WHY PLAN FROM SCRATCH? ACCELERATING DIFFUSION VLAS VIA TRAJECTORY RECYCLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion-based Vision-Language-Action (dVLA) models are a powerful paradigm for continuous robotic control, but their real-world deployment is bottlenecked by slow inference driven by iterative diffusion processes and massive multimodal backbones. Standard chunked execution discards the unexecuted tail of every chunk at each re-plan. This squanders prior computational effort and repeatedly invokes the heavy dVLA backbone, bounding achievable latency reduction. We propose WPFS (Why Plan From Scratch?), a framework that accelerates dVLAs by repurposing discarded action chunks as structural priors. WPFS introduces a specialized Corrector that refines these priors into valid trajectories using only a fraction of the original denoising steps. We instantiate two variants: WPFS-Large, which leverages the base model for refinement, and WPFS-Small, which utilizes a distilled model to bypass the massive language backbone entirely. To ensure reliability, an uncertainty-aware self-verification mechanism quantifies decision disagreement in parallel, triggering a full-scale fallback when the drafts disagree, with a single threshold calibrated from the policy's own plan-to-plan spread. Evaluated across state-of-the-art dVLAs on simulation benchmarks (LIBERO, SimplerEnv) and a physical robot, WPFS achieves extreme latency reductions while maintaining or improving task success rates. Code is available at `https://anonymous.4open.science/r/WPFS-FFDD`.

1 INTRODUCTION

Diffusion-based Vision-Language-Action (dVLA) models (Black et al., 2024; Intelligence et al., 2025; 2026; Li et al., 2024a; Team et al., 2024) have emerged as a powerful paradigm for embodied intelligence. Unlike traditional VLAs restricted to coarse action discretization or unimodal regression, dVLAs formulate continuous robotic control as an iterative denoising process. This formulation explicitly captures highly complex, multimodal action distributions without suffering from mode averaging. However, transitioning dVLAs to physical systems is severely bottlenecked by their inherently slow inference speeds, stemming primarily from the iterative nature of diffusion sampling and the computational overhead of multimodal backbones (Chi et al., 2023). Because robust physical control requires high-frequency, low-latency action generation to adapt to dynamic environments (Zhao et al., 2023), mitigating this computational burden without sacrificing task proficiency is a pressing research imperative. Addressing this bottleneck is paramount for enabling responsive, real-time closed-loop deployment while reducing the operational costs of embodied foundation models.

Prior efforts to accelerate dVLA inference generally fall into two orthogonal paradigms: reducing planning frequency and reducing per-step computational cost. The first paradigm predominantly relies on action chunking (Zhao et al., 2023; Xu et al., 2026), amortizing heavy planning costs by predicting multi-step trajectories but executing only a short prefix to maintain reactivity. The second paradigm streamlines step generation via techniques like token pruning (Yang et al., 2025; Wang et al., 2026), cross-frame caching (Xu et al., 2025), or asynchronous execution (Black et al., 2025). While these paradigms can integrate synergistically, inherent inefficiencies severely bound their practical limits. Action chunking in dynamic scenes necessitates frequent receding-horizon replanning, where the unexecuted tail of an active chunk is routinely discarded. This naive truncation is computationally wasteful and frequently introduces temporal distributional shifts, resulting in jerky, non-smooth robotic motions. Conversely, existing intra-cycle cost-reduction strategies still necessitate executing the majority of the massive dVLA backbone during each cycle. This depen-

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.

dency inherently bounds the maximum achievable latency reduction, highlighting the need for a framework that bypasses heavy backbone computation while preserving continuous control.

This limitation raises a fundamental question: *can the unexecuted segments of an action chunk be recycled to synthesize new, valid trajectories without re-invoking the massive dVLA backbone?* In dynamic manipulation, a generated chunk typically becomes kinematically misaligned due to environmental shifts or compounding execution errors. When forced to replan, standard chunked execution discards these tail-end actions. However, because dVLAs are continuously generative, these stale segments still encode rich kinematic priors and critical structural consistency with previously executed actions. Rather than naive truncation, leveraging these discarded segments as high-quality initial conditions for a localized refinement process can realign the trajectory with the updated observation at a fraction of the original planning cost. Crucially, isolating this local refinement into a compact auxiliary model allows the control loop to bypass repeated invocations of the massive native backbone, simultaneously slashing per-step latency and the frequency of full-scale planning.

