REACTIVE REAL-TIME FLOW POLICIES VIA ASYNCHRONOUS DISTRIBUTION ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Generalist robot policies such as vision-language-action models (VLAs) have achieved remarkable generalization, but their inference delays can conflict with the demands of real-time control. Asynchronous execution avoids pauses between action chunks by predicting the next sequence of actions while the robot carries out the previous one. In this paper, we study whether asynchronous execution produces the same action distribution as the original VLA. We find that, for non-Markovian demonstrations, asynchronous execution can produce a fundamentally different action distribution, which can limit the policy's reactivity. In our method, we seek to restore this reactivity by aligning the asynchronously produced action distribution with that of the original VLA through two complementary mechanisms. First, *Recursive Flow-Field Distillation* trains the asynchronous policy using the VLA's action-generation flow. We characterize the learned distribution theoretically and show experimentally that our asynchronous policy can generate nearly the full range of actions the original VLA would produce, while existing asynchronous methods recover only a fraction of that range. Second, *Propose–Resolve* prepares multiple action sequences asynchronously and uses the latest observation to select among them based on a lightweight approximation of their likelihood under the VLA's action distribution. Our resulting method matches the original VLA's success on LIBERO and retains about 80% of its success on RoboMimic, about 30 percentage points more than existing asynchronous methods.

1 INTRODUCTION

Generalist policies, such as Vision-language-action (VLA) models, are becoming better at learning general-purpose behavior from large, diverse datasets (Kim et al., 2024; Black et al., 2024; Shukor et al., 2025). Yet the inference times of these large models limit real-time control: robots may need to dispatch actions at a much higher rate than the policy can produce them. *Action chunking* partly mitigates this mismatch by predicting multiple future actions at once, at the cost of reactivity (Zhao et al., 2023; Chi et al., 2023). Under *synchronous execution*, however, the robot must still wait between chunks while the next prediction is generated. While such pauses may be acceptable for some tasks, they limit throughput and become inherently detrimental in continuous or dynamic settings (Black et al., 2025a).

Asynchronous execution addresses this limitation by starting computation of the next action chunk before execution of the current chunk has finished, such that inference overlaps with ongoing control (Black et al., 2025a;b; Ho et al., 2026). Real-Time Chunking (RTC) enables this by matching the overlapping portions of consecutive chunks (Black et al., 2025a). Its central insight is that asynchronous execution remains reliable only when the new prediction is compatible with the actions already committed by the preceding chunk. This original approach to continuous execution, however, introduces a new mismatch: because the next chunk is generated before the current one has finished executing, it is conditioned on an earlier state. During inference, the robot continues moving, so that at *handoff*, when execution passes to the newly generated chunk, the robot will already be in a different state from the one on which that chunk was conditioned. This can limit reactivity and reduce performance (Black et al., 2025a; Park & Tulsiani, 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

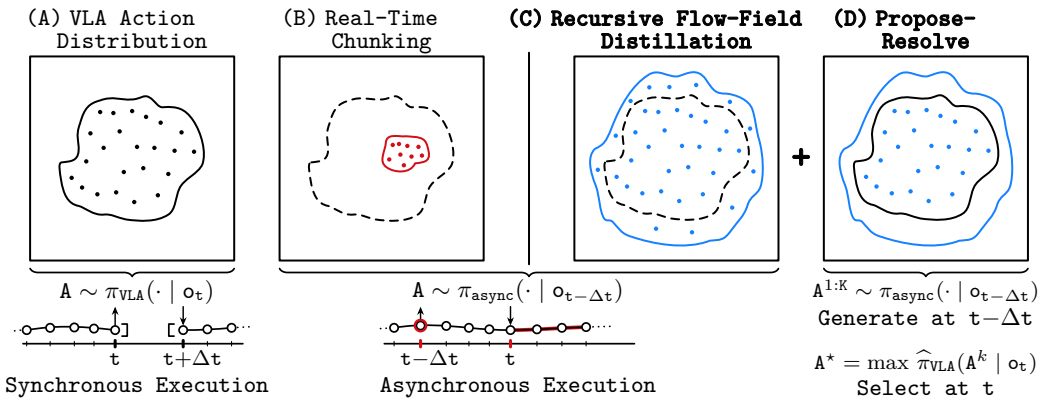


Figure 1: **Asynchronous execution and distribution alignment.** The point clouds schematically represent different action distributions at t, with each point corresponding to a possible action sample; inference of such a sample takes Δt. *(A) Synchronous execution* queries the VLA at time t, so execution has to wait until the action is generated at $t + \Delta t$. *(B–C) Asynchronous execution* instead starts generation at $t - \Delta t$, while the robot carries out its committed actions, so that the next chunk is prepared for execution at t. *(B) Real-Time Chunking* produces a narrower subset of the actions compared to the VLA when the training data is non-Markovian. *(C) Recursive Flow-Field Distillation* trains the asynchronous policy using the VLA's action-generation flow, recovering a broader proposal distribution. *(D) Propose–Resolve* uses the latest observation to estimate each candidate's likelihood under the corresponding VLA distribution and selects the highest-scoring continuation.

To investigate this mismatch, we study the relationship between two possible action chunks at the same handoff state: one produced asynchronously by RTC, and the other by querying the VLA synchronously. We find that, even when both policies are learned from the exact same demonstrations, non-Markovian data can cause asynchronous execution methods to produce a fundamentally different action distribution from the VLA. Figure 1(A–B) exemplifies how RTC, while removing the inference delay, might produce only a narrow subset of the actions that the VLA would generate.

In our method, we seek to recover the original VLA's action distribution asynchronously to restore the reactivity that current asynchronous execution methods lose. To avoid inheriting the history dependence of non-Markovian demonstrations, we first change what the asynchronous policy is trained to predict. Instead of supervising the asynchronous policy with the continuation recorded in the demonstration, we train it to reproduce the VLA's behavior after the already-committed actions have been executed. We realize this objective through *Recursive Flow-Field Distillation* (RFD), transferring the VLA's action-generation flow into the asynchronous policy. When the future observation is uncertain, this learning target marginalizes over the VLA action distributions associated with the possible observations that could be reached. As illustrated in Figure 1(C), this can produce a broader proposal distribution than the VLA distribution associated with the observation ultimately reached. The broader proposal distribution must therefore be aligned with the distribution the VLA would produce once the actual observation becomes available. To this end, *Propose–Resolve* proposes multiple candidate actions asynchronously and resolves among them once the observation at handoff arrives. A lightweight resolver fits a diagonal-Gaussian approximation to the VLA distribution conditioned on the reached observation, uses it to estimate each candidate's likelihood, and selects the highest-scoring continuation. This exploits a computational asymmetry: generating a new action sequence requires expensive iterative flow denoising, whereas evaluating a likelihood of an already-generated candidate can be a much cheaper operation. Figure 1(D) illustrates how this second step filters the broader proposal set toward the VLA distribution (A), aligning the asynchronously generated behavior with what the VLA would produce from the reached observation.

In summary, our method moves the expensive generative computation into the past, while still executing a reactive action that the VLA itself could produce from the observation available in the present. Our experiments show that the new learning objective recovers nearly the full range of actions the original VLA would produce, while existing asynchronous methods recover only a fraction of that range. Together with Propose–Resolve, we evaluate this approach in closed-loop execu-

tion: our complete method matches the original VLA's success on LIBERO (Liu et al., 2023) and retains about 80% of its success on RoboMimic (Mandlekar et al., 2022), approximately 30 percentage points more than existing asynchronous methods. Interestingly, our ablations show that closer distributional alignment of RFD is substantially more important on RoboMimic than on LIBERO, suggesting that the value of reactivity versus continuation coherence is task dependent.

2 Preliminaries and Problem Setting

Synchronous execution. We first consider the standard execution of an action-chunking VLA. Given an observation o_t, the policy generates an H-step action chunk $B_t \sim \pi_{\text{VLA}}(\cdot \mid o_t)$. Only the first s actions, $A_t := B_t[0:s]$, are executed before the policy is queried again. By slight abuse of notation, we also write $A_t \sim \pi_{\text{VLA}}(\cdot \mid o_t)$ for the induced marginal over this executed segment. Under synchronous execution, the robot must wait at time t for the VLA to finish inference of A_t.

Asynchronous execution. To make the next action chunk available without waiting at time t, asynchronous execution starts its inference d control steps earlier, at $t-d$, while the robot continues executing the current plan. For example, 150 ms of inference at 20 Hz (i.e. actions required every 50ms) corresponds to $d = 3$ control steps of advance preparation. The actions that execute during this inference window are already committed; we collect them in the prefix $P_t := (a_{t-d}, \ldots, a_{t-1})$. Asynchronous methods use this known prefix to condition or constrain generation of a new H-step chunk. Let $\widetilde{B}_{t-d} \sim \pi_{\text{async}}(\cdot \mid o_{t-d}, P_t)$ denote the chunk produced by this asynchronous generation procedure. The next s actions used after handoff are $U_t := \widetilde{B}_{t-d}[d : d+s]$, which we call the executed *continuation*. The resulting execution plan can therefore be written as $[P_t; U_t; \ldots]$: P_t is fixed by the preceding plan, while U_t is newly generated for execution after handoff. Again, by slight abuse of notation, we write $U_t \sim \pi_{\text{async}}(\cdot \mid o_{t-d}, P_t)$ for the continuation distribution induced by the complete asynchronous generation procedure.

