyncVLA achieves the best results on *Move Near* and *Put in Drawer* across both protocols, and also establishes new highs on *O/C Drawer* under the M setting.

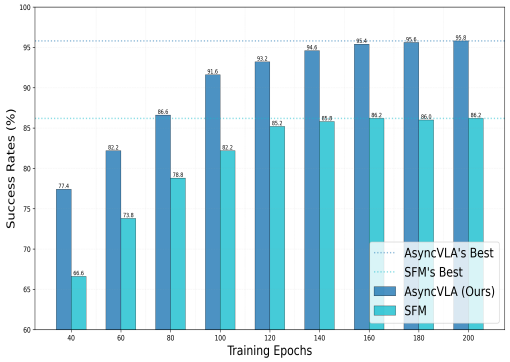
4.3 Ablation Study

We conduct ablation studies on four tasks from the WidowX Robot benchmark. We evaluate eight variants in total: "w/o Unified Training" uses vanilla SFM training but retains SFM-rater-AFM inference; "w/o AFM (10/20 steps)" removes AFM and uses SFM-only inference with 10 or 20 denoising steps; "w/o Confidence rater" keeps AFM but removes the confidence rater, where each action token is masked independently with probability 0.5; "w/ TSI labeling" trains the confidence rater using task-success-indicator (TSI) labels, i.e., $q_{t:t+L} = 1$ for successful trajectories and $q_{t:t+L} = 0$ otherwise; "w/ Delta refinement" replaces the AFM correction stage with a transformer-based refinement module that predicts action deltas; "w/ Direct refinement" replaces the AFM correction stage with a transformer-based refinement module that directly predicts refined actions; and "AsyncVLA (Ours)" denotes the complete method. Unless otherwise specified, both SFM and AFM use a 10-step denoising schedule.

As shown in Table 4, AsyncVLA consistently achieves the best performance on all four tasks, reaching an average success rate of 70.8%. Removing unified training causes a dramatic drop to



(a) Training loss curve comparison when only a quarter of the LIBERO-Spatial dataset is used for training.



(b) Success rate comparison in the training process. Evaluation is conducted on LIBERO-Spatial test suite.

7.3%, showing that aligning training with both synchronous and asynchronous inference is essential. Without AFM, performance drops to 47.9% with 10-step SFM and only recovers to 51.1% even with 20 SFM steps, indicating that simply increasing first-stage denoising is much less effective than adding a dedicated correction stage. Introducing AFM without a learned confidence rater already improves the average success rate to 62.5%, a gain of 14.6 points over SFM-only inference with 10 steps, which verifies the value of asynchronous self-correction itself. Both Delta refinement (61.5%) and Direct refinement (62.5%) remain below AsyncVLA despite sharing its confidence mask. This comparison isolates the correction mechanism rather than confidence-based selection; Appendix F.2 reports a separate cardinality-matched selection control. Finally, training the confidence rater with TSI labeling improves over random masking and the two generic refinement baselines, but still underperforms the full method (64.6% vs. 70.8%). The reason is that TSI provides only trajectory-level supervision and propagates the same binary outcome to all actions in the sequence, regardless of whether a particular step is helpful or harmful. Such coarse labels obscure token-level action quality and make it difficult for the confidence rater to learn fine-grained correction cues. In contrast, the dense labeling strategy in Equation (7) provides more precise supervision and leads to the strongest overall performance.

4.4 Training Data Efficiency

To demonstrate the data efficiency of the proposed unified training method in data-constrained settings, we separately train the models using AsyncVLA's unified training method or vanilla SFM's training method. The models are trained for 200 epochs with constant learning rates using a quarter of the LIBERO-Spatial dataset. The comparison of training loss curves is shown in Figure 3a, where all plotted loss is normalized per unmasked token. The training loss of AsyncVLA decreases faster and is remarkably and constantly lower than the training loss of SFM. Even when the training loss of SFM reaches a floor of 0.0076 and remains almost unchanged after 150 epochs, the AsyncVLA's loss continues to decrease and ultimately reaches 0.0042 at the 200-th epoch.

We evaluate the success rates of AFM and SFM on LIBERO-Spatial test suite every 20 epochs in Figure 3b. AsyncVLA constantly outperforms SFM by at least 7.8%. When the success rate of SFM actually stops increasing and fluctuates around 86.2% after 140 epochs, the success rate of AsyncVLA still stably improves and finally reaches 95.8%. This demonstrates that AsyncVLA better exploits the training data with longer training epochs in data-constrained settings. Such exploitation is attributed to the training data efficiency of the proposed unified training method that equivalently plays the role of data augmentation.

5 Real-World Experiments

To further validate our framework, we conduct extensive real-world experiments on the AgileX PiPER robot, as shown in Figure 4. The robot is equipped with a fixed front-view camera and a wrist camera for visual observation, and receives natural-language task instructions together with proprioceptive states as policy inputs. We evaluate four manipulation tasks covering precise grasping,



Figure 4: Real-world evaluated tasks on the AgileX PiPER robot. From left to right: (**1**) placing a carrot into a bowl; (**2**) taking a pen out of a pen holder; (**3**) pouring water from a cup into a bowl; and (**4**) placing a spoon onto a plate.

Table 5: Real-world success rates (%) on four manipulation tasks. Each task is evaluated over 50 trials on the AgileX PiPER robot.

Model	Carrot → Bowl	Pen Extraction	Pour Water	Spoon → Plate	Avg.
OpenVLA-OFT	82.0	68.0	46.0	66.0	65.5
π_0	72.0	76.0	48.0	64.0	65.0
$\pi_{0.5}$	86.0	78.0	68.0	76.0	77.0
AsyncVLA (Ours)	**94.0**	**86.0**	**82.0**	**86.0**	**87.0**

object extraction, liquid pouring, and object placement. For each task, we conduct 50 trials under mild variations in object poses and initial states, and report the task-wise and average success rates.

As shown in Table 5, AsyncVLA achieves the best performance on all four tasks, reaching an average real-world success rate of 87.0%. The improvements are consistent across tasks and particularly pronounced on error-sensitive scenarios such as *pour water* and *pen extraction*, where slight action deviations can easily accumulate and cause failure. Together, these results support AsyncVLA's effectiveness after task-specific fine-tuning. The configuration-paired analysis in Appendix H.4 further characterizes the gains over $\pi_{0.5}$ within this controlled real-world evaluation.

6 Limitations

Although these gains support selective correction, their interpretation remains tied to the method's assumptions and evaluation scope. Because confidence is relative within each chunk, globally poor predictions may still be retained as context; Gaussian context noise also only approximates structured SFM errors. On the empirical side, rater diagnostics reuse the checkpoint-selection split without establishing task-success calibration, while paired comparisons concern fixed checkpoints rather than training-seed variability. These qualifications matter beyond controlled trials: stronger visual shifts reduce success, and the real-world evidence covers only one robot and four tasks. Broader deployment therefore warrants further validation of these gains under new conditions. Update-matched rater training, targeted-AFM step sensitivity, and transfer to non-robotic generative tasks, including image generation, also remain open.