Motivated by this insight, we introduce **WPFS**, a novel framework that fundamentally accelerates dVLA inference and elevates execution success on dynamic tasks. WPFS repurposes discarded, out-of-date action chunks as informative structural priors. By injecting targeted noise and utilizing a specialized *Corrector*, WPFS refines these priors into valid trajectories via a few truncated denoising steps conditioned on the latest observation, thereby bypassing the full-scale diffusion solver entirely. To ensure execution reliability, we introduce an uncertainty-aware self-verification mechanism: the *Corrector* generates multiple batched drafts and quantifies spatial dispersion. High disagreement indicates that the current observation leaves the next actions underdetermined, and triggers a fallback to the base dVLA for a full replan. This mechanism preserves ultra-fast inference during nominal execution while replacing the fixed execution horizon of standard chunking with a single threshold calibrated from the policy's own plan spread. Finally, we instantiate our framework in two variants: **WPFS-Large**, which uses the frozen base policy as the *Corrector*, and **WPFS-Small**, a lightweight 28M-parameter distilled student dedicated exclusively to refinement, completely decoupling high-frequency control from the massive VLA.

In our experiments, we evaluate WPFS across three diffusion VLAs (π_0, $\pi_{0.5}$, and DB-OFT) on two simulated benchmarks and a physical ROSMASTER X3 Plus manipulator. Across all simulated environments, WPFS consistently shatters the latency ceilings of prior intra-cycle accelerators, achieving speedups of up to **3.91**$\times$ while simultaneously maintaining or improving baseline success rates. Moreover, on highly dynamic real-world interception tasks where standard policies catastrophically fail, our framework elevates the average success rate from 34.7% to **74.7%** at a latency of just 4.72 ms per action, unlocking robust, real-time closed-loop control.

2 PRELIMINARIES

Diffusion-based VLA Policies. A dVLA policy π_θ couples a vision-language model (VLM) backbone with a continuous-time Action Expert. Specifically, a multimodal observation o_t is processed into a perceptual KV cache, which conditions the Action Expert to synthesize an H-step action chunk $A_t = [a_t, \dots, a_{t+H-1}]$. While disparate architectures differ in their training objectives, both flow-matching (e.g., π_0) and standard diffusion (e.g., DB-OFT, Xie et al., 2025) frameworks share a fundamental continuous-time structure. They corrupt a clean chunk A via a forward Gaussian process $x_\tau = a_\tau A + b_\tau \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, I)$ and $A \sim p_\theta(\cdot \mid o_t)$. A neural network is trained to invert this mapping to estimate the clean chunk, effectively acting as a denoiser $D_\theta(x, \tau, o_t)$. Standard planning initializes this generative process at pure noise ($\tau = 1$) and integrates the conditional ODE/SDE over N sequential network evaluations to yield the final action chunk, enabling dVLAs to model complex, multimodal action distributions.

System Latency and Amdahl's Ceilings. A full dVLA inference cycle generally executes a compute-heavy *perception* stage (image encoding, VLM prefill) followed by a memory-bound, multi-step *solver* (action denoising). Using π_0 as a representative case study (Figure 1a), a 57.8 ms cycle (RTX 5090) overruns the 50 ms budget of a 20 Hz controller, split between perception (61%) and the solver (38%). This dichotomy imposes strict Amdahl ceilings across architectures (Figure 1b): for π_0, completely eliminating perception caps theoretical speedups at $2.56\times$, while a zero-cost solver caps at $1.61\times$. Escaping these limits requires attacking both stages simultaneously—which WPFS achieves by bypassing inference cycles entirely.