For ease of notation, we henceforth consider a single handoff at time t and omit its subscript, writing $A := A_t$, $U := U_t$, and $P := P_t$. We denote the observation available when generation begins by $o^- := o_{t-d}$, and the observation available at handoff by $o^+ := o_t$. Figure 2 (left) summarizes the asynchronous generation timeline and the corresponding variables introduced above. We thus obtain two action distributions for the same s-step interval beginning at o^+:

$$A \sim \pi_{\text{VLA}}(\cdot \mid o^+), \qquad U \sim \pi_{\text{async}}(\cdot \mid o^-, P). \tag{1}$$

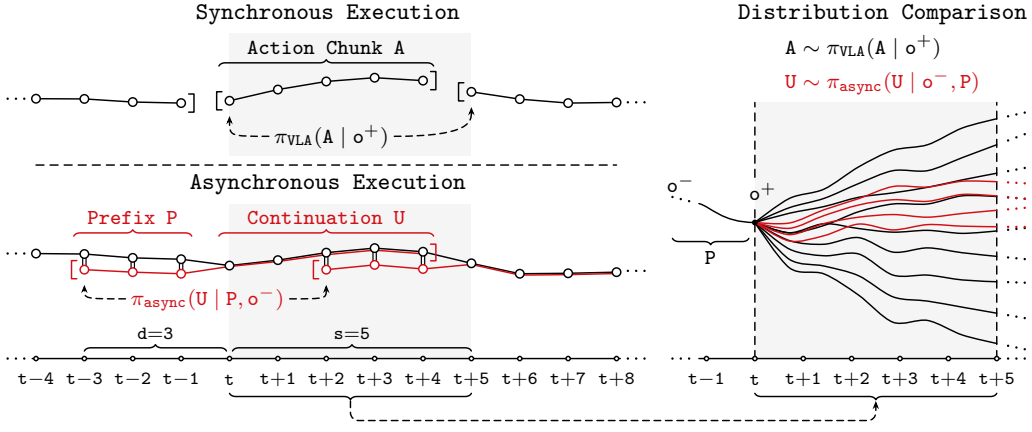


Figure 2: **Execution timing and action distributions at handoff.** Schematic with $d = 3$ and $s = 5$. **Left:** *Synchronous execution* queries the VLA at time t for the five actions executed over $[t, t+s]$. Asynchronous execution instead begins generation at $t-d$: the first d actions form the committed prefix P, while the following s actions form the continuation U executed after handoff. Note that the schematic assumes prefix-compatible generation, so U begins from the state reached after executing P; enforcing this compatibility is the original objective from RTC. **Right:** The resulting distributions over the same s-step execution interval. Black curves show possible VLA action chunks conditioned on o^+, while red curves show asynchronous continuations generated from (o^-, P).

Figure 2 (right) illustrates these two distributions over the same execution interval: the VLA distribution conditioned on o^+, and the asynchronous distribution generated based on (o^-, P). At time t, the asynchronous continuation is already available, whereas obtaining a new VLA sample would require another inference. This motivates the central question of our analysis: *how does the action distribution produced by asynchronous execution compare with that of the original VLA?*

3 Distributional Analysis

Train-Time Real-Time Chunking (TT-RTC) explicitly learns the asynchronous continuation distribution from demonstrations (Black et al., 2025b), giving us a direct setting in which to compare it with the original VLA. Consider one demonstrated segment $(o^-, P, o^+, U_{\mathcal{D}}) \sim \mathcal{D}$, corresponding to a trajectory in which the committed prefix P carries the system from o^- to o^+, followed by continuation $U_{\mathcal{D}}$. Across the demonstration distribution $\mathcal{D}$, this defines two natural prediction problems and their population targets:

$$\begin{array}{rcl@{\qquad}rcl} (o^-, P) & \longrightarrow & U_{\mathcal{D}} & \pi_{\text{TT-RTC}}(U \mid o^-, P) & \approx & p_{\mathcal{D}}(U \mid o^-, P) \\ o^+ & \longrightarrow & U_{\mathcal{D}} & \pi_{\text{VLA}}(A \mid o^+) & \approx & p_{\mathcal{D}}(U \mid o^+) \end{array} \tag{2}$$

Both policies are therefore trained from the same demonstrations to predict the same future action segment; only the point in the trajectory on which that prediction is conditioned differs. Assuming a fully observed system with deterministic dynamics, the handoff observation is itself determined by the earlier observation and committed prefix, $o^+ = f(o^-, P)$. Since there is no uncertainty about the state reached at handoff, one might expect predicting the continuation from (o^-, P) to recover the same distribution as querying the policy at o^+. This formally captures our central analysis question at the population level: $p_{\mathcal{D}}(U \mid o^-, P) \stackrel{?}{=} p_{\mathcal{D}}(U \mid o^+)$. Proposition 1 characterizes its answer.

Proposition 1 (Continuation mismatch)**.** *Let O^- and O^+ denote the random observations at inference time and handoff under $\mathcal{D}$; P and U denote the committed prefix and subsequent continuation. Assume fully observed deterministic dynamics, such that $O^+ = f(O^-, P)$. Then*

$$\mathbb{E}_{O^-,P}\left[D_{\text{KL}}\big(p_{\mathcal{D}}(U \mid O^-, P) \,\|\, p_{\mathcal{D}}(U \mid O^+)\big)\right] = I_{\mathcal{D}}(U; O^-, P \mid O^+). \tag{3}$$

Consequently, the two population continuation targets coincide almost everywhere if and only if the demonstrated continuation satisfies the Markov property with respect to the handoff observation:

$$p_{\mathcal{D}}(U \mid O^-, P) = p_{\mathcal{D}}(U \mid O^+) \quad \textit{a.e.} \qquad \Longleftrightarrow \qquad U \perp (O^-, P) \mid O^+. \tag{4}$$

That is, once O^+ is known, the earlier observation O^- and committed prefix P provide no additional information about the continuation U.

Surprisingly, deterministic dynamics alone are not sufficient for the two continuation distributions to coincide. Recent work studying robotic imitation learning finds that this property does not generally hold in human demonstrations (Lazzati et al., 2026; Zeng et al., 2026). For such non-Markovian data, even a perfectly learned $\pi_{\text{TT-RTC}}$ will generally produce a different action distribution from the VLA. In our method, we therefore study how π_{async} can be aligned directly with π_{VLA} to preserve the VLA's range of possible responses. Appendix A provides the proof of Proposition 1 and discusses how the argument extends beyond deterministic dynamics and full observability.

4 Method

Starting from a base policy π_{VLA} trained on demonstrations $\mathcal{D}$, we use its learned action distribution as the reference for constructing π_{async}. We exploit this reference at two stages of asynchronous execution. First, for the continuation generated at o^-, we change what the asynchronous policy is trained to predict by aligning its output with the VLA distribution that will be relevant at handoff. Second, once o^+ becomes available, we use the VLA distribution conditioned on that observation to choose among continuations generated earlier. We realize these two interventions through **Recursive Flow-Field Distillation** and **Propose–Resolve**, respectively.

4.1 Recursive Flow-Field Distillation

Because the training objectives below operate on full H-step action chunks, we write each demonstrated transition as $(o^-, P, o^+, B_{\mathcal{D}})$, where $U_{\mathcal{D}} = B_{\mathcal{D}}[d : d+s]$ is the executed continuation analyzed in Section 3. We retain the conditioning (o^-, P), but replace the demonstrated future actions with supervision from the frozen VLA conditioned on o^+. A direct implementation, which we call *teacher-sample supervision*, samples a full teacher chunk $B_V \sim \pi_{\mathrm{VLA}}(\cdot \mid o^+)$ and uses $(o^-, P, B_{V,0:H-d})$ as an ordinary continuation-training example. However, this exposes the student to the teacher's generative distribution only through sampled action chunks. Because both teacher and student are flow policies, we can instead transfer the teacher's vector field directly.

Given a teacher chunk $B_V \sim \pi_{\mathrm{VLA}}(\cdot \mid o^+)$, we sample noise ϵ and a flow time τ and construct the intermediate flow state $z_\tau = \tau\epsilon + (1-\tau)B_V$. For the asynchronous policy, we shift the first $H-d$ positions of this intermediate state behind the committed prefix, yielding $\bar{z}_\tau = [P; z_{\tau,0:H-d}]$. We denote the resulting asynchronous policy by π_{RFD} and its velocity field by v_{RFD}. We then match its continuation velocities at positions $d{:}H$ to the VLA's velocities at positions $0{:}H-d$:

$$\mathcal{L}_{\mathrm{RFD}} = \mathbb{E}_{B_V,\epsilon,\tau}\left[\mathrm{MSE}\big(v_{\mathrm{RFD}}(\bar{z}_\tau, \tau \mid o^-)_{d:H}, v_{\mathrm{VLA}}(z_\tau, \tau \mid o^+)_{0:H-d}\big)\right]. \tag{5}$$

We call this procedure *recursive* because the VLA learned from the demonstrations is itself fed back into the learning process: its learned flow field provides the supervision used to train π_{RFD}. Proposition 2 characterizes the population continuation distribution induced by $\mathcal{L}_{\mathrm{RFD}}$. Appendix B provides the proof of Proposition 2 and compares RFD with teacher-sample supervision.