7 Conclusion

We present AsyncVLA, a two-stage framework for confidence-aware continuous action generation. A learned rater scores the initial SFM predictions, and AFM selectively regenerates low-confidence tokens while retaining the remaining actions as context. Unified SFM/AFM training supports both stages in a shared policy and enables reuse of the visual-language KV-cache. The reported controls favor targeted correction over the tested synchronous-step and random-mask alternatives. Against SFM, AsyncVLA records 25 recoveries and 3 regressions on 96 paired WidowX configurations; the cardinality-matched random-mask control supports confidence-based selection. The PiPER evaluation reports a 10.0-percentage-point gain over $\pi_{0.5}$ on 200 paired configurations. In the quarter-data LIBERO-Spatial study, the reported success rates reach 95.8% for AsyncVLA and 86.2% for SFM after 200 epochs. Future work should examine absolute-error-aware confidence under deployment shift, training-seed stability, and integration with cross-chunk streaming denoising or adaptive execution.

REPRODUCIBILITY STATEMENT

Code and training/evaluation scripts are available at https://anonymous.4open.science/r/AsyncVLA/. Section 4.1 and Appendix E detail the datasets, preprocessing, model, and training setup. Appendix C provides pseudocode; Appendices F–H provide additional experiments, statistical procedures, and the real-robot protocol.

AI USE STATEMENT

Generative AI assisted language editing, literature discovery and bibliographic verification, LaTeX formatting, and drafting the reproducibility and AI use statements. Automated manuscript feedback covered clarity, methodology, and experimental reporting. For this revision, AI helped organize author-provided results, check arithmetic, and qualify claims; it did not generate new experimental observations. The authors are responsible for verifying the data, analyses, and final content.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. URL https://arxiv.org/abs/2204.01691.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40, 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025b. doi: 10.15607/RSS.2025.XXI.010.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems*, 2025c. URL https://neurips.cc/virtual/2025/poster/117747.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Żołna, Scott Reed, Sergio Gómez Colmenarejo, Jon Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Tom Rothörl, José Enrique Chen, Yusuf Aytar, Dave Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. RoboCat: A self-improving generalist agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023. URL https://arxiv.org/abs/2306.11706.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael

Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. URL https://arxiv.org/abs/2212.06817.

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

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. WorldVLA: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, Jun 2025.

Jiayi Chen, Wenxuan Song, Pengxiang Ding, Ziyang Zhou, Han Zhao, Feilong Tang, Donglin Wang, and Haoang Li. Unified diffusion VLA: Vision-language-action model via joint discrete denoising diffusion process. *arXiv preprint arXiv:2511.01718*, 2025a.

Xinyi Chen, Yilun Chen, Yanwei Fu, Ning Gao, Jiaya Jia, Weiyang Jin, Hao Li, Yao Mu, Jiangmiao Pang, Yu Qiao, Yang Tian, Bin Wang, et al. InternVLA-M1: A spatially guided vision-language-action framework for generalist robot policy. *arXiv preprint arXiv:2510.13778*, 2025b.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Shaoqi Dong, Chaoyou Fu, Haihan Gao, Yi-Fan Zhang, Chi Yan, Chu Wu, Xiaoyu Liu, Yunhang Shen, Jing Huo, Deqiang Jiang, Haoyu Cao, Yang Gao, Xing Sun, Ran He, and Caifeng Shan. VITA-VLA: Efficiently teaching vision-language models to act via action expert distillation. *arXiv preprint arXiv:2510.09607*, 2025.

Yunhai Feng, Jiaming Han, Zhuoran Yang, Xiangyu Yue, Sergey Levine, and Jianlan Luo. Reflective planning: Vision-language models for multi-stage long-horizon robotic manipulation. *arXiv preprint arXiv:2502.16707*, 2025.

Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

Dechen Gao, Boqi Zhao, Andrew Lee, Ian Chuang, Hanchu Zhou, Hang Wang, Zhe Zhao, Junshan Zhang, and Iman Soltani. Vita: Vision-to-action flow matching policy. *arXiv preprint arXiv:2507.13231*, 2025.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090.

Ankit Goyal, Hugo Hadfield, Xuning Yang, Valts Blukis, and Fabio Ramos. VLA-0: Building state-of-the-art VLAs with zero modification. *arXiv preprint arXiv:2510.13054*, 2025.

Sunghwan Han, Youngtae Han, and Youngmin Yi. AdaVLA: Adaptive step flow matching for training-free acceleration of vision-language-action models. *arXiv preprint arXiv:2608.29208*, 2026. URL https://arxiv.org/abs/2608.29208v1.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. ThinkAct: Vision-language-action reasoning via reinforced visual latent planning. *arXiv preprint arXiv:2507.16815*, 2025a.

Dehao Huang, Aoxiang Gu, Chengjie Zhang, Bolin Zou, Wenlong Dong, Zilang Cen, Yue Wang, and Hong Zhang. VLAConf: Calibrated task-success confidence for vision-language-action models. *arXiv preprint arXiv:2605.29605*, 2026. URL https://arxiv.org/abs/2605.29605v2.

Wenhui Huang, Changhe Chen, Han Qi, Chen Lv, Yilun Du, and Heng Yang. MoTVLA: A vision-language-action model with unified fast-slow reasoning. *arXiv preprint arXiv:2510.18337*, 2025b.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782, 2023. URL https://proceedings.mlr.press/v205/huang23c.html.

Yuhua Jiang, Shuang Cheng, Yihao Liu, Ermo Hua, Che Jiang, Weigao Sun, Yu Cheng, Feifei Gao, Biqing Qi, and Bowen Zhou. Nirvana: A specialized generalist model with task-aware memory mechanism. *arXiv preprint arXiv:2510.26083*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2024a. URL https://proceedings.mlr.press/v270/kim25c.html.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.017.

Yeseung Kim, Dohyun Kim, Jieun Choi, Jisang Park, Nayoung Oh, and Daehyung Park. A survey on integration of large language models with intelligent robots. *arXiv preprint arXiv:2404.09228*, 2024b. URL https://arxiv.org/abs/2404.09228.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

Baicheng Li, Dong Wu, Zike Yan, Xinchen Liu, Lusong Li, Zecui Zeng, and Hongbin Zha. Reflection-based task adaptation for self-improving VLA. *arXiv preprint arXiv:2510.12710*, 2025a. URL https://arxiv.org/abs/2510.12710.

Chenxuan Li, Jiaming Liu, Guanqun Wang, Xiaoqi Li, Sixiang Chen, Liang Heng, Chuyan Xiong, Jiaxin Ge, Renrui Zhang, Kaichen Zhou, and Shanghang Zhang. A self-correcting vision-language-action model for fast and slow system manipulation. *arXiv preprint arXiv:2405.17418*, 2024a. URL https://arxiv.org/abs/2405.17418.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. SimpleVLA-RL: Scaling VLA training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025b.

Peiyan Li, Yixiang Chen, Hongtao Wu, Xiao Ma, Xiangnan Wu, Yan Huang, Liang Wang, Tao Kong, and Tieniu Tan. BridgeVLA: Input-output alignment for efficient 3d manipulation learning with vision-language models. *arXiv preprint arXiv:2506.07961*, 2025c.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024b. URL https://arxiv.org/abs/2411.19650.

Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong, Hanbo Zhang, and Huaping Liu. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*, 2024c.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728, 2024d. URL https://proceedings.mlr.press/v270/li25c.html.