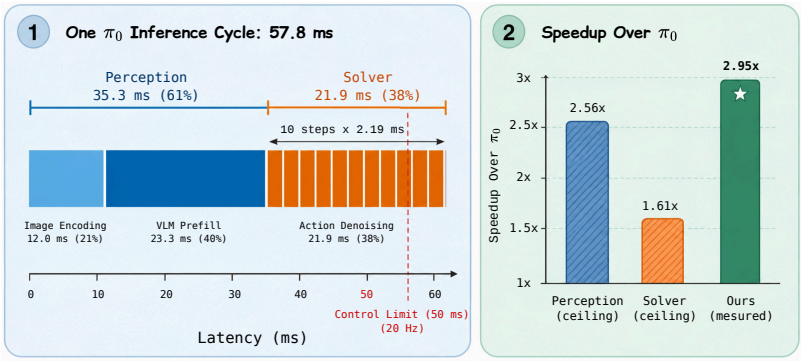


Figure 1: **Latency dichotomy and Amdahl ceilings (π_0 case study). (a)** Inference splits into compute-bound perception and memory-bound solver stages. **(b)** Amdahl ceilings: optimizing only one stage heavily restricts maximum acceleration (hatched). Our framework circumvents this by skipping whole cycles, reaching a $2.95\times$ speedup (solid).

Action Chunking and the Truncation Dilemma. To circumvent the prohibitive latency of the full diffusion solver, state-of-the-art dVLAs (Black et al., 2024; Intelligence et al., 2025; Li et al., 2024a) universally rely on action chunking. Instead of invoking the policy at every environmental step, the model predicts a long-horizon trajectory of H actions but executes only a shorter prefix of $h < H$ steps. While this strategy amortizes the massive inference cost over h steps, it introduces two critical vulnerabilities. First, selecting the execution horizon h forces a brittle trade-off: a large h leads to open-loop execution that fails to react to dynamic environmental shifts, whereas a small h necessitates frequent re-planning, defeating the purpose of amortization. Second, chunking merely delays the computational bottleneck rather than resolving it. Whenever re-planning is triggered, the control loop suffers a catastrophic latency spike. Furthermore, the unexecuted chunk tail is naively discarded, squandering prior computational effort.

3 Proposed Method

3.1 Overview of the WPFS framework

Instead of discarding unexecuted action chunks when the environment shifts, WPFS reclaims them as strong structural priors. By conditioning this stale tail on the *current* observation, the framework bypasses the massive multi-step solver, requiring only a drastically truncated schedule to realign the trajectory. As illustrated in Figure 2, WPFS comprises a **recycling operator** (Section 3.2) that maps the unexecuted tail into a valid initial condition at an intermediate state τ_0; an uncertainty-aware **self-verification certificate** (Section 3.3) that bounds the execution horizon ℓ_t by the spatial variance across K independently noised corrections; and a **Corrector** driving the refinement, instantiated either as the frozen policy π_θ (**WPFS-Large**) or as a 28M-parameter distilled network f_ϕ (**WPFS-Small**, Section 3.4).

3.2 Trajectory Recycling: Warm-Starting the Generative Process

To generalize our recycling mechanism across disparate model families (e.g., DB-OFT and π_0), we abstract the diverse corruption schedules (a_τ, b_τ) using a single *noise-to-signal ratio*:

$$r_\tau = \frac{b_\tau}{a_\tau}, \tag{1}$$

which equates to $\tau/(1-\tau)$ for flow policies and $\sqrt{(1-\bar{\alpha}_t)/\bar{\alpha}_t}$ for diffusion policies. By formulating the warm-start mechanism entirely in terms of r_τ, our framework transfers seamlessly between standard diffusion and flow-matching models without requiring architecture-specific translations.

Constructing the structural prior. Let $A \in \mathbb{R}^{H\times d_a}$ denote the active action chunk, from which s steps have already been executed. To construct a valid prior for the updated observation, the unexe-