Proposition 2 (Target of π_{RFD})**.** *Let $\mathcal{D}$ induce any joint distribution over (O^-, P, O^+). Assume that the frozen VLA provides the exact flow-matching field for the interpolation used in training, that π_{RFD} reaches the population optimum of the RFD objective, and that its flow is integrated exactly. Then the induced distribution over the executed continuation is*

$$\pi^\star_{\mathrm{RFD}}(\cdot \mid o^-, P) = \mathbb{E}_{O^+ \sim p_{\mathcal{D}}(\cdot \mid o^-, P)}\left[\pi_{\mathrm{VLA}}(\cdot \mid O^+)\right]. \tag{6}$$

Under the assumptions of Proposition 1—full observability and deterministic dynamics, such that (o^-, P) uniquely determines o^+—Equation (6) reduces to $\pi^\star_{\mathrm{RFD}}(\cdot \mid o^-, P) = \pi_{\mathrm{VLA}}(\cdot \mid o^+)$. Thus, RFD removes the continuation mismatch identified in Section 3, even when the demonstrations are non-Markovian. In practice, these assumptions do not generally hold, which Equation (6) naturally encapsulates through uncertainty in $p_{\mathcal{D}}(O^+ \mid o^-, P)$. In this case, where (o^-, P) does not uniquely determine o^+, $\pi^\star_{\mathrm{RFD}}$ must average the VLA's distributions over the possible O^+ consistent with the information available at (o^-, P). Consequently, $\pi^\star_{\mathrm{RFD}}$ is not in general guaranteed to match $\pi_{\mathrm{VLA}}(\cdot \mid o^+)$; the remaining discrepancy reflects uncertainty over the actual handoff-conditioned VLA distribution that cannot be resolved from (o^-, P) alone.

4.2 Propose–Resolve

At handoff, the observation o^+ provides information that was unavailable when the continuation was generated from o^-. The distribution $\pi_{\mathrm{VLA}}(\cdot \mid o^+)$ captures how the VLA would react to this newly observed state, but sampling such a reaction requires the very inference delay we seek to avoid. Our second insight is that benefiting from this VLA reaction does not require sampling a new action from it. If multiple continuations U have already been prepared, we can instead evaluate how compatible each is with $\pi_{\mathrm{VLA}}(\cdot \mid o^+)$. Consider K candidate continuations $U^{1:K}$ *proposed* in advance by an asynchronous policy π_{async}. Once o^+ becomes available, we *resolve* among these candidates by seeking the one with highest likelihood under the handoff-time VLA distribution:

$$k^\star = \arg\max_{k \in \{1,\dots,K\}} \pi_{\mathrm{VLA}}(U^k \mid o^+). \tag{7}$$

This resolution can only begin once o^+ has been received. To dispatch the next action without waiting, it must therefore complete within a single control step and be substantially cheaper than generating a new VLA sample. We learn a lightweight conditional density approximation for this purpose. For each training observation o^+, we draw M continuations $U_V^{1:M} \sim \pi_{\mathrm{VLA}}(\cdot \mid o^+)$ and estimate their per-coordinate mean $\hat{\mu}(o^+)$ and standard deviation $\hat{\sigma}(o^+)$. A lightweight predictor conditioned on o^+ learns the corresponding parameters $\mu_\phi(o^+)$ and $\sigma_\phi(o^+)$, defining

$$\hat{\pi}_\phi(U \mid o^+) = \mathcal{N}\big(U; \mu_\phi(o^+), \mathrm{diag}\,\sigma^2_\phi(o^+)\big). \tag{8}$$

At inference, we use $\hat{\pi}_\phi$ in place of π_{VLA} in Equation 7 to rank the prepared continuations. Notably, this replaces the potentially multimodal VLA distribution with a diagonal Gaussian. For action generation, such a unimodal representation is fundamentally problematic in multimodal settings, since averaging across distinct modes can produce actions that correspond to none of them (Chi et al., 2023). For our resolution, however, this failure mode is not necessarily inherited: the Gaussian only ranks already-generated continuations and cannot synthesize an averaged trajectory itself. Intuitively, the burden of representing multimodal action structure therefore remains with the proposer, while the resolver provides a ranking of those proposals informed by o^+. Appendix C formalizes the relationship between the Gaussian score and π_{VLA}, and provides a more grounded analysis, with examples, of when unimodal scoring is and is not informative for multimodal distributions. Finally, our complete method uses Recursive Flow-Field Distillation to obtain the asynchronous policy π_{RFD} and this policy to generate the proposals for Propose–Resolve.

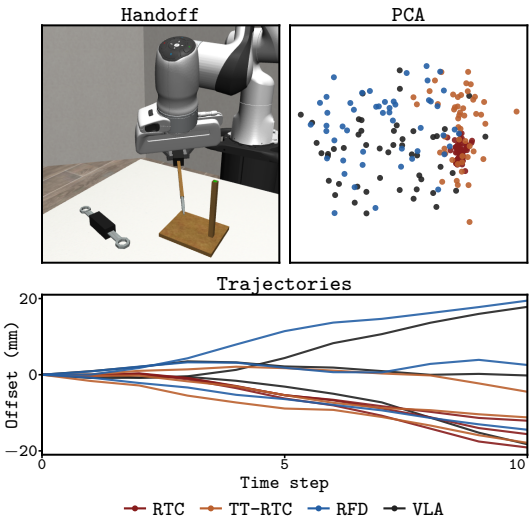
5 Experiments

Experimental setup. We evaluate a flow-based VLA, SmolVLA (Shukor et al., 2025), on three RoboMimic tasks (Mandlekar et al., 2022) (ToolHang, Square, and Transport) and on LIBERO (Liu et al., 2023). We compare against three approaches to asynchronous execution: (i) RTC (Black et al., 2025a) modifies flow generation at inference time to remain compatible with the committed prefix and preceding plan. (ii) TT-RTC (Black et al., 2025b) instead learns prefix-conditioned continuation generation during training, as analyzed in Section 3. (iii) PAINT (Ho et al., 2026) retains the frozen policy and uses inference-time inversion and repainting to generate stochastic prefix-compatible continuations. Our complete method combines Recursive Flow-Field Distillation (RFD) with Propose–Resolve (PR). Detailed training and evaluation configurations are provided in Appendix E.

5.1 Distribution Alignment

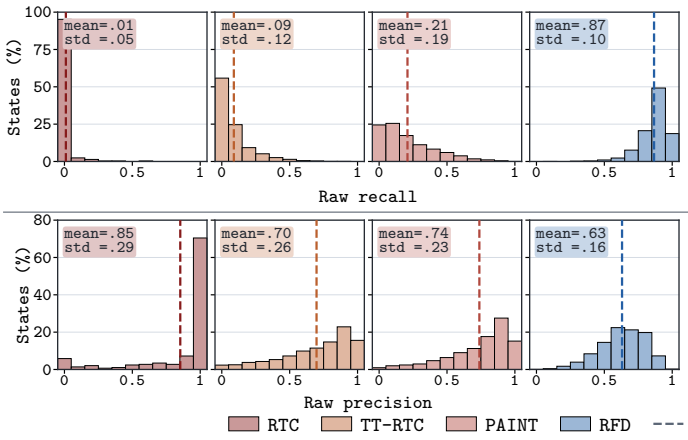
We first investigate how the action distributions produced by the asynchronous methods compare with the distribution of the original VLA. In particular, TT-RTC provides an empirical probe of the continuation-target mismatch characterized in Proposition 1. RFD, in turn, tests whether the future-VLA supervision characterized in Proposition 2 recovers the VLA action distribution in practice.

Figure 3 exemplifies on a single handoff in ToolHang how we compare the distributions. For the same handoff tuple (o^-, P, o^+), we approximate each asynchronous policy's continuation distribution by sampling many continuations conditioned on (o^-, P), and approximate the VLA distribution by repeatedly sampling from o^+. The trajectory panel shows how the sampled actions evolve through physical space, while the PCA projection makes differences in distributional coverage easier to see. RTC is tightly concentrated, TT-RTC occupies a broader region, and RFD spans a region comparable to the VLA. We quantify this using support recall and precision. Recall measures how much of the VLA's range of possible actions is retained by the asynchronous distribution, while precision measures how much of the asynchronous distribution remains within the range of actions supported by the VLA. Figure 3 reports both metrics for this state. Because support is estimated from finite sample banks, even two independent VLA banks achieve only about 85% self-coverage.



Method	Raw precision (%)	Raw recall (%)
RTC	99.9	0.0
TT-RTC	87.1	14.1
RFD	60.6	86.1
VLA (self-coverage)	85.6	85.6

Figure 3: **One ToolHang handoff at $d = 10$.**



	RoboMimic		LIBERO	
	Demo	Rollout	Demo	Rollout
RTC	0.03	0.01	0.08	0.11
TT-RTC	0.06	0.08	0.21	0.29
PAINT	0.17	0.13	0.46	0.52
RFD	**0.84**	**0.48**	**0.83**	**0.82**
VLA	0.89	0.88	0.86	0.86

Recall

	RoboMimic		LIBERO	
	Demo	Rollout	Demo	Rollout
RTC	0.69	0.57	0.75	0.75
TT-RTC	0.62	0.39	0.73	0.68
PAINT	0.62	0.40	0.79	0.69
RFD	0.47	0.30	0.69	0.64
VLA	0.89	0.88	0.86	0.86

Precision

Figure 4: **Distribution comparison at o^+.** For each handoff tuple (o^-, P, o^+) at $d = 10$, we draw 128 continuations from each asynchronous method conditioned on (o^-, P) and independent banks of 128 support and 128 query continuations from $\pi_{\text{VLA}}(\cdot \mid o^+)$. **Left:** Per-state precision and recall over $N = 2{,}775$ ToolHang demonstration handoffs (291 for RTC), shown as histograms. **Right:** Mean precision and recall across RoboMimic and LIBERO, evaluated on both demonstration states and states from VLA rollouts. Appendix D details how precision and recall are computed, extends this analysis across delays d, and additionally reports the distribution obtained under the teacher-sample supervision introduced in Section 4.1.