Zekai Li, Jiaming Tang, and Zhijian Liu. FlashVLA: Streaming action decoding for fast and asynchronous VLA inference. *arXiv preprint arXiv:2608.27384*, 2026. URL https://arxiv.org/abs/2608.27384v1.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Xiao Liang, Zhijiang Guo, Le Song, and Cheng-Lin Liu. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025d.

Zhixuan Liang, Yizhuo Li, Tianshuo Yang, Chengyue Wu, Sitong Mao, Liuao Pei, Xiaokang Yang, Jiangmiao Pang, Yao Mu, and Ping Luo. Discrete diffusion VLA: Bringing discrete diffusion to action decoding in vision-language-action policies. *arXiv preprint arXiv:2508.20072*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. URL https://arxiv.org/abs/2210.02747.

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

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can RL bring to VLA generalization? An empirical study. *arXiv preprint arXiv:2505.19789*, 2025.

Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, Delin Qu, Haoming Song, Qizhi Chen, Xiang Deng, and Jiangmiao Pang. F1: A vision-language-action model bridging understanding and generation to actions. *arXiv preprint arXiv:2509.06951*, 2025. URL https://arxiv.org/abs/2509.06951.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied AI. *arXiv preprint arXiv:2405.14093*, 2024. URL https://arxiv.org/abs/2405.14093.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025. URL https://arxiv.org/abs/2502.09992.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Jim Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. *arXiv preprint arXiv:2310.08864*, 2023.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Dong Wang, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, Cheng Ruan, Maoqing Yao, Haoran Yang, Jiacheng Bao, Bin Zhao, and Xuelong Li. EO-1: An open unified embodied foundation model for general robot control. *arXiv preprint arXiv:2508.21112*, 2025a. URL https://arxiv.org/abs/2508.21112.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring spatial representations for visual language-action model. *arXiv preprint arXiv:2501.15830*, 2025b.

Harshitha Rajaprakash, Aditeya Prajapati, Rong Xue, Abrar Anwar, and Jesse Thomason. SAFECAST: Robust failure detection for VLA policies with contrast-set training and calibration. *arXiv preprint arXiv:2608.04246*, 2026. URL https://arxiv.org/abs/2608.04246v1.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Ranjan Sapkota, Yang Cao, Konstantinos I. Roumeliotis, and Manoj Karkee. Vision-language-action (VLA) models: Concepts, progress, applications and challenges. *arXiv preprint arXiv:2505.04769*, 2025. URL https://arxiv.org/abs/2505.04769.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

StarVLA Community. StarVLA: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026. URL https://arxiv.org/abs/2604.05014.

Nan Sun, Yongchang Li, Chenxu Wang, Huiying Li, and Huaping Liu. CollabVLA: Self-reflective vision-language-action model dreaming together with human. *arXiv preprint arXiv:2509.14889*, 2025.

Alexander Swerdlow, Mihir Prabhudesai, Siddharth Gandhi, Deepak Pathak, and Katerina Fragkiadaki. Unified multimodal discrete diffusion. *arXiv preprint arXiv:2503.20853*, Mar 2025.

Homer Rich Walke, Kevin Black, Tony Z. Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A dataset for robot learning at scale. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1723–1736, 2023. URL https://proceedings.mlr.press/v229/walke23a.html.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. VLA-Adapter: An effective paradigm for tiny-scale vision-language-action model. *arXiv preprint arXiv:2509.09372*, 2025.

Junjie Wen, Minjie Zhu, Jiaming Liu, Zhiyuan Liu, Yicun Yang, Linfeng Zhang, Shanghang Zhang, Yichen Zhu, and Yi Xu. dVLA: Diffusion vision-language-action model with multimodal chain-of-thought. *arXiv preprint arXiv:2509.25681*, 2025a.

Junjie Wen, Yichen Zhu, Minjie Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Xiaoyu Liu, Chaomin Shen, Yaxin Peng, and Feifei Feng. DiffusionVLA: Scaling robot foundation models via unified diffusion and autoregression. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66558–66574, 2025b. URL https://proceedings.mlr.press/v267/wen25g.html.

Yuqing Wen, Hebei Li, Kefan Gu, Yucheng Zhao, Tiancai Wang, and Xiaoyan Sun. LLaDA-VLA: Vision language diffusion action models. *arXiv preprint arXiv:2509.06932*, 2025c.

Zhenyu Wu, Yuheng Zhou, Xiuwei Xu, Ziwei Wang, and Haibin Yan. MoManipVLA: Transferring vision-language-action models for general mobile manipulation. *arXiv preprint arXiv:2503.13446*, 2025.

Runze Xu, Xiaolong Shan, Shuang Dai, Yu Wang, and Jincheng Yu. Knowing when to stop: Adaptive action chunking via internal cross-attention dynamics in VLAs. *arXiv preprint arXiv:2609.00908*, 2026. URL https://arxiv.org/abs/2609.00908v2.

Zhiyuan Xu, Kun Wu, Junjie Wen, Jinming Li, Ning Liu, Zhengping Che, and Jian Tang. A survey on robotics with foundation models: toward embodied AI. *arXiv preprint arXiv:2402.02385*, 2024. URL https://arxiv.org/abs/2402.02385.

Jianwei Yang, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, Yuquan Deng, Lars Liden, and Jianfeng Gao. Magma: A foundation model for multimodal ai agents. *arXiv preprint arXiv:2502.13130*, 2025.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, Zixiao Huang, Mingjie Wei, Yuqing Xie, Ke Yang, Bo Dai, Zhexuan Xu, Xiangyuan Wang, Xu Fu, Zhihao Liu, Kang Chen, Weilin Liu, Gang Liu, Boxun Li, Jianlei Yang, Zhi Yang, Guohao Dai, and Yu Wang. RLinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. *arXiv preprint arXiv:2509.15965*, 2025. URL https://arxiv.org/abs/2509.15965.

Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, Zhihao Liu, Kang Chen, Wenhao Tang, Quanlu Zhang, Weinan Zhang, Chao Yu, and Yu Wang. RLinf-VLA: A unified and efficient framework for VLA+RL training. *arXiv preprint arXiv:2510.06710*, 2025. URL https://arxiv.org/abs/2510.06710.

Andy Zhai, Brae Liu, Bruno Fang, Chalse Cai, Ellie Ma, Ethan Yin, Hao Wang, Hugo Zhou, James Wang, Lights Shi, Lucy Liang, Make Wang, Qian Wang, Roy Gan, Ryan Yu, Shalfun Li, Starrick Liu, Sylas Chen, Vincent Chen, and Zach Xu. Igniting vlms toward the embodied space. *arXiv preprint arXiv:2509.11766*, 2025.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. *arXiv preprint arXiv:2412.04987*, 2024.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Tsung-Yi Lin, Gordon Wetzstein, Ming-Yu Liu, and Donglai Xiang. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1702–1713, 2025. doi: 10.1109/CVPR52734.2025.00166.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025a. URL https://arxiv.org/abs/2510.10274.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*, 2025b.

Zhide Zhong, Haodong Yan, Junfeng Li, Xiangchen Liu, Xin Gong, Tianran Zhang, Wenxuan Song, Jiayi Chen, Xinhu Zheng, Hesheng Wang, and Haoang Li. FlowVLA: Visual chain of thought-based motion reasoning for vision-language-action models. *arXiv preprint arXiv:2508.18269*, 2025.