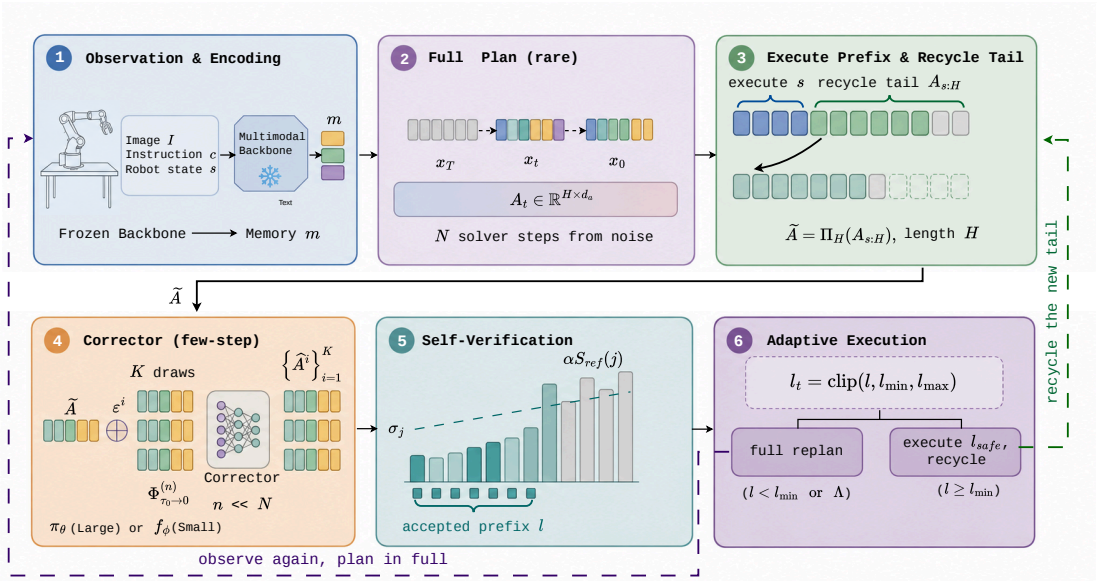


Figure 2: **The WPFS framework.** The stale tail (Block 3) is recycled into a strong prior $\tilde{A}$. A few-step Corrector (Block 4) refines it into K batched drafts. A Self-Verification module (Block 5) measures spatial disagreement σ_j across drafts to dynamically bound the accepted execution prefix ℓ_t (Block 6)—bypassing the computationally heavy Full Plan (Block 2) without sacrificing success.

cuted tail is temporally shifted such that index 0 aligns with the current timestep, and subsequently padded to the original horizon H by replicating its final action:

$$\tilde{A} = \Pi_H(A_{s:H}), \qquad [\Pi_H(B)]_j = \begin{cases} B_j, & j < |B|, \\ B_{|B|-1}, & \text{otherwise.} \end{cases} \tag{2}$$

In the WPFS framework, we strictly use this zero-order hold padding rather than zero-padding for one reason: in robotic manipulation, a held pose represents a stable, reachable spatial command. Conversely, zero-padding in a normalized action space inherently commands a sudden jump to the dataset mean, inducing catastrophic jerk.

The correction operator. We re-enter the generative path at an intermediate state τ_0 using $\tilde{A}$ in place of a clean sample, and run the policy's own solver from τ_0 to 0 for $n \ll N$ steps. To support self-verification (Section 3.3), a single correction step simultaneously generates K candidate *drafts* by batching the evaluation with independent noise:

$$x^i = a_{\tau_0}\tilde{A} + b_{\tau_0}\varepsilon^i, \qquad \hat{A}^i = \Phi^{(n)}_{\tau_0 \to 0}(x^i; o_t), \qquad i = 1, \ldots, K, \tag{3}$$

where $\Phi^{(n)}$ denotes n solver steps driven by the corrector $\psi \in \{\theta, \phi\}$ and conditioned on the *current* observation o_t. The K candidate drafts share o_t, thus sharing one image encoding and VLM prefill; only the lightweight action pathway widens to batch K, adding a negligible 4% latency overhead.

To choose (τ_0, n) from theory rather than by tuning, assume an **optimal denoiser**—where $D_\theta(x, \tau_0, o) = \mathbb{E}[A \mid x_{\tau_0} = x, o]$ follows directly from Tweedie's formula (Efron, 2011)—under which trajectory recycling is a *localized search*:

Proposition 1 (Correction as Localized Search). *If $x = a_{\tau_0}\tilde{A} + b_{\tau_0}\varepsilon$ is the warm start generated via Eq. (3), a one-step denoiser evaluation yields the localized posterior mean:*

$$D_\theta(x, \tau_0, o) = \frac{\int A\, p_\theta(A \mid o)\, \kappa(A)\, \mathrm{d}A}{\int p_\theta(A \mid o)\, \kappa(A)\, \mathrm{d}A}, \quad \textit{where} \quad \kappa(A) = \exp\left(-\frac{\|A - \tilde{A} - r_{\tau_0}\varepsilon\|^2}{2\, r_{\tau_0}^2}\right). \tag{4}$$

Equation (4) reveals a powerful premise: *the stale draft biases the search toward a behavioral mode, while the fresh observation refines the trajectory within it.* The one-step correction is the mean of the *current* plan distribution $p_\theta(\cdot \mid o_t)$ reweighted by an isotropic kernel of bandwidth r_{τ_0} centered near the stale prior. Consequently, r_{τ_0} dictates the search radius, while n controls the sampling behavior.