The single-state example already shows the pattern we want to measure: RFD reaches approximately the VLA's own self-coverage in recall, whereas TT-RTC recovers only 14% of the VLA support. We now compute precision and recall independently at every handoff state across different ToolHang demonstration trajectories to determine whether this behavior persists beyond a single example. Figure 4 (left) shows the resulting distribution of per-state scores. The aggregate statistics confirm that RTC recovers almost none of the VLA support, while TT-RTC and PAINT broaden the continuation distribution but still miss a substantial part of the VLA's distribution. RFD instead reaches recall close to the VLA self-reference across most states. This supports the prediction motivated by Proposition 1: replacing demonstration-continuation supervision with supervision from the VLA at handoff recovers substantially more of the action distribution the VLA itself would produce. Figure 4 (right) summarizes the same comparison across RoboMimic and LIBERO, on both demonstration and rollout states. LIBERO shows the same pattern: RFD's recall is close to the VLA self-reference, while the baseline distributions only cover a fraction of that distribution. On RoboMimic rollout states, RFD's recall decreases to 0.48. Appendix D.1 gives a per-task comparison, which shows that this drop comes from Square's recall reducing to almost 0 as d increases.

RFD shows a clear recall–precision asymmetry: its recall remains close to the VLA's self-coverage, while its precision ranges from 35% to 80% of the corresponding VLA self-coverage, depending on the setting. Equation 6 predicts that RFD matches the handoff-time VLA distribution only when (o^-, P) uniquely determines o^+; otherwise, it averages over the VLA distributions associated with possible handoff observations. The observed precision gap is therefore consistent with such averaging before the actual o^+ is known, although approximation and finite-sample effects can also contribute. Importantly, high recall alone does not imply better execution: a broader proposal distribution can recover more of the VLA's possible responses while still containing actions that are inappropriate for the particular handoff reached. We therefore next test whether these distributional differences translate into improved closed-loop performance.

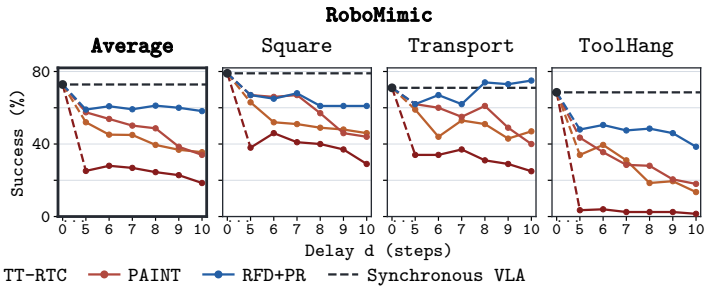
5.2 Real-Time Execution

Figure 5 shows closed-loop success across RoboMimic and LIBERO as the asynchronous delay d increases. RFD+PR remains comparatively stable while the asynchronous baselines progressively deteriorate. On RoboMimic, the complete method retains approximately 80% of the original VLA's success, while on LIBERO it matches the original VLA on average. These results show that asyn-

LIBERO

Success (%)

Delay d	5	10	15	20
RTC	47.5	45.1	43.8	40.3
TT-RTC	56.3	52.9	49.9	44.6
PAINT	53.0	46.5	45.8	43.0
RFD+PR	**64.3**	**63.4**	**63.8**	**60.7**
Sync VLA	62.5			



LIBERO Average

	RTC	TT-RTC	PAINT	RFD
K1	45.1	**52.9**	46.5	51.0
PR	58.2	62.5	61.2	**63.4**
Δ (pp)	+13.1	+9.6	**+14.7**	+12.4

RoboMimic Average

	RTC	TT-RTC	PAINT	RFD
K1	18.5	35.5	34.0	**41.0**
PR	21.5	50.8	48.0	**58.2**
Δ (pp)	+3.0	+15.3	+14.0	**+17.2**

Figure 5: **Real-time execution success and ablation. Top:** Rollout success on LIBERO and RoboMimic across asynchronous delays, compared with the synchronous VLA reference. Our method uses $K = 32$ RFD proposals followed by Propose–Resolve. **Bottom:** Ablation at $d = 10$, comparing each asynchronous proposer with a single continuation ($K = 1$) and with Propose–Resolve over $K = 32$ candidates.

chronous execution can preserve much of the original policy's performance when alternatives are prepared in advance and selected at handoff.

Component attribution. The ablation in Figure 5 (bottom) confirms that the gains do not come from RFD alone. With a single proposal ($K = 1$), RFD is strongest on RoboMimic and performs similarly to TT-RTC on LIBERO. After resolving $K = 32$ candidates, however, RFD+PR achieves the highest average success on both benchmarks, indicating that the broader proposal distribution becomes most useful when the realized handoff observation can determine which continuation to execute. With SmolVLA, the resolution of the candidates takes approximately $12\,\mathrm{ms}$, so this additional computation step remains compatible with real-time control. Appendix E.3 describes the resolver implementation and provides a more detailed timing attribution.

Taken together with Section 5.1, these ablations show that the asynchronous methods induce meaningfully different continuation distributions, but the usefulness of those distributions depends strongly on the task. With a single proposal ($K = 1$), this difference is already visible in closed-loop success: RTC performs much worse than the broader continuation methods on RoboMimic, whereas all four proposers are considerably closer on LIBERO. Handoff-time selection shows the same contrast: on RoboMimic, PR improves RTC by only 3.0 percentage points but improves RFD by 17.2 points, showing that the additional options provided by RFD are particularly valuable for task success. In contrast, on LIBERO, PR improves even RTC by 13.1 points, and all four proposers end up performing similarly, indicating that the different continuation distributions have much less impact on task success in this setting. Thus, the continuation distribution and task success are not related in a simple one-to-one way: reproducing more of the VLA distribution is important in some settings, while in others substantially different continuation distributions can perform similarly. These results point to a broader tension between renewing behavior from the current observation and preserving coherence with the preceding plan.

6 Related Work

Generalist Robot Policies. Generalist robot policies increasingly combine large-scale robot data with pretrained vision-language representations and action heads based on autoregressive prediction, diffusion, or flow matching (Brohan et al., 2023b;a; Embodiment Collaboration et al., 2025; Octo Model Team et al., 2024; Kim et al., 2024; Li et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025; Shukor et al., 2025; NVIDIA et al., 2025). Their growing inference cost has motivated work on faster and more efficient execution.

Real-Time VLA Execution. One direction reduces the cost of the policy query itself, through optimized action heads, parallel prediction, or faster generative decoding (Kim et al., 2025; Li et al., 2026; Lu et al., 2026; Luan et al., 2026). Our work instead considers the complementary regime in which policy inference remains slower than the desired control rate and must overlap with execution.

Asynchronous Real-Time Execution. RTC, TT-RTC, and PAINT realize this overlap through different mechanisms for generating prefix-compatible continuations (Black et al., 2025a;b; Ho et al., 2026), but all must act from information available before handoff. A growing body of work addresses the resulting staleness by predicting execution-time state or visual features (Tang et al., 2025; Jiang et al., 2026), incorporating newly measured state during generation (Park & Tulsiani, 2026), correcting existing predictions (Sendai et al., 2025), or using future observations for supervision or candidate selection (Zhu et al., 2026; Chen et al., 2026). These approaches address the same stale-conditioning problem through complementary mechanisms. Our focus is instead on characterizing the distributional mismatch induced by continuation learning, aligning the asynchronous proposal distribution with the VLA at handoff, and studying how those proposals can be resolved using the realized handoff observation.

Policy Distillation. Generative-policy distillation provides another route to efficient control by transferring expensive iterative policies into cheaper generators (Prasad et al., 2024; Wang et al., 2024; Yin et al., 2024b;a). *Recursive Flow-Field Distillation* differs in that teacher and student correspond to different points in the execution timeline, transferring the VLA behavior at handoff into an asynchronous continuation policy conditioned earlier.

7 Discussion

This work studies asynchronous VLA execution from a distributional perspective. We show that existing continuation objectives generally do not recover the action distribution that the original VLA would produce at handoff, and introduce a new learning objective that instead targets the future VLA distribution. Recursive Flow-Field Distillation provides a practical realization of this objective, and our support analysis provides strong evidence that it learns the intended distribution substantially more faithfully than existing continuation methods. Combined with Propose–Resolve, this enables asynchronous execution with performance close to the original synchronous policy without requiring a new generative VLA pass at handoff. More broadly, our contribution is to frame asynchronous execution as a separation between *which futures should be prepared before handoff* and *how the realized observation should resolve among them afterward*; RFD and PR provide one concrete instantiation of this view.