A Technical Appendices and Supplementary Material

A.1 Impact Statement

This paper presents work whose goal is to advance the field of machine learning. We do not identify any specific impacts of this work that require particular emphasis here.

B Inference Time Analysis

Table 6: Wall-clock inference time of each component on an NVIDIA GeForce RTX 4090 GPU with 2 camera images.

Component	Inference Time	Percentage
SFM	83.2 ± 1.4 ms	86.8%
Confidence rater	2.6 ± 0.1 ms	2.7%
AFM	10.1 ± 0.3 ms	10.5%
Overall	95.9 ± 1.6 ms	100.0%

The wall-clock inference time of each component is shown in Table 6, which is timed on an NVIDIA GeForce RTX 4090 GPU with 2 camera images and 10 FM steps. During inference, the majority of the computational cost is incurred by SFM. In SFM, the model computes a full forward pass over all VL and action tokens. This includes constructing the entire VL KV-cache from scratch. Consequently, SFM accounts for 86.8% of the total inference time. In contrast, AFM is substantially more efficient because it reuses the VL KV-cache produced in the very first pass. After SFM initializes the KV-cache, AFM does not recompute these caches in subsequent steps. Instead, it only updates the subset of action tokens whose confidence is below the threshold. The confidence rater introduces 2.7% overhead, since it is executed only once per action chunk, produces a lightweight scalar confidence value for each action token, and does not participate in iterative FM-style updates. Overall, the reuse of VL KV-cache and the partial-token update mechanism explain why AFM inference is significantly faster than SFM, while still enabling self-correction and higher success rates.

C Training and Inference Pseudo codes

Algorithm 1 Asynchronous Flow Matching Inference

Require: $\boldsymbol{o}_t,\ \ell,\ \hat{\boldsymbol{a}}^{\text{SFM}}_{t:t+L},\ \boldsymbol{m} \in \{0,1\}^L$; step size δ

For all l with $m_l = 0$: set $\hat{\boldsymbol{a}}_l^{\,1} \leftarrow \hat{\boldsymbol{a}}_l^{\text{SFM}}$;
For all l with $m_l = 1$: set $\hat{\boldsymbol{a}}_l^{\,1} \leftarrow \boldsymbol{n}_l \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
Set $\tau \leftarrow 1$
while $\tau > 0$ **do**
    Predict velocity: $\hat{\boldsymbol{v}}_{t:t+L} \leftarrow V_\theta\big(\boldsymbol{o}_t, \ell, \hat{\boldsymbol{a}}^{\tau}_{t:t+L}\big)$
    For all l with $m_l = 0$: $\hat{\boldsymbol{a}}_l^{\,\tau-\delta} \leftarrow \hat{\boldsymbol{a}}_l^{\,\tau}$
    For all l with $m_l = 1$: $\hat{\boldsymbol{a}}_l^{\,\tau-\delta} \leftarrow \hat{\boldsymbol{a}}_l^{\,\tau} - \delta\, \hat{\boldsymbol{v}}_l$
    $\tau \leftarrow \max(0,\ \tau - \delta)$
end while
return Final action $\tilde{\boldsymbol{a}}_{t:t+L} = \hat{\boldsymbol{a}}^{0}_{t:t+L}$

D AsyncVLA vs. Discrete Diffusion VLA

Table 7 summarizes the key differences between AsyncVLA and discrete diffusion VLA (such as LLaDA-VLA (Wen et al., 2025c), dVLA (Wen et al., 2025a), discrete-diffusion VLA (Liang et al., 2025), and UD-VLA (Chen et al., 2025a)) in four dimensions: action representation, prediction parameterization, remasking mechanism, and confidence estimation. Discrete diffusion VLA operates

Algorithm 2 Unified Training for AFM and SFM

Require: Dataset $\mathcal{D}$; FM model V_θ; batch size B

repeat
  Sample $\{(\boldsymbol{o}_t^{(i)}, \boldsymbol{a}_{t:t+L}^{(i)}, \ell^{(i)})\}_{i=1}^{B} \sim \mathcal{D}$
  for $i = 1, \dots, B$ **do**
    Sample $y^{(i)} \sim \mathcal{U}(0,1)$; $m_l^{(i)} \sim \text{Bernoulli}(y^{(i)})$
    Sample $\tau^{(i)} \sim \text{Beta}(1.5, 1)$; $\boldsymbol{n}^{(i)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
    Sample $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\mathbf{0}, \sigma_c^2 \boldsymbol{I})$
    $\boldsymbol{u}^{(i)} \leftarrow \boldsymbol{n}^{(i)} - \boldsymbol{a}^{(i)}$
    $\tilde{\boldsymbol{a}}^{(i)} \leftarrow \boldsymbol{a}_{t:t+L}^{(i)} + \boldsymbol{\epsilon}^{(i)} \odot (\mathbf{1} - \boldsymbol{m}^{(i)})$
    $\hat{\boldsymbol{a}}^{\tau^{(i)}} \leftarrow \tilde{\boldsymbol{a}}^{(i)} - \tau^{(i)} \big(\tilde{\boldsymbol{a}}^{(i)} - \boldsymbol{n}_{t:t+L}^{(i)}\big) \odot \boldsymbol{m}^{(i)}$
    $\hat{\boldsymbol{v}}^{(i)} \leftarrow V_\theta\big(\boldsymbol{o}_t^{(i)}, \ell^{(i)}, \hat{\boldsymbol{a}}^{\tau^{(i)}}\big)$
  end for
  $\mathcal{L} \leftarrow \frac{1}{B} \sum_{i=1}^{B} \big\| [\hat{\boldsymbol{v}}^{(i)} - \boldsymbol{u}^{(i)}] \odot \boldsymbol{m}^{(i)} \big\|^2$
  Update θ by back-propagation on $\mathcal{L}$
until converged

Table 7: Comparison between Discrete Diffusion VLA and AsyncVLA.

Aspect	**Discrete Diffusion VLA**	**AsyncVLA**
Action type	Discrete	Continuous
Prediction parameterization	Target (x-prediction)	Velocity (v-prediction)
ReMasking mechanism	Mask token	Gaussian noise
Confidence computation	Logits	Confidence rater

on *discrete* action tokens and typically adopts *target prediction* (i.e., x-prediction) under a *mask-token* corruption scheme, where confidence can be directly derived from the model's *logits*. In contrast, AsyncVLA models *continuous* actions and performs *velocity prediction* (v-prediction) under *Gaussian noise* masking; its confidence is computed by a separate *confidence rater* rather than raw logits. As demonstrated in our experiments, AsyncVLA achieves a significantly higher success rate compared to discrete diffusion VLA, further validating the superiority of continuous action representations and Gaussian noise masking in improving model performance and generalization ability.

Although this paper primarily uses v-prediction for its favorable optimization and sampling properties, the AsyncVLA formulation can be extended to alternative diffusion parameterizations with minimal changes. Concretely, the asynchronous masking-and-reconstruction pipeline, the noise-corruption process, and the confidence rater interface remain unchanged; one only needs to swap the denoiser head (and the corresponding training target) from predicting v to predicting either the clean target x (x-prediction) or the added noise ϵ (ϵ-prediction). Since v, x, and ϵ are deterministically related given the diffusion schedule, the same model architecture and sampling loop can be reused by replacing the prediction type and applying the standard conversion between parameterizations during training and inference.