Crucially, as $r_{\tau_0} \to 0$, the kernel concentrates at $\tilde{A}$, making the correction a no-op. Conversely, as $r_{\tau_0} \to \infty$ with $n = 1$, it collapses to the unconditioned posterior mean $\mathbb{E}[A \mid o_t]$, which in multimodal distributions often falls into out-of-distribution whitespace. For $n > 1$ the solver no longer returns this single mean, which in practice mitigates mode averaging, although Proposition 1 does not establish that the multi-step output samples the localized distribution. This establishes a principled design space. The bandwidth r_{τ_0} must stay below the geometric separation of competing modes, while n prevents mode-averaging at higher noise.

3.3 Self-Verification: A Self-Calibrated Agreement Certificate

Since a single correction step is computationally inexpensive, the critical system-level decision is determining its *execution horizon*—how many steps of the generated trajectory to trust. While standard action chunking relies on a brittle, static hyperparameter h (Section 2), WPFS determines this horizon dynamically per decision, utilizing a free diagnostic statistic naturally produced by the K batched drafts. Specifically, the K drafts from Eq. (3) share the same observation, stale prior, and noise level τ_0, differing only in the injected noise ε, their spatial dispersion is not random error. Rather, it reflects the variability of the policy's own conditional action distribution, resolved along the chunk horizon.

Lemma 2 (Draw Dispersion Tracks Local Posterior Covariance)**.** *Assuming the optimal denoiser behavior established in Section 3.2, for n=1 and $\hat{A}(\varepsilon)$ generated as in Eq. (3), the sensitivity of the correction to the noise draw strictly follows:*

$$\frac{\partial \hat{A}}{\partial \varepsilon} \;=\; \frac{1}{r_{\tau_0}} \operatorname{Cov}[A \mid x_{\tau_0},\, o_t]\,. \tag{5}$$

To first order, the drafts scatter proportionally to the local posterior covariance. *Where the current observation strongly determines the action, the drafts collapse onto a single mode; where it leaves a genuine choice, they diverge.* This provides a free, localized identifiability signal. The certificate thus certifies agreement, not correctness—a scheduling signal rather than a safety guarantee: drafts from one model can agree on the same wrong action, and such a shared error passes the test; Appendix B.2 measures what the signal is worth in practice.

The verification test. Partition the action channels into continuous ones $\mathcal{P}$ and discrete ones $\mathcal{D}$, whose values concentrate on finitely many modes. At each position j we measure the spatial disagreement over $\mathcal{P}$:

$$\sigma_j \;=\; \operatorname*{med}_{1 \le i < i' \le K} \big\| \hat{A}^i[j]_{\mathcal{P}} - \hat{A}^{i'}[j]_{\mathcal{P}} \big\|_2. \tag{6}$$

We accept position j only when this dispersion is bounded *and* every discrete channel is unanimous after projection onto its nearest mode (q_d; Appendix A):

$$\mathrm{ok}_j \;=\; \Big(\sigma_j \le \alpha\, S_{\mathrm{ref}}(j) \Big) \;\wedge\; \bigwedge_{d \in \mathcal{D}} \Big(q_d\big(\hat{A}^1[j]_d\big) = \cdots = q_d\big(\hat{A}^K[j]_d\big) \Big). \tag{7}$$

The certificate outputs the longest accepted *prefix*, $\ell = \max\{n' : \mathrm{ok}_j \text{ for all } j < n'\}$: a prefix rather than a raw count, because actions execute sequentially and any command following an uncertain state inherits that uncertainty.

Self-calibration via policy baselines. The bounding scale $S_{\mathrm{ref}}(j)$ is the cornerstone of the certificate, enabling a single mathematical test to carry across architectures with one dimensionless multiplier rather than a dataset-specific absolute threshold ($\alpha = 1$ for the flow policies, 1.5 for DB-OFT). Rather than employing an absolute spatial threshold, we utilize *the policy's own plan-to-plan spread.* $S_{\mathrm{ref}}(j)$ is pre-computed by evaluating Eq. (6) on $M = 8$ independent full plans drawn from common active starting states. Because baseline plans naturally agree on immediate actions but drift at longer horizons, S_{ref} intrinsically scales with j. Equation (7) thus enforces a principled operating rule: *execute position j only if the correction is at least as confident as a fresh plan would be.* This

isolates a single dimensionless multiplier $\alpha \approx 1$. To ensure threshold stability, S_{ref} is frozen per task and exclusively calibrated on initial active states to avoid degenerate variance collapse when the robot is idle.