Limitations. Our current resolver has two main limitations. First, although resolution avoids flow generation, it still extracts features from the frozen VLA after o^+ arrives. Its latency is therefore partly tied to the underlying VLA architecture; a resolver based on a lightweight state encoder could make the same framework applicable at substantially higher control rates and to slower VLAs. Second, we approximate the handoff-time VLA distribution with a diagonal Gaussian. This deliberately simple model is sufficient for ranking useful proposals in our experiments, but cannot represent the full multimodal structure of the original policy. More expressive conditional density models could provide a stronger resolution rule without changing the proposal mechanism.

Future Work. More broadly, our results suggest that reproducing the handoff-time VLA distribution is not always the universally desirable continuation objective. On some tasks, preparing alternatives that resemble a newly queried VLA is important; on others, continuation-based proposals remain equally effective once the reached observation is used for selection. This points to a more general question for asynchronous policies: *what should the policy predict before the future state is known?* Rather than treating continuation of the previous plan or renewal toward the future VLA as universally correct, future work could study when each objective is appropriate and how both can be combined. We view asynchronous execution as a useful setting for studying this broader trade-off between preserving prior intent and renewing behavior from newly available observations.

GENERATIVE AI USE STATEMENT

Generative AI tools (ChatGPT, OpenAI) were used during the preparation of this work to assist with drafting and editing text, developing and checking mathematical claims and proofs, providing feedback on methodology and experiments, generating and debugging experimental code, and assisting with interpretation of results. All AI-assisted text, code, and mathematical arguments were reviewed and verified by the authors, who take full responsibility for the final manuscript and reported results.

REPRODUCIBILITY STATEMENT

We provide the assumptions and complete proofs underlying our theoretical results in Appendices A–B, together with additional analysis of the resolver in Appendix C. Appendix D provides additional distributional evaluations, and Appendix E describes the policy training, Recursive Flow-Field Distillation, Propose–Resolve implementation, and timing setup used in our experiments. We report the model checkpoints, action horizons, execution intervals, teacher-sampling procedure, resolver construction, and computational setup required to reproduce the main experimental configurations.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael 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, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control, 2024. URL https://arxiv.org/abs/2410.24164. arXiv:2410.24164.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-Time Execution of Action Chunking Flow Policies. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, 2025a. doi: 10.52202/085713-1122. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/300ccb2187dedd4edcc07f7e76d8e553-Abstract-Conference.html.

Kevin Black, Allen Z. Ren, Michael Equi, and Sergey Levine. Training-Time Action Conditioning for Efficient Real-Time Chunking, 2025b. URL https://arxiv.org/abs/2512.05964. arXiv:2512.05964.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control, 2023a. URL https://arxiv.org/abs/2307.15818. arXiv:2307.15818.

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, 2023b. URL https://arxiv.org/abs/2212.06817. arXiv:2212.06817.

Wenxi Chen, Kaidi Zhang, Chi Lin, Zhiyuan Zhang, Yu She, Yuejiang Liu, Raymond A. Yeh, Shaoshuai Mou, and Yan Gu. DREAM-Chunk: Reactive Action Chunking with Latent World Model, 2026. URL https://arxiv.org/abs/2606.18589. arXiv:2606.18589.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://roboticsproceedings.org/rss19/p026.html.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2 edition, 2006. ISBN 9780471241959. doi: 10.1002/047174882X. URL https://onlinelibrary.wiley.com/doi/book/10.1002/047174882X.

Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu,

Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic Learning Datasets and RT-X Models, 2025. URL https://arxiv.org/abs/2310.08864. arXiv:2310.08864.

Trong-Bao Ho, Quang-Tan Nguyen, Thien-Loc Ha, Gia-Binh Nguyen, Viet-Thanh Nguyen, Long Dinh, Minh N. Vu, Duy M. H. Nguyen, An Thai Le, and Ngo Anh Vien. Start Right, Arrive Right: Asynchronous Execution via Initial Noise Selection, 2026. URL https://arxiv.org/abs/2606.19774. arXiv:2606.19774.

Hai Jiang, Yixian Zou, Binbin Liang, Boqian Liu, Fanman Meng, and Shuaicheng Liu. FuturERTC: Real-Time Robot Execution with Anticipatory-Conditioned Action Chunking, 2026. URL https://arxiv.org/abs/2607.24008. arXiv:2607.24008.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag 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, 2024. URL https://arxiv.org/abs/2406.09246. arXiv:2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success, 2025. URL https://arxiv.org/abs/2502.19645. arXiv:2502.19645.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved Precision and Recall Metric for Assessing Generative Models. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/0234c510bc6d908b28c70ff313743079-Abstract.html.

Filippo Lazzati, Kyle Stachowicz, William Chen, Alberto Maria Metelli, Andrew Wagenmaker, and Sergey Levine. Why Does Action Chunking Improve Behavioral Cloning Performance in Robotic Control?, 2026. URL https://arxiv.org/abs/2608.02547. arXiv:2608.02547.

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, 2024. URL https://arxiv.org/abs/2411.19650. arXiv:2411.19650.

Zekai Li, Jiaming Tang, and Zhijian Liu. FlashVLA: Streaming Action Decoding for Fast and Asynchronous VLA Inference, 2026. URL https://arxiv.org/abs/2608.27384. arXiv:2608.27384.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

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. Curran Associates, Inc., 2023. doi: 10.52202/075280-1939. URL https://papers.nips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Yuxiang Lu, Zhe Liu, Xianzhe Fan, Zhenya Yang, Jinghua Hou, Junyi Li, Kaixin Ding, and Hengshuang Zhao. FASTER: Rethinking Real-Time Flow VLAs, 2026. URL https://arxiv.org/abs/2603.19199. arXiv:2603.19199.

Wuyang Luan, Junhui Li, Weiguang Zhao, Wenjian Zhang, Tieru Wu, and Rui Ma. SnapFlow: One-Step Action Generation for Flow-Matching VLAs via Progressive Self-Distillation, 2026. URL https://arxiv.org/abs/2604.05656. arXiv:2604.05656.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1678–1690. PMLR, 2022. URL https://proceedings.mlr.press/v164/mandlekar22a.html.

Thomas P. Minka. Expectation Propagation for Approximate Bayesian Inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pp. 362–369, 2001. URL https://tminka.github.io/papers/ep/.

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, 2025. URL https://arxiv.org/abs/2503.14734. arXiv:2503.14734.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An Open-Source Generalist Robot Policy, 2024. URL https://arxiv.org/abs/2405.12213. arXiv:2405.12213.

Sungjae Park and Shubham Tulsiani. $\pi\mathbf{R}^2$: Reactive Real-time Flow Policies, 2026. URL https://arxiv.org/abs/2607.26055. arXiv:2607.26055.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael 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, 2025. URL https://arxiv.org/abs/2504.16054. arXiv:2504.16054.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency Policy: Accelerated Visuomotor Policies via Consistency Distillation, 2024. URL https://arxiv.org/abs/2405.07503. arXiv:2405.07503.

Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave No Observation Behind: Real-time Correction for VLA Action Chunks, 2025. URL https://arxiv.org/abs/2509.23224. arXiv:2509.23224.

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, 2025. URL https://arxiv.org/abs/2506.01844. arXiv:2506.01844.

Jiaming Tang, Yufei Sun, Yilong Zhao, Shang Yang, Yujun Lin, Zhuoyang Zhang, James Hou, Yao Lu, Zhijian Liu, and Song Han. VLASH: Real-Time VLAs via Future-State-Aware Asynchronous Inference, 2025. URL https://arxiv.org/abs/2512.01031. arXiv:2512.01031.

Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-Step Diffusion Policy: Fast Visuomotor Policies via Diffusion Distillation, 2024. URL https://arxiv.org/abs/2410.21257. arXiv:2410.21257.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T. Freeman. Improved Distribution Matching Distillation for Fast Image Synthesis, 2024a. URL https://arxiv.org/abs/2405.14867. arXiv:2405.14867.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T. Freeman, and Taesung Park. One-step Diffusion with Distribution Matching Distillation, 2024b. URL https://arxiv.org/abs/2311.18828. arXiv:2311.18828.

Michael Zeng, Abhinav Agarwal, Ajay Bati, Brian Lee, Siddharth Ancha, and Russ Tedrake. Revisiting Open-Loop Execution in Robotics: Toward Reactive, Higher-Performing Policies, 2026. URL https://arxiv.org/abs/2608.15938. arXiv:2608.15938.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://roboticsproceedings.org/rss19/p016.html.

Yixiang Zhu, Yonghao Chen, Zijie Yang, Yusong Hu, and Xinyu Chen. DEFLECT: Temporal Counterfactual Preference Learning for Delay-Robust Asynchronous VLAs, 2026. URL https://arxiv.org/abs/2605.19294. arXiv:2605.19294.