D.1 Fine-Grained Parameter Exploration on LIBERO

To examine the robustness of AsyncVLA to the hyperparameters of the confidence-aware refinement mechanism, we perform a fine-grained parameter study on LIBERO. Starting from the default setting ($\alpha = 0.01, \beta = 0.98, T = 0.5$), we vary one parameter at a time while keeping the others fixed. Specifically, we test a larger positive-target coefficient $\alpha = 0.05$, a smaller negative-target coefficient $\beta = 0.95$, and two alternative refinement thresholds $T = 0.25$ and $T = 0.75$.

As shown in Table 9, varying one parameter at a time leads to only minor changes in performance. The average success rate remains within a narrow range of 97.2–97.4, while all four LIBERO suites stay highly stable. These results indicate that AsyncVLA is not sensitive to these hyperparameters. In particular, changing α and β only mildly affects the soft supervision targets used for training the

Table 8: Dataset Weights

Dataset	**Weight**
Bridge-V2	24.14%
RT-1	13.80%
TOTO	10.34%
VIOLA	10.34%
RoboTurk	10.34%
Jaco Play	10.34%
Berkeley Autolab UR5	10.34%
Berkeley Fanuc Manipulation	10.34%

Table 9: Fine-grained parameter exploration of AsyncVLA on LIBERO. We vary one parameter at a time from the default setting ($\alpha = 0.01, \beta = 0.98, T = 0.5$). Results are reported in success rate (%).

Model	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-Long	Avg.
AsyncVLA (Ours)	98.4	**99.2**	**98.6**	**93.4**	**97.4**
$\alpha = 0.05$	98.2	99.0	98.4	93.0	97.2
$\beta = 0.95$	98.4	**99.2**	98.2	93.2	97.3
$T = 0.25$	98.2	**99.2**	98.2	93.0	97.2
$T = 0.75$	**98.6**	99.0	98.4	93.0	97.3

confidence rater, whereas changing T only adjusts the threshold for selecting low-confidence tokens for refinement. Overall, the method exhibits robust behavior across a reasonable range of parameter choices.

E Implementation Details

E.1 Details of Training

In Table 8, we summarize the detailed datasets and their weights during the pre-training stage of AsyncVLA. We select part of the Open X-Embodiment dataset (O'Neill et al., 2023) as our pre-training robot demonstration data. We perform pre-training on 4 H200 GPU nodes (8 GPUs per node, 32 GPUs in total) under BF16 precision with gradient checkpointing enabled. We use flash-attention-2 and ZeRO-2 optimizer sharding for efficient training and proper GPU memory usage. The global batch size is set to 2048. The pre-training process takes roughly 2.5 days. We use the cosine decay learning rate scheduler with the largest learning rate set as 1×10^{-4}. Further fine-tuning on LIBERO, Bridge-V2, and Fractal is performed on a single H200 node with 8 GPUs, requiring 15–32 hours depending on the dataset size. The chat template includes the prompt "You are a helpful physical assistant." at the beginning of each sample. Throughout all stages, we use AdamW optimizer with weight decay set to 0, β_1 = 0.9, and β_2 = 0.999. In the mask generation, we set the confidence threshold as $T = 0.5$. When training the confidence rater, we set $\alpha = 0.01$, $\beta = 0.98$, and $\epsilon = 1 \times 10^{-6}$.

E.2 Details of Flow Matching

For both SFM and AFM, we use a uniform schedule of 10 discretization steps from noise at $\tau = 1$ towards the target action at $\tau = 0$. The SFM-only 10/20-step comparison is reported in Table 4. A separate 5/10/20-step sensitivity study of targeted AFM has not been completed; the default schedule should therefore not be interpreted as an established optimum for AFM.

E.3 Structure of Confidence Rater

The backbone of the confidence rater is a 4-layer transformer with a linear layer rate head, which sums up to 308M parameters and takes up 7.56% of the total 4.08B parameters of the overall VLA model. Each of the four transformer blocks contains multi-head self-attention with 32 attention heads and a feed-forward network width of 6144.

F Additional Analyses of Asynchronous Correction

F.1 Evaluation Units and Statistical Scope

The additional WidowX analyses use four tasks with 24 evaluation configurations per task, giving 96 episodes per method. Unless otherwise specified, comparisons use the same Qwen2.5-VL-3B backbone, Bridge-V2 data, one policy-training seed, and 10 integration steps in each of SFM and AFM. The default confidence threshold is $T = 0.5$. Each compared method executes its own closed-loop rollout: paired initial configurations and random streams do not imply identical subsequent trajectories. These experiments characterize the evaluated checkpoints, not variation across independently trained policies.

Aggregation and uncertainty. Success rates are percentages and gains are percentage points. Exact counts are shown where available. The descriptive 20-step SFM mean of 51.1% and alternating-training mean of 68.7% retain the convention of averaging the displayed, rounded task-wise percentages; they are not exact pooled success counts. For example, the four displayed alternating rates average to 68.725%. This rounding convention can differ from pooling unrounded episode outcomes by 0.1 percentage point. Paired effect sizes and tests below instead use the explicit paired counts.

We compute two-sided exact McNemar tests from discordant outcomes. For the three paired comparisons below, the reported bootstrap intervals can be reproduced from the empirical paired differences $z_i \in \{-1, 0, 1\}$, with $+1$ indicating a success unique to AsyncVLA. For each comparison, we reset `random.Random(42)` and use its `choices` method to draw from $(-1, 0, 1)$ with weights equal to the observed outcome counts. We draw 10,000 samples of N differences, compute $100N^{-1}\sum_i z_i$ for each sample, and take the linearly interpolated 2.5th and 97.5th percentiles. These are episode-level intervals conditional on the evaluated configurations and checkpoints. The five-noise-seed comparison in Appendix F.3 instead uses the noise seed as its statistical unit.

F.2 Paired AFM Effect and Selection Controls

Confidence-guided AFM recovers more failures than it introduces on the paired WidowX evaluation. SFM and AsyncVLA share initialization, environment seeds, and a deterministic SFM-noise stream with fresh noise for each chunk; AFM uses a separate stream. Task-wise SFM/AsyncVLA success counts are $14/17$, $13/16$, $8/14$, and $11/21$, each out of 24, for spoon, carrot, stack, and eggplant, respectively. The totals are $46/96$ and $68/96$.

Table 10: Paired WidowX outcomes for SFM-only inference and AsyncVLA on 96 configurations. Each method runs a separate closed-loop rollout per configuration.

Outcome	Episodes
Both succeed	43
Only SFM succeeds	3
Only AsyncVLA succeeds	25
Both fail	25

The 25 recoveries and 3 regressions in Table 10 yield a gain of 22.9 points, McNemar $p = 2.7 \times 10^{-5}$, and a paired-bootstrap 95% interval of $[+13.5, +32.3]$ points. The regressions also show that selective correction is not uniformly beneficial on every episode.

The three mask-selection controls in Table 11 obtain $60/96$, $59/96$, and $68/96$ successes, respectively. The independent random control masks each token with probability 0.5, as in the original no-rater

Table 11: Inference and selection controls on one fixed WidowX action-noise realization. SFM-only rows have no second stage. Success rates retain the reporting precision described in Appendix F.1.