From certificate to control schedule. The accepted prefix is clipped to $\ell_t = \text{clip}(\ell, \ell_{\min}, \ell_{\max})$ for execution. If $\ell < \ell_{\min}$, the certificate *rejects* the correction, immediately triggering a full planning cycle. Three bounding guards complete the schedule: (i) a full plan always executes a fixed minimum prefix ℓ_{safe}; (ii) A continuous lineage (a plan and its subsequent corrections) is strictly cut after Λ environment steps to bound perceptual drift; (iii) $\ell_{\max}$ caps a single correction to prevent arbitrary open-loop execution. Consequently, the static execution horizon h is elevated into a *dynamic random variable* ℓ_t dictated strictly by the policy's own real-time confidence.

3.4 WPFS-Small: Distilling the Corrector

Although WPFS-Large minimizes full planning cycles, running every correction through the massive policy backbone remains computationally heavy, severely capping overall acceleration. Unlocking an order-of-magnitude speedup requires aggressively distilling this Corrector. Fortunately, unlike generating trajectories from pure noise, *correcting* an existing structural prior is a strictly localized problem. This inherently simpler task requires far less representational capacity, allowing a lightweight student model to perform the refinement equally well at a fraction of the cost.

Denoiser Matching Objective. Because the teacher's raw output is parameterization-dependent, we directly regress the estimated clean chunk—the denoiser D_θ—in action space:

$$\mathcal{L}_D \;=\; \mathbb{E}\Big[\, \big\| \, D_\phi(x,\tau,m) - D_\theta(x,\tau,o) \, \big\|_W^2 \Big], \tag{8}$$

with W re-weighting channels and positions, and m the student's memory (detailed below). The loss is taken in the clean action space on purpose: since $D = (x - b_\tau \hat{\epsilon})/a_\tau$ in the noise parameterization and $D = x - \tau\hat{v}$ in the velocity one, Eq. (8) equals $r_\tau^2 \|\hat{\epsilon}_\phi - \hat{\epsilon}_\theta\|^2$ and $b_\tau^2 \|\hat{v}_\phi - \hat{v}_\theta\|^2$ respectively, whereas an unweighted loss in either prediction space would spend capacity on the nearly noiseless final steps. Formulating the loss via D_θ effectively unifies the objective across all model families.

Dispersion-Aware Distillation. A fundamental vulnerability arises when integrating the student with the verification certificate (Section 3.3). The certificate relies on the spatial dispersion σ_j across K draws. Because $\mathcal{L}_D$ is a first-moment objective, it does not constrain this variance. A smooth student could achieve a competitive $\mathcal{L}_D$ while systematically *under-dispersing*, causing Eq. (7) to accept prefixes the teacher would reject ($\ell \to \ell_{\max}$), silently disabling the fallback. We resolve this by explicitly regularizing the student's dispersion to match the teacher's:

$$\mathcal{L} \;=\; \mathcal{L}_D \;+\; \lambda \sum_{j<\ell_{\max}} \big(\bar{\sigma}_j(\phi) - \bar{\sigma}_j(\theta)\big)^2, \tag{9}$$

where we substitute the median in Eq. (6) with the mean $\bar{\sigma}_j$ to maintain differentiability. The broader implication is clear: *when a control loop relies on a diagnostic statistic of a model's output, that statistic must be explicitly preserved in the distillation objective.*

Student Architecture. We emphasize a critical architectural bypass: *no layer of the language model backbone runs inside a correction.* Instead, a lightweight 28M-parameter cross-attention transformer (detailed in Appendix C) serves all three policies and is trained via DAgger (Ross et al., 2011). At every correction the student re-encodes the *current* observation with the policy's own frozen vision encoder—most of a correction's cost—and cross-attends to those visual tokens and the instruction embedding; from the preceding full plan it reuses only the *plan context* z. We also inject a sinusoidal *plan age* u to help the student differentiate active plans from padded tails. Consequently, the heavier the dVLA backbone, the higher the computational dividend of this distilled corrector.