A Continuation Learning and VLA Replanning

A.1 Proof of Proposition 1

Proof. All expectations and conditional distributions are taken under $\mathcal{D}$. The conditional-KL representation of mutual information gives (Cover & Thomas, 2006, Ch. 2)

$$I_{\mathcal{D}}(U; O^-, P \mid O^+) = \mathbb{E}_{O^-,P,O^+}\left[D_{\mathrm{KL}}\big(p_{\mathcal{D}}(U \mid O^-, P, O^+) \,\|\, p_{\mathcal{D}}(U \mid O^+)\big)\right]. \tag{9}$$

Because $O^+ = f(O^-, P)$, additionally conditioning on O^+ does not change $p_{\mathcal{D}}(U \mid O^-, P)$, and the outer expectation over O^+ is redundant. Substitution therefore yields Eq. 3. By nonnegativity of KL, this expectation is zero exactly when its two conditional distributions agree almost everywhere. Conditional mutual information is zero exactly when $U \perp (O^-, P) \mid O^+$, establishing Eq. 4. □

A minimal deterministic counterexample. Consider fully observed scalar dynamics $O_{k+1} = O_k + a_k$, with a one-action prefix and a one-action continuation ($d = s = 1$). The dataset contains two equally likely trajectories:

$$-1 \xrightarrow{P=+1} 0 \xrightarrow{U=+1} +1, \qquad +1 \xrightarrow{P=-1} 0 \xrightarrow{U=-1} -1. \tag{10}$$

In each demonstration, the demonstrator repeats the action used to reach the origin. Both trajectories therefore have the same handoff observation, but the prefix reveals which continuation follows. Specifically,

$$p_{\mathcal{D}}(U \mid O^+ = 0) = \tfrac{1}{2}\delta_{+1} + \tfrac{1}{2}\delta_{-1}, \qquad p_{\mathcal{D}}(U \mid O^- = -1, P = +1) = \delta_{+1}, \tag{11}$$

where δ_a denotes a point mass at a; the other earlier context similarly gives δ_{-1}. Each prefix-conditioned distribution has KL divergence $\log 2$ from the handoff-conditioned mixture, so the expected gap is $\log 2$ nats. No uncertainty about the physical state is involved: the difference arises because demonstration behavior depends on how that state was reached. Under exact population learning, TT-RTC retains this dependence, whereas the observation-conditioned VLA represents both continuations.

A.2 Beyond Deterministic Handoffs

Proposition 1 deliberately removes uncertainty about the reached observation to isolate the effect of history-dependent demonstration behavior. We now drop this restriction while continuing to compare the population targets in Eq. 2.

Uncertainty in $p_{\mathcal{D}}(O^+ \mid O^-, P)$ can arise from stochastic dynamics or noisy observations. Partial observability can produce the same effect even with deterministic physical dynamics: different hidden states consistent with the earlier observation can lead to different handoff observations after the same prefix. The following analysis accommodates these cases directly through the joint distribution of the observed variables.

Comparison using the information available before handoff. The demonstration-continuation target averages over possible handoff observations. For comparison, define the corresponding average of the handoff-conditioned target:

$$\begin{aligned} p_{\mathcal{D}}(U \mid O^-, P) &= \mathbb{E}_{O^+ \mid O^-,P}\left[p_{\mathcal{D}}(U \mid O^-, P, O^+)\right], \\ \bar{p}_{\mathcal{D}}(U \mid O^-, P) &:= \mathbb{E}_{O^+ \mid O^-,P}\left[p_{\mathcal{D}}(U \mid O^+)\right]. \end{aligned} \tag{12}$$

Both expressions use the same distribution over reached observations. They differ in whether the continuation distribution retains the earlier observation and prefix after the handoff observation is specified. Consequently, $U \perp (O^-, P) \mid O^+$ remains sufficient for these two averaged targets to coincide.

Applying joint convexity of KL to Eq. 12, then averaging over (O^-, P), gives

$$\begin{aligned} &\mathbb{E}_{O^-,P}\left[D_{\mathrm{KL}}\big(p_{\mathcal{D}}(U \mid O^-, P) \,\|\, \bar{p}_{\mathcal{D}}(U \mid O^-, P)\big)\right] \\ &\quad \le \mathbb{E}_{O^-,P,O^+}\left[D_{\mathrm{KL}}\big(p_{\mathcal{D}}(U \mid O^-, P, O^+) \,\|\, p_{\mathcal{D}}(U \mid O^+)\big)\right] \\ &\quad = I_{\mathcal{D}}(U; O^-, P \mid O^+). \end{aligned} \tag{13}$$

When $O^+ = f(O^-, P)$, each mixture reduces to a single conditional distribution and the bound becomes the equality in Proposition 1. Otherwise, averaging can cancel differences between the conditional distributions. The conditional-independence condition is therefore sufficient, but no longer necessary, for equality of the averaged targets. Stochasticity should not be interpreted as necessarily increasing the KL mismatch.

Comparison at the realized handoff. Equality to the averaged reference $\bar{p}_{\mathcal{D}}(U \mid O^-, P)$ does not imply equality to $p_{\mathcal{D}}(U \mid o^+)$ at the observation actually reached. Even when the demonstrations satisfy the conditional-independence condition, different possible handoff observations may induce different continuation distributions. A single distribution conditioned only on (o^-, P) cannot equal all of them unless they coincide across the possible handoff observations.

There are thus two distinct information questions: whether the earlier context adds information about demonstrated behavior once O^+ is known, and whether observing O^+ adds information about the relevant handoff-conditioned target. Proposition 1 isolates the first by eliminating the second. Under partial observability, history may also reveal hidden physical state, so even an expert that is Markovian in the full state need not satisfy the same property with respect to the observation alone.

B Population Analysis of Recursive Flow-Field Distillation

Proof of Proposition 2. Fix (o^-, P), let d be the prefix length, and set $m = H - d$. All expectations below are conditional on this fixed context. Draw $O^+ \sim p_{\mathcal{D}}(\cdot \mid o^-, P)$, then a full H-step teacher chunk B_V conditioned only on O^+, and independent Gaussian noise ϵ. With $Z_\tau = \tau\epsilon + (1-\tau)B_V$, write $X_\tau = Z_{\tau,0:m}$. We assume finite second moments, independent flow-time sampling with positive density on $(0,1)$, unrestricted population optimization, and sufficient regularity for a unique probability flow with well-defined endpoint limits. The exact-teacher assumption means that

$$v_{\mathrm{VLA}}(z, \tau \mid o^+) = \mathbb{E}\big[\epsilon - B_V \mid Z_\tau = z, \tau, O^+ = o^+\big]. \tag{14}$$

Denote the student's continuation field by $w(x,\tau) := v_{\mathrm{RFD}}([P;x], \tau \mid o^-)_{d:H}$ and its teacher target by $T_\tau := v_{\mathrm{VLA}}(Z_\tau, \tau \mid O^+)_{0:m}$. The population MSE minimizer is the conditional mean of its target. Applying Eq. 14 and the tower property gives

$$\begin{aligned} w^\star(x,\tau) &= \mathbb{E}[T_\tau \mid X_\tau = x, \tau] \\ &= \mathbb{E}[(\epsilon - B_V)_{0:m} \mid X_\tau = x, \tau]. \end{aligned} \tag{15}$$

The second equality uses the conditional independence of the teacher sample and noise from the earlier context given O^+. It averages over both the unknown handoff observation and the teacher's omitted last d coordinates.

Since $(\epsilon - B_V)_{0:m} = \partial_\tau X_\tau$, Eq. 15 is the marginal flow-matching field for the probability path of X_τ (Lipman et al., 2023). Exact integration from $\tau = 1$ to $\tau = 0$, with P fixed, therefore transports Gaussian continuation noise to the law of $(B_V)_{0:m}$. Taking the first $s \le m$ actions yields

$$\pi^\star_{\mathrm{RFD}}(\cdot \mid o^-, P) = \mathbb{E}_{O^+ \mid o^-, P}\left[\pi_{\mathrm{VLA}}(\cdot \mid O^+)\right], \tag{16}$$

where both policies denote distributions over the executed s-step segment. This proves the proposition. □

Relationship to teacher-sample supervision. Under the same population sampling law and MSE normalization, teacher-sample supervision uses the target $G := (\epsilon - B_V)_{0:m}$ instead of T_τ. Equation 14 implies $\mathbb{E}[G - T_\tau \mid Z_\tau, \tau, O^+] = 0$. Because $w(X_\tau, \tau)$ is determined by Z_τ and the fixed early context, expanding the squared error makes the cross term vanish, giving

$$\mathcal{L}_{\mathrm{sample}}(w) = \mathcal{L}_{\mathrm{RFD}}(w) + \mathbb{E}[\mathrm{MSE}(G, T_\tau)]. \tag{17}$$

The final term is independent of the student, so the two objectives have the same population minimizers. RFD replaces a sample-derived velocity label with the teacher's conditional mean velocity; it changes the supervision, not the ideal target. These identities concern population sampling and an exact teacher, rather than a guarantee for a fixed finite teacher bank or finite-step implementation.

C Interpreting Gaussian Resolution

Moment-matched Gaussian approximation. Fix a handoff observation o^+ and write $p(U) := \pi_{\mathrm{VLA}}(U \mid o^+)$, with each continuation represented by n scalar coordinates. Assume finite means $\mu_j = \mathbb{E}_p[U_j]$ and positive finite variances $\sigma_j^2 = \mathrm{Var}_p(U_j)$. For a diagonal Gaussian $q_{m,v} = \mathcal{N}(m, \operatorname{diag} v)$ with $v_j > 0$, its expected negative log density is

$$\mathbb{E}_p[-\log q_{m,v}(U)] = \frac{1}{2}\sum_{j=1}^{n}\left[\log(2\pi v_j) + \frac{\sigma_j^2 + (\mu_j - m_j)^2}{v_j}\right]. \tag{18}$$

Minimizing coordinate-wise gives $m_j = \mu_j$ and $v_j = \sigma_j^2$. Thus, whenever the forward KL divergence is finite,

$$q^\star = \arg\min_{q \in \mathcal{G}_{\mathrm{diag}}} D_{\mathrm{KL}}(p\|q) = \mathcal{N}(\mu, \operatorname{diag}\sigma^2). \tag{19}$$

This is the standard moment-matching characterization of Gaussian approximation (Minka, 2001). The empirical moments estimated from M teacher samples in Section 4.2 approximate these population parameters, and the learned predictor approximates their dependence on o^+. This justifies the moment targets; it does not assume that p is Gaussian or that the learned $\hat{\pi}_\phi$ equals $q^\star$ exactly.