Inference procedure	Second-stage mask	Success (%)
SFM, 10 steps	–	47.9
SFM, 20 steps	–	51.1
Full-remasking AFM	100.0%	60.4
Independent random mask	50.0%	62.5
Cardinality-matched random mask	35.1%	61.5
Confidence-selected mask	35.1%	70.8

ablation. For the cardinality-matched control, each chunk computes its own confidence mask on its own trajectory, records its cardinality K, discards the selected positions, and samples exactly K positions without replacement. Separate rollouts share initial configurations, environment seeds, SFM-noise streams, and per-token AFM-noise streams; mask positions use an independent random seed. Thus, cardinality is matched to each random-control chunk's own confidence prediction, not to a mask transplanted from an AsyncVLA trajectory. Confidence selection improves over cardinality-matched random selection by 9.4 points, with 11 recoveries and 2 regressions, McNemar $p = 0.0225$, and a paired-bootstrap 95% interval of $[+2.1, +16.7]$ points. This result isolates selection at the reported noise realization; it is not a training-seed stability result.

Interpretation of the original refinement ablations. The Delta and Direct refinement controls in Table 4 share the confidence mask, SFM context, minibatches, optimizer, and 10,000 update budget. Each uses an approximately 308M-parameter transformer with four layers, 32 attention heads, and feed-forward width 6144. Their outputs are either $a^{\mathrm{SFM}} + \Delta a$ or directly predicted corrected actions. These comparisons test the correction mechanism with selection held fixed; the cardinality-matched comparison above tests selection itself. The 7.3% no-unified-training result uses SFM-only training followed by SFM-rater-AFM inference, whereas the 47.9% no-AFM result uses unified training followed by SFM-only inference. The former therefore probes training–inference alignment, not the quality of SFM-only inference.

F.3 Full Remasking across Action-Noise Seeds

Full remasking is a 10-step SFM pass followed by a fresh-noise, 10-step AFM pass. It reuses only the visual-language KV-cache and preserves no first-stage action values. To distinguish the single-realization result from a generic second-pass benefit, we repeat SFM and full remasking over five action-noise seeds on the same 96 configurations and the same trained checkpoint.

Table 12: Full-remasking control across five action-noise seeds. Entries are successes out of 96 fixed episodes. The last row gives mean success (%) and sample standard deviation in percentage points across noise seeds, not training seeds.

Noise seed	SFM	Full-remasking AFM
1	$46/96$	$58/96$
2	$49/96$	$49/96$
3	$47/96$	$50/96$
4	$48/96$	$48/96$
5	$50/96$	$51/96$
Mean $\pm$ sample SD	50.0 ± 1.6	53.3 ± 4.1

The mean paired difference is $+3.3$ points, with a Student-t 95% interval of $[-3.2, +9.9]$ points and a two-sided paired-test $p = 0.231$ (four degrees of freedom). The apparent advantage of an unconditional second pass is therefore unresolved in this five-seed control. It should not be conflated with evidence for confidence-guided contextual correction.

F.4 Refinement Frequency and Error Alignment

For valid, non-padding action positions $\mathcal{V}$ in a chunk, define

$$r = \frac{1}{L_v} \sum_{l \in \mathcal{V}} \mathbb{1}\{p_l < T\}, \qquad L_v = |\mathcal{V}|. \tag{8}$$

Statistics include chunks with no selected token. At $T = 0.5$, AFM selects about one third of the valid tokens in both the WidowX evaluation and the PiPER validation analysis.

Table 13: Mean fraction of valid action tokens selected for AFM at $T = 0.5$. WidowX uses evaluation chunks; PiPER uses 40 validation trajectories and is a separate analysis.

Domain	Task	Selected (%)
WidowX	Put spoon on towel	34.2
	Put carrot on plate	33.5
	Stack cubes	41.6
	Put eggplant in basket	31.1
	Overall	35.1
PiPER	Carrot-to-bowl	29.8
	Pen extraction	37.6
	Pouring water	43.1
	Spoon-to-plate	28.4
	Overall	34.7

At least one token is selected in 78.3% of WidowX chunks and 76.9% of PiPER chunks. Only 4.6% and 3.9% of chunks, respectively, select more than 75% of their valid tokens. Stacking and pouring have the largest selection rates in their respective domains; a larger rate alone does not establish better error discrimination.

Diagnostic event and data boundary. We evaluate error alignment on 40 PiPER validation trajectories, 10 per task. These trajectories were used for checkpoint selection but not gradient updates. For each valid action token, error is the MSE over valid normalized action dimensions; padding is excluded. Across all N valid tokens in this split, we use a linearly interpolated 70th percentile and select exactly $\lceil 0.30N \rceil$ high-error tokens, resolving boundary ties by episode and then token index. This is a global evaluation event, distinct from the within-chunk relative pseudo-labels in Equation (7). Metrics pool valid tokens and are post-selection descriptive diagnostics, not estimates from an untouched test set.

Table 14: PiPER error-alignment diagnostics. Low confidence $p_l < T$ predicts the global high-error event defined above. Ranking and discrimination metrics do not establish probabilistic calibration.

Metric	Result
Spearman correlation: confidence vs. SFM MSE	-0.71
AUROC	0.86
AUPRC	0.76
Precision	70.6%
Recall	81.5%
F1	75.7%

Selected tokens have $2.9\times$ the pre-refinement MSE of unselected tokens. AFM reduces the selected tokens' MSE by 36.9%, while unselected action values remain fixed by construction. These observations support demonstration-relative error alignment, not the calibrated task-success estimation or failure detection studied by VLAConf and SAFECAST (Huang et al., 2026; Rajaprakash et al., 2026). They do not make p_l a calibrated success probability or establish reliability when an entire action chunk is poor.

F.5 WIDOWX THRESHOLD–REFINEMENT TRADE-OFF

We vary only the inference threshold, keeping model parameters, matched initial configurations and environment seeds, and the 10-step SFM/AFM schedules fixed. This experiment is on WidowX and is distinct from the LIBERO parameter study in Appendix D.1.

Table 15: WidowX inference-threshold sweep on one fixed noise realization. The $T = 0$ endpoint is SFM-only inference; $T = 1$ performs fresh-noise full remasking and preserves no first-stage action values.

Threshold	Selected tokens (%)	Success (%)
$T = 0.00$	0.0	47.9
$T = 0.25$	17.9	66.7
$T = 0.50$	35.1	70.8
$T = 0.75$	62.4	68.8
$T = 1.00$	100.0	60.4

Success is non-monotonic in the refinement rate and is highest at the default threshold in this sweep. The full-remasking endpoint must be interpreted alongside the unresolved five-noise-seed result in Appendix F.3, rather than as evidence that any additional pass reliably helps.

F.6 CONTEXT-CORRUPTION ABLATION

Imperfect training context closes most of the gap between clean ground-truth context and inference-time SFM predictions. The variants in Table 16 share the backbone, Bridge-V2 data, initialization, mask sampling, training budget, 10-step SFM/AFM schedules, $T = 0.5$, and the 96-episode evaluation. Only unmasked training context changes.