4 Experiment Results

This section empirically evaluates our framework to answer two core questions: **(Q1)** Does trajectory recycling maintain task success while eliminating full planning cycles, and how does it compare against state-of-the-art accelerators (Sec. 4.2)? **(Q2)** Do these efficiency gains translate to physical hardware, where dynamic scenes and strict control frequencies compound the latency bottleneck

(Sec. 4.3)? Additionally, we validate the fundamental premise of WPFS—that unexecuted chunks suffer from kinematic rather than semantic failure—in Appendix A.4.

4.1 Experimental Settings

Benchmarks and Metrics. We evaluate WPFS across simulated and physical domains. In simulation, we benchmark on LIBERO (Liu et al., 2023) (4 suites, 10 tasks/suite, 500 episodes/suite) and the WidowX (Bridge) split of SimplerEnv (Li et al., 2024b) (24 layouts, 120 episodes/task). For physical validation, we deploy on a ROSMASTER X3 Plus manipulator (Sec. 4.3). The primary metrics are task success rate and normalized inference speedup. To isolate algorithmic efficiency from hardware-level contention, speedups are computed using deterministic model-call counts multiplied by profiled idle-GPU latencies rather than wall-clock time. For real-world deployment, we report the GPU latency per *executed action* to directly assess control frequency compliance.

Backbones. As a model-agnostic wrapper, WPFS is instantiated on three diverse dVLA architectures spanning different generative frameworks, chunk geometries, and backbone scales: π_0 (Black et al., 2024) (50-step chunk via a narrow action expert), $\pi_{0.5}$ (Intelligence et al., 2025) (10-step chunk directly in action space), and DB-OFT (Xie et al., 2025) (16-step DDIM chunk driven by a 7B LLM backbone). Baseline evaluations run the unmodified policies at their default execution schedules ($10/50$, $5/10$, and $5/16$ executed actions per chunk, respectively).

Baselines. We compare against four recent accelerators that tackle the latency bottleneck from two orthogonal directions (Section 2). Three methods reduce the cost *per planning cycle*: *SpecPrune-VLA* (Wang et al., 2026) via dual-granularity visual token pruning, *VLA-Cache* (Xu et al., 2025) via KV-cache reuse for static tokens, and *EfficientVLA* (Yang et al., 2025) via combined layer pruning and token selection. The fourth method, *AAC* (Liang et al., 2026), targets planning *frequency* by dynamically adapting the executed chunk length, serving as the closest conceptual baseline to our self-verification module. All baselines share the identical model checkpoints and evaluation harness.

Implementation Details. Our core components (the recycling operator, verification certificate, and 28M student architecture) remain identical across all backbones; only the noise schedules are adapted to match each policy's geometry. Specifically, π_0 and $\pi_{0.5}$ re-enter at $\tau_0 = 0.5$ ($r_{\tau_0} = 1$) using $n = 1$ solver step, while DB-OFT re-enters at $\bar{\alpha} = 0.967$ ($r_{\tau_0} = 0.19$) using $n = 2$ DDIM steps. All setups generate $K = 4$ candidate drafts. The verification multiplier is $\alpha = 1.0$ for the flow policies and $\alpha = 1.5$ for DB-OFT. Horizon bounds, lineage limits, and training recipes are detailed in Appendix C. All simulations run within a unified harness on a single RTX 5090.

4.2 Q1: Recycling on LIBERO and SimplerEnv

Main Results on LIBERO. For π_0, WPFS-Large improves the baseline average success rate from 94.1% to 94.9% at a $1.52\times$ speedup, while WPFS-Small achieves 95.0% at a $2.95\times$ speedup. These gains are particularly pronounced in the Long suite ($86.2\% \rightarrow 89.8\%$), where chunks are highly susceptible to kinematic aging prior to replanning. In contrast, existing accelerators trade task success for speed: AAC yields a $1.70\times$ speedup but drops performance by 4.1 points, VLA-Cache gains $1.21\times$ for a 3.0-point drop, and SpecPrune-VLA gains $1.19\times$ for a 1.5-point drop. On $\pi_{0.5}$, where baseline success is saturated at 97.5%, all methods remain within a 1-point margin; however, cycle-level baselines peak at a $1.38\times$ speedup, whereas WPFS-Small reaches $3.91\times$ at 97.8% success.