What the Gaussian ranking measures. For a fixed handoff observation, the Gaussian normalizing factor is identical across candidates. Hence,

$$\arg\max_k q^\star(U^k) = \arg\min_k \sum_{j=1}^{n} \frac{(U_j^k - \mu_j)^2}{\sigma_j^2}. \tag{20}$$

Deviations along coordinates where the teacher varies little are penalized more strongly than deviations along coordinates where it varies substantially. This score also has an interpretation directly under the original teacher, without assuming Gaussianity. For any fixed candidate u and an independent teacher continuation $A \sim p$,

$$\mathbb{E}_{A\sim p}\left[\sum_{j=1}^{n}\frac{(u_j - A_j)^2}{\sigma_j^2}\right] = \sum_{j=1}^{n}\frac{(u_j - \mu_j)^2}{\sigma_j^2} + n. \tag{21}$$

To obtain this identity, expand $u_j - A_j = (u_j - \mu_j) - (A_j - \mu_j)$: the cross term has zero expectation, and the remaining variance term contributes one per coordinate. Gaussian ranking therefore selects the available candidate minimizing expected variance-normalized squared discrepancy to a teacher draw. This interpretation requires only the stated moment assumptions, not a unimodal teacher distribution.

A multimodal example. Consider the two-dimensional teacher

$$p = \tfrac{1}{2}\mathcal{N}((-1,1), \eta^2 I) + \tfrac{1}{2}\mathcal{N}((1,1), \eta^2 I), \qquad 0 < \eta \ll 1. \tag{22}$$

Its mean is $(0,1)$ and its coordinate variances are $(1+\eta^2, \eta^2)$. Suppose the candidate bank contains $(-1,1)$, $(1,1)$, $(-1,-1)$, and $(1,-1)$. The first coordinate separates the teacher's two modes, whereas the second is consistently near 1. Equation 20 assigns the latter two candidates an additional penalty of $4/\eta^2$, strongly preferring either of the first two. The Gaussian need not represent the two modes separately to reject candidates that violate their shared second-coordinate behavior. Moreover, the resolver returns an existing candidate; it does not synthesize or execute the mean $(0,1)$.

Limitations and interpretation. The same example shows where this reasoning can fail. If $(0,1)$ is added to the candidate bank, the Gaussian ranks it above both component centers, although its teacher density is very small for small η. Selection avoids creating an averaged trajectory, but does not prevent selecting an undesirable intermediate candidate already present in the bank. Diagonal covariance also discards correlations across action coordinates and timesteps, and distributions with identical coordinate-wise moments receive identical Gaussian summaries. Consequently, Gaussian

ranking need not agree with the ideal VLA-likelihood ranking in Eq. 7, reconstruct the teacher distribution, or preserve its mode probabilities. Finite-sample moment estimates and imperfect parameter prediction introduce further approximation.

The resulting division of labor is therefore deliberate: the proposer supplies structured alternative continuations, while the resolver uses the handoff observation to compare them through a cheap moment-based score. The score need not serve as a faithful generative policy to be useful for this selection problem; whether its ranking is sufficient is evaluated through the closed-loop experiments.

D Additional Distribution Analysis

Support metric. For each handoff state, the VLA provides 256 action samples, split into 128 support samples and 128 independent query samples; each asynchronous method provides 128 proposals. Following Kynkäänniemi et al. (2019), we estimate the support of a sample set as the union of balls centered at its samples, with each radius given by the distance to that sample's fifth nearest neighbor within the set, excluding itself. Distances are computed over the policy-normalized executed action segment as

$$d(A, B) = \frac{\|\mathrm{vec}(A - B)\|_2}{\sqrt{sD_a}},$$

using only physical action coordinates. Precision is the fraction of asynchronous proposals contained in the estimated VLA support, while recall is the fraction of independent VLA queries contained in the estimated asynchronous support. Figures 3 and 4 report these raw quantities without normalization. VLA self-coverage is obtained by testing the VLA query bank against the VLA support bank. The supplementary plots in Figures 6 and 7 report relative precision and recall, dividing both raw metrics independently at each handoff state by that state's VLA self-coverage; these normalized values may exceed one.

VLA self-coverage reference. At $d = 10$, using the same states and equal task weighting as Figure 4, mean VLA self-coverage is 0.8900 on demonstration states and 0.8836 on rollout states for RoboMimic, and 0.8578 and 0.8592, respectively, for LIBERO. These averages first average self-coverage within each task, then equally weight the three RoboMimic tasks or 40 LIBERO tasks. For the ToolHang histogram population, mean self-coverage is 0.8954 (0.8906 for RTC's smaller state subset).

D.1 Generalization Across Delay

Figure 6 extends the distribution analysis across inference delays. On demonstration states, RFD maintains high recall across delays, indicating that the learned objective remains robust as the committed prefix grows. We additionally evaluate states encountered during VLA rollouts, where RFD must generalize to observations and prefixes not seen during training. This generalization remains strong on ToolHang and degrades only moderately on Transport, but drops substantially with increasing delay on Square. Thus, the main limitation in proposal coverage appears when predicting the handoff-time VLA behavior from out-of-distribution rollout states.

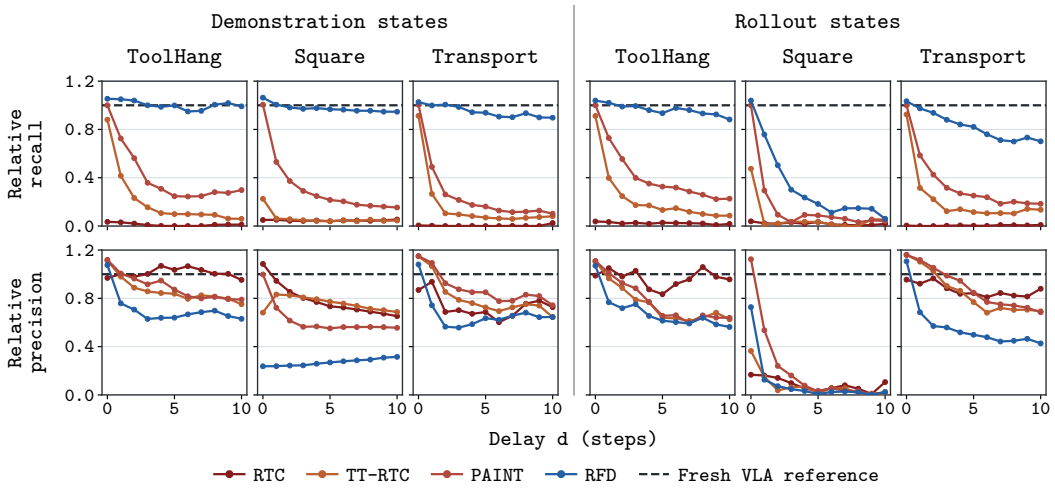


Figure 6: **Distribution alignment across delays.** The support comparison from Figure 4, extended across delays $d = 0, \ldots, 10$ on ToolHang, Square, and Transport, showing how relative recall and precision evolve on demonstration and rollout states.

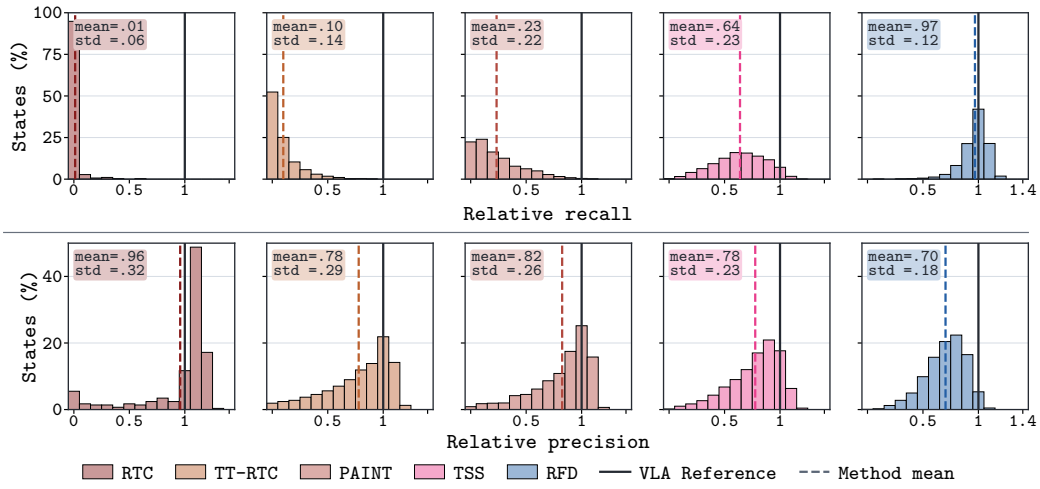


Figure 7: **Action support with teacher-sample supervision.** The ToolHang comparison from Figure 4, now including teacher-sample supervision (TSS) and reporting precision and recall relative to each state's VLA self-coverage.