Table 16: Unmasked-context ablation on WidowX, using one policy-training seed. Columns denote the spoon, carrot, stack, and eggplant tasks; success rates are percentages.

Training context	Spoon	Carrot	Stack	Eggplant	Avg.
Clean ground truth	66.7	62.5	50.0	79.2	64.6
Gaussian, $\sigma_c = 0.01$	66.7	62.5	54.2	83.3	66.7
Gaussian, $\sigma_c = 0.03$	70.8	66.7	54.2	83.3	68.8
Gaussian, $\sigma_c = 0.05$	70.8	66.7	58.3	87.5	70.8
Gaussian, $\sigma_c = 0.10$	66.7	62.5	54.2	83.3	66.7
Gaussian, $\sigma_c = 0.20$	62.5	58.3	50.0	79.2	62.5
Frozen-SFM predictions	70.8	66.7	58.3	83.3	69.8

For actual-SFM context, the final frozen unified policy runs in SFM mode once per augmented training-sample presentation. Predictions are regenerated after augmentation without a persistent prediction cache, using the seed key `hash(42, trajectory_id, timestep, augmentation_view_id)`. No validation or evaluation trajectory supplies this context. This construction preserves presentation-level stochastic diversity while changing the context-generation mechanism.

Clean context yields 64.6%, moderate Gaussian corruption with $\sigma_c = 0.05$ yields 70.8%, and frozen-SFM context yields 69.8%. Excessive corruption degrades performance. The difference between Gaussian and frozen-SFM context is one episode out of 96. Gaussian corruption is therefore an efficient approximation to imperfect context, not an exact model of structured SFM errors or a demonstrated superior choice.

F.7 CONFIDENCE-RATER TRAINING STRATEGIES

Post-hoc training provides a stationary final-policy distribution for the rater, while policy updates change both SFM predictions and hidden representations. We compare post-hoc, joint, and alternating

schedules independently in the WidowX and PiPER domains. Results are not transferred between domains. In each domain, strategies start from the same policy checkpoint and process 10,000 additional minibatches. The PiPER starting checkpoint is selected by average FM validation loss after the 30,000-step policy-training stage.

All schedules use rater-loss weight 0.1; the joint objective is $\mathcal{L}_{\mathrm{FM}} + 0.1\mathcal{L}_{\mathrm{rate}}$. Joint pseudo-labels are regenerated online every minibatch. SFM prediction and label construction are stop-gradient: rater loss updates only the rater, and FM loss updates the policy. Post-hoc training gives 10,000 rater updates with a frozen policy; joint training gives 10,000 updates to each component; and the 10,000-iteration alternating schedule gives 5,000 updates to each component. The sampled minibatch budget is matched, but parameter-group updates and FLOPs are not.

Table 17: Rater-training schedules evaluated on 96 WidowX episodes. Success rates are percentages. The alternating average follows the rounded-task convention in Appendix F.1.

Schedule	Spoon	Carrot	Stack	Eggplant	Avg.
Joint	62.5	58.3	50.0	83.3	63.5
Alternating	70.8	62.5	58.3	83.3	68.7
Post-hoc, frozen policy	70.8	66.7	58.3	87.5	70.8

Table 18: Separately executed rater-training diagnostics on the 40 PiPER validation trajectories, using the post-selection event in Appendix F.4.

Schedule	AUROC	AUPRC
Joint	0.78	0.64
Alternating	0.85	0.74
Post-hoc, frozen policy	0.86	0.76

Neither tested alternative exceeds post-hoc training under these equal-minibatch, unequal-update schedules. However, alternating training receives only half as many rater updates. An update-matched alternating run would require 20,000 global iterations to give 10,000 updates per component; that control has not been completed. The reported outcomes therefore do not establish strategy-level superiority under matched optimizer updates or FLOPs, nor do the diagnostic AUROC/AUPRC values establish calibration.

F.8 Rater Inputs and Backbone Compatibility

Both visual-language context and initially generated actions contribute to the tested rater. The input ablation in Table 19 uses the same 96-episode WidowX evaluation.

Table 19: Rater-input ablation on WidowX. The combined input uses final-layer visual-language states together with the SFM action chunk.

Rater input	Success (%)
Visual-language tokens only	60.4
SFM action tokens only	64.6
Visual-language and SFM action tokens	70.8

We also repeat the SFM/AFM contrast with Qwen3-VL-4B, using the same Bridge-V2 data, an adapted head/rater recipe, and the same 96-episode protocol.

Qwen3-VL-4B improves both point estimates while retaining a positive AFM contrast. This limited comparison does not establish backbone-independent or multi-seed generality.

Table 20: Backbone-controlled WidowX comparison. Rates are percentages and gains are percentage points. Each setting uses one policy-training seed, so this is compatibility evidence rather than training-seed stability evidence.

VLM backbone	SFM only	AsyncVLA	Gain
Qwen2.5-VL-3B	47.9	70.8	+22.9
Qwen3-VL-4B	52.1	72.9	+20.8

G Asynchronous Correction under Distribution Shift

G.1 Official Visual Matching and Variant Aggregation

We disable AFM in the same trained Google Robot model while retaining the SFM schedule, paired seeds and initializations, and the official SimplerEnv protocol. Thus, the comparison in Table 21 changes confidence-guided correction rather than the trained checkpoint. M denotes Visual Matching and A denotes Variant Aggregation.

Table 21: Google Robot with/without-AFM ablation; $+\mathrm{AFM}$ denotes the full AsyncVLA pipeline. Success rates and gains are in percent and percentage points, respectively. Average gains use task-level values before rounding; thus the A gain is 12.8, not the difference of the displayed rounded averages.

	Visual Matching (M)			Variant Aggregation (A)		
Task	SFM	$+\mathrm{AFM}$	Gain	SFM	$+\mathrm{AFM}$	Gain
Pick Coke	92.0	98.0	+6.0	82.4	91.6	+9.2
Move Near	73.8	82.3	+8.5	69.2	81.7	+12.5
O/C Drawer	56.6	70.5	+13.9	41.5	58.0	+16.5
Put in Drawer	31.6	50.4	+18.8	13.2	26.0	+12.8
Average	63.5	75.3	+11.8	51.6	64.3	+12.8

The average gain is 11.8 points under M and 12.8 under A. The average fraction of selected tokens rises from 31.6% to 40.8% under the visual variants, but most initial action values are still retained.

Table 22: Mean selected-token fractions (%) in the Google Robot evaluation, at $T = 0.5$.

Task	Visual Matching	Variant Aggregation
Pick Coke	23.4	30.7
Move Near	30.9	37.5
Open/Close Drawer	34.8	44.2
Put in Drawer	37.3	50.8
Average	31.6	40.8

G.2 Controlled SimplerEnv Shifts

To separate shift factors more directly, we evaluate the same Bridge-V2-trained policy with and without AFM under controlled perturbations, without OOD fine-tuning. The perturbation definitions are:

- Camera azimuth in $\{-15, 0, +15\}^\circ$, lateral offset in $\{-5, +5\}$ cm, and height offset in $\{0, +5\}$ cm.
- Lighting-intensity multipliers in $\{0.65, 1, 1.35\}$ and color temperatures in $\{3500, 6500\}$ K.
- Object offsets $\Delta x, \Delta y \sim \mathcal{U}(-4, 4)$ cm and yaw $\mathcal{U}(-30, 30)^\circ$.