D.2 Teacher-Sample Supervision

Figure 7 compares RFD with the teacher-sample supervision introduced in Section 4.1, which replaces demonstration continuations with samples from the VLA at o^+ but trains on the sampled endpoints rather than the teacher flow field. Teacher-sample supervision improves substantially over the original continuation objectives, but does not recover the VLA support as completely as RFD. We tested supervision with $K \in \{1, 8, 16\}$ teacher samples per condition and observed the same qualitative limitation, whereas RFD achieved near-complete recall directly. Although both objectives target the same teacher distribution in the population limit, transferring the teacher flow field provides a clear empirical advantage for learning that distribution in practice. We therefore use RFD as the proposer throughout the remaining experiments.

E Implementation Details

E.1 Setup

For RoboMimic, we initialize from `lerobot/smolvla_base` and fine-tune one SmolVLA policy per task (Square, Transport, ToolHang), training the action expert and projection layers while keeping the vision-language backbone frozen. We use the selected 60k-update checkpoints for all three tasks. For LIBERO, we use the released `HuggingFaceVLA/smolvla_libero` checkpoint directly and train only our additional asynchronous components. RTC, TT-RTC, and PAINT are implementations of the published algorithms adapted to the corresponding backbone and execution protocol.

Evaluation protocol. All policies use action horizon $H = 50$, with ten Euler steps per flow integration. We execute $s = 10$ actions per chunk on RoboMimic and $s = 25$ on LIBERO. On RoboMimic, we evaluate 100 episodes each on Square and Transport and 200 on ToolHang; the reported average weights the three task success rates equally. On LIBERO, we evaluate all 40 tasks with 50 episodes per task, for 2,000 episodes per condition. Initial states are shared across methods, delays, and the synchronous VLA reference. We evaluate $d \in \{5, \dots, 10\}$ on RoboMimic and $d \in \{5, 10, 15, 20\}$ on LIBERO.

Training configuration. Tables 1 and 2 report the optimization settings and training seeds for the checkpoints used in the closed-loop experiments. Each component trained in this work uses a single training seed.

Table 1: **Optimization settings.** Shared settings for base-VLA fine-tuning, TT-RTC, RFD, and resolver training.

Hyperparameter	Base VLA / TT-RTC / RFD	Resolver
Optimizer	AdamW	AdamW
Adam (β_1, β_2)	$(0.9, 0.95)$	$(0.9, 0.95)$
Weight decay	10^{-10}	10^{-4}
Gradient clipping norm	10	5
Learning-rate schedule	Cosine with warm-up	Cosine with warm-up
Warm-up steps	1,000	500
Terminal learning rate	2.5×10^{-6}	5% of peak

Table 2: **Training configurations.** Policy update counts extend through the evaluated checkpoint; resolver counts give the full training budget. Learning rates are configured peaks. Arrows and sums denote successive stages.

Component	Task	Peak LR	Batch	Updates	Seed
Base VLA	Square	10^{-4}	16	60,000	1000
Base VLA	Transport	10^{-4}	8	60,000	1000
Base VLA	ToolHang	10^{-4}	16	60,000	1000
Base VLA	LIBERO	—	—	Released checkpoint	—
TT-RTC	Square	10^{-4}	16	100,000	1000
TT-RTC	Transport	2.5×10^{-5}	16	25,000	1000
TT-RTC	ToolHang	$10^{-4} \to 2.5 \times 10^{-5}$	$16 \to 64$	$10{,}000 + 25{,}000$	1000
TT-RTC	LIBERO	2.5×10^{-5}	16	25,000	20260921
RFD	Square	10^{-4}	16	100,000	1000
RFD	Transport	$10^{-4} \to 2.5 \times 10^{-5}$	$8 \to 16$	$50{,}000 + 16{,}000$	1000
RFD	ToolHang	$10^{-4} \to 2.5 \times 10^{-5}$	$16 \to 64$	$50{,}000 + 25{,}000$	1000
RFD	LIBERO	2.5×10^{-5}	16	25,000	20260921
Resolver	Square	3×10^{-4}	16	240,000	20260825
Resolver	Transport	3×10^{-4}	16	240,000	20260825
Resolver	ToolHang	3×10^{-4}	16	240,000	20260903
Resolver	LIBERO	10^{-4}	256	10,000	20260921

Training stages. TT-RTC and RFD update counts exclude preceding base-VLA training. ToolHang TT-RTC and RFD resume the 50k base checkpoint with optimizer state before their final fresh-optimizer stage; Transport RFD likewise uses a fresh optimizer for its final stage. The ToolHang base VLA and initial continuation stages retain the original 100k-step schedule; Square RFD reaches its terminal learning rate at 50k updates and retains it thereafter. TT-RTC uses the final checkpoint on all tasks; RFD uses the final checkpoint except on Square, where the 100k checkpoint is selected from an offline precision/recall sweep extending to 200k updates.

Baseline inference. RTC uses ten Euler steps, the EXP prefix-weight schedule, a maximum guidance weight of 10, and guidance through the full predicted-endpoint Jacobian. Its guidance horizon is $H - s$: 40 actions on RoboMimic and 25 on LIBERO. PAINT uses one repainting cycle: a ten-step generation pass, a ten-step inversion of the generated chunk with its first d actions replaced by the committed prefix, and a ten-step regeneration pass. The regeneration noise combines the inverted prefix noise with the original suffix noise. Thus, PAINT uses 30 flow evaluations per proposal at positive delay. All three passes use the observation available when asynchronous inference begins.

E.2 Recursive Flow-Field Distillation

For each logged observation, we precompute 8 continuations from the frozen VLA. During RFD training, one teacher continuation is sampled, combined with fresh Gaussian noise at a sampled flow time, and used to query the frozen VLA's velocity field at the reached observation o^+. The student receives the earlier observation o^- together with the committed prefix P, and is trained with the delay-aligned velocity-matching objective from Section 4.1.

E.3 Propose–Resolve Implementation

The resolver is trained from frozen-VLA action samples and frozen action-expert features. For each training observation o^+, we draw $M = 32$ VLA continuations and compute their empirical per-coordinate mean and standard deviation over the resolved action segment. A lightweight MLP predicts the corresponding mean and log-standard-deviation from frozen VLA features, using robust Smooth-L1 regression. At inference, features are computed once from the newly observed o^+, and the resulting diagonal Gaussian scores all K proposed continuations by their log density. PR executes the highest-scoring candidate. The VLA remains frozen throughout resolver training, and the same resolver can be applied to proposals from RFD or any of the asynchronous baselines.

Architecture and checkpoint selection. The resolver is an MLP with hidden widths 754, 512, and 256. Each hidden linear layer is followed by LayerNorm and GELU; dropout is zero. Separate linear heads predict the mean and log-standard-deviation over the executed action segment. We select resolver checkpoints by minimum validation negative log-likelihood per action coordinate, using episode-balanced averaging on ToolHang. Validation is performed every 5,000 updates on RoboMimic and every 500 updates on LIBERO. The selected checkpoints are at 225,000, 140,000, and 70,000 updates for Square, Transport, and ToolHang, respectively, and 4,500 updates for LIBERO.

E.4 Timing Attribution

Propose–Resolve separates computation into two stages with different timing requirements. Proposal generation occurs while the committed prefix is being executed and therefore has a budget of d control steps. Resolution begins only once the handoff observation o^+ becomes available and lies directly on the control-critical path.

Table 3 reports the measurements across candidate counts. For SmolVLA, preparing $K = 32$ alternatives takes 90.8 ms on average (91.4 ms at p99), while resolution takes 12.0 ms on average (12.3 ms at p99). Thus, resolution remains within a 50 ms control period at p99, while the more expensive proposal generation occurs before handoff within the asynchronous execution window.

Measurements use one NVIDIA GeForce RTX 4090, with RFD proposals, energy-argmax resolution, 50-action chunks, and ten flow steps. SmolVLA uses two 512×512 camera images with BF16 execution. Generation is measured at $d = 5$ and resolution at $d = 10$. Proposal and resolution

Table 3: **Compute latency of Propose–Resolve.** RFD proposals with energy-argmax resolution on an NVIDIA GeForce RTX 4090, using 50-action chunks and ten flow steps. All times are in milliseconds. Proposal generation overlaps with execution; resolution follows observation of o^+. At $K = 1$, resolution is bypassed. Total means sum separately measured proposal and resolution means; they are not joint pipeline measurements.

Model	K	Propose mean / p99	Resolve mean / p99	Total mean
SmolVLA	1	27.0 / 27.6	0 / 0	**27.0**
SmolVLA	8	39.8 / 40.4	12.0 / 12.4	**51.8**
SmolVLA	16	56.5 / 57.1	12.0 / 12.5	**68.6**
SmolVLA	32	90.8 / 91.4	12.0 / 12.3	**102.8**

are benchmarked separately and include preprocessing, CPU–GPU transfers, model computation, and CPU outputs; camera acquisition and robot I/O are excluded. Proposal generation shares the encoded observation context across candidates, while resolution computes the fresh-observation features once and scores the entire candidate bank jointly.