- Two distractors placed at least 8 cm from the target and 10 cm from the gripper, with at most 50% target occlusion.

The combined condition applies these perturbations together. Each reported shifted condition uses six configuration templates, four tasks, and four rollout seeds per task-template pair: $6 \times 4 \times 4 = 96$ paired episodes, or 24 per task. The four shifted rows therefore total 384 trials per method; the templates do not constitute an exhaustive Cartesian product of the perturbation ranges. For task index t and episode index e, the environment, SFM-noise, AFM-noise, and random-mask seeds are $1000 + 100t + e$, $2000 + 100t + e$, $3000 + 100t + e$, and $4000 + 100t + e$, respectively.

Table 23: Controlled-shift evaluation of a fixed WidowX checkpoint. Rates are percentages. Gains here are differences of the displayed rounded success rates; the rounding convention can differ by 0.1 point from an exact count-based difference. Each shifted condition contains 96 paired episodes.

Setting	SFM only	AsyncVLA	Gain	Selected tokens
Standard WidowX	47.9	70.8	+22.9	35.1
Changed camera	38.5	59.4	+20.9	41.8
Changed lighting	42.7	64.6	+21.9	39.6
Changed object layout	35.4	56.3	+20.9	43.7
Combined shift	27.1	46.9	+19.8	49.8

For example, the changed-camera rates correspond to 37 and 57 successes out of 96. Both methods degrade as shifts become stronger. Under the combined shift, 20-step SFM reaches 30.2%, compared with 27.1% for 10-step SFM and 46.9% for AsyncVLA. The additional synchronous denoising steps do not reproduce the observed correction benefit in this setting. Nevertheless, AFM does not solve visual distribution shift, and a larger refinement ratio does not itself establish OOD error alignment. These results concern one policy-training seed; broader training-seed and real-world OOD generality remain untested.

H Real-World Collection, Evaluation, and Paired Analysis

H.1 Demonstrations and Data Split

Two operators collect 100 successful demonstrations for each of the four PiPER tasks, giving 400 trajectories. Teleoperation uses 10-Hz SpaceMouse control with six-degree-of-freedom delta end-effector commands and a binary gripper action. Front-view and wrist RGB cameras run at 30 Hz; nearest frames are synchronized to control observations. A demonstration is retained only if success persists for two seconds and there is no stop, intervention, unintended collision, workspace exit, or sensor dropout longer than 200 ms. Pause intervals are masked out of the loss. A task/operator-stratified split with seed 42 gives 360 training and 40 validation trajectories, with 10 validation trajectories per task. Each method trains a single four-task policy using one training seed.

H.2 Method-Specific Adaptation and Shared Budget

OpenVLA-OFT starts from the official OpenVLA-7B checkpoint, with rank-32 LoRA and a trained action head and projector. The π_0 and $\pi_{0.5}$ baselines fully fine-tune their official base checkpoints; $\pi_{0.5}$ receives no additional semantic loss in this experiment. AsyncVLA starts from its final Open-X-pretrained Qwen2.5-VL-3B checkpoint, fully fine-tunes the VLM and shared FM module, then freezes the policy and trains a randomly initialized 308M-parameter rater.

All methods share the same inputs, 360/40 split, seven-dimensional action interface, and 30,000 policy-training steps, with no extra PiPER data for any method. They use AdamW, BF16, batch size 128, 5% warm-up, cosine decay, and matched augmentation. Learning rates are 2×10^{-4} for OpenVLA-OFT and 5×10^{-5} for the π baselines. AsyncVLA uses 10^{-4} for its language-FM components and 2×10^{-5} for its vision encoder. Held-out action loss selects checkpoints; test success is never used for selection. AsyncVLA additionally receives 10,000 rater-only updates. The policy-step and data budgets are matched, but the backbone sizes, adaptation mechanisms, and FLOPs are not.

H.3 Control, Success Criteria, and Counterbalancing

All policies operate at 10 Hz, predict 16-action chunks, and execute eight actions before replanning, corresponding to a 0.8-second replanning interval. Episodes last at most 60 seconds and require success to persist for two seconds. Initial object and target positions vary by ± 4 cm per axis, yaw by $\pm 15^\circ$, and joint angles by ± 0.03 rad. For carrot and spoon placement, the object must be fully supported by the target bowl or plate. For pen extraction, the pen must remain outside its holder. Pouring requires transfer of at least 70% of the initial water, with less than 10% spilled by weight. Timeouts, drops, safety stops, and interventions count as failures.

Methods share 50 initial configurations per task, keyed by task and configuration. Execution order is cyclically counterbalanced using rotations of [OpenVLA-OFT, π_0, $\pi_{0.5}$, AsyncVLA]. Reset tolerances are 10 mm for objects, 5 mm for receptacles, 5° for yaw, 0.5° for joints, and 60 ± 2 mL for initial water volume. Invalid resets are repeated before execution rather than counted as policy trials. Anonymized-video scoring is blind to method identity, and ambiguous trials receive a second review.

H.4 Paired Outcomes and Their Scope

The task-wise results in Table 5 correspond to totals of $131/200$, $130/200$, $154/200$, and $174/200$ successes for OpenVLA-OFT, π_0, $\pi_{0.5}$, and AsyncVLA, respectively. The configuration-paired $\pi_{0.5}$/AsyncVLA outcomes provide information beyond those marginal success rates.

Table 24: Paired real-world outcomes for $\pi_{0.5}$ and AsyncVLA on the same 200 task configurations. This is configuration-level evidence from one trained policy per method.

Outcome	Episodes
Both succeed	147
Only AsyncVLA succeeds	27
Only $\pi_{0.5}$ succeeds	7
Both fail	19

The paired gain is 10.0 percentage points, with McNemar $p = 8.21 \times 10^{-4}$ and a paired-bootstrap 95% interval of $[+4.5, +15.5]$ points, using the calculation in Appendix F.1. The experiment covers one robot platform and four tasks. It does not measure variability across training seeds or establish broad real-world OOD robustness.

Measured correction cost. The timings in Table 6 cover the complete 10-step schedule per stage, with two camera images on an RTX 4090. The rater and AFM together add 12.7 ms, or 15.3% relative to the 83.2 ms SFM baseline. The measured 95.9 ms total is below the 0.8-second replanning interval in this protocol. The rater accounts for 7.56% of the system's parameters and 2.7% of total latency. These are device- and protocol-specific timings, not a guarantee for higher-frequency reactive control. The fraction of selected action values is also not, by itself, a measure of the fraction of backbone FLOPs executed. RTC's execution-overlap inpainting (Black et al., 2025c) and FlashVLA's cross-chunk streaming denoising (Li et al., 2026) address scheduling choices beyond the fixed execution protocol here; integrations with our within-chunk correction have not been evaluated.

Remaining controls. The additional analyses do not include a completed 20,000-iteration update-matched alternating run or a targeted-AFM 5/10/20-step sweep. Five repeated action-noise realizations are not five independently trained models. The PiPER rater diagnostics reuse a checkpoint-selection split, relative confidence may preserve globally inaccurate chunks, and stronger shifts still reduce absolute success. These boundaries limit claims about training-strategy superiority, probabilistic calibration, and general robustness.