Main Results on SimplerEnv. WPFS demonstrates the largest margins on the computationally constrained DB-OFT backbone. WPFS-Large elevates success from 67.9% to 74.6% ($+6.7$) and WPFS-Small achieves 74.0% at a $2.05\times$ speedup, significantly outperforming the strongest baseline (EfficientVLA, 70.9%) and AAC (62.1%). This improvement is largely driven by the PutCarrot task ($60.0\% \rightarrow 82.5\%$), where the default execution schedule ($5/16$ actions) fails to adapt to dynamic scene complexities. Our certificate dynamically extends the horizon for highly-agreed chunks and instantly truncates misaligned ones—an adaptive capability absent in fixed schedules. Conversely, on StackCube every method except AAC—WPFS included—lands within one episode of the 31.7% baseline. This indicates that failure modes on this task stem from core policy limitations rather than inference scheduling, making it orthogonal to acceleration techniques.

Efficiency Analysis. Because pruning and caching operate *inside* a planning cycle, baselines strictly inherit Amdahl ceilings (Section 2): regardless of compression, no baseline exceeds $1.38\times$ on $\pi_{0.5}$

Table 1: Performance Evaluation on LIBERO and SimplerEnv. **Bold** is best, underline is second-best. Proposed methods are highlighted in blue.

		Task Success Rate (%)				Avg.	Avg.
LIBERO Suites →		Spatial	Object	Goal	Long	**Spd (×)**	**SR (%)**
π_0	Baseline	**96.0**	98.4	**95.6**	86.2	1.00	94.1
	SpecPrune-VLA [ICML26]	95.6	98.4	93.4	83.0	1.19	92.6
	VLA-Cache [NIPS25]	94.0	98.8	93.0	78.6	1.21	91.1
	EfficientVLA [NIPS25]	95.6	99.0	**95.6**	85.6	1.34	94.0
	AAC [CVPR26]	90.2	96.6	92.8	80.4	1.70	90.0
	WPFS-Large (Ours)	95.6	**99.4**	94.6	**89.8**	1.52	94.9
	WPFS-Small (Ours)	95.6	99.0	**95.6**	**89.8**	**2.95**	**95.0**
$\pi_{0.5}$	Baseline	98.6	98.8	98.4	94.0	1.00	97.5
	SpecPrune-VLA [ICML26]	98.0	98.4	98.4	93.6	1.25	97.1
	VLA-Cache [NIPS25]	97.6	98.2	98.0	93.2	1.34	96.8
	EfficientVLA [NIPS25]	**99.0**	98.8	98.0	92.6	1.38	97.1
	AAC [CVPR26]	98.0	**99.4**	97.8	93.6	1.35	97.2
	WPFS-Large (Ours)	98.8	99.2	**98.8**	**94.6**	1.83	**97.9**
	WPFS-Small (Ours)	**99.0**	99.0	**98.8**	94.4	**3.91**	97.8
SimplerEnv Tasks →		Carrot	Spoon	Stack	Eggplant	**Spd (×)**	**SR (%)**
DB-OFT	Baseline	60.0	85.8	**31.7**	94.2	1.00	67.9
	SpecPrune-VLA [ICML26]	59.2	85.8	30.8	92.5	1.13	67.1
	VLA-Cache [NIPS25]	57.5	86.7	**31.7**	90.8	1.02	66.7
	EfficientVLA [NIPS25]	67.5	**90.0**	**31.7**	94.2	1.59	70.9
	AAC [CVPR26]	60.8	78.3	25.9	83.3	1.18	62.1
	WPFS-Large (Ours)	**82.5**	88.3	30.8	**96.7**	1.43	**74.6**
	WPFS-Small (Ours)	79.2	**90.0**	**31.7**	95.0	**2.05**	74.0

Table 2: Real-robot evaluation on static and dynamic manipulation tasks (50 trials per task).

	Task Success Rate (%)			Avg.	Lat.	Spd.
Method	Static	5 cm/s	6 cm/s	**SR (%)**	**(ms)**	(×)
$\pi_{0.5}$ (Base)	76.0	28.0	0.0	34.7	18.16	1.00
WPFS-Large (Ours)	**78.0**	82.0	16.0	58.7	9.71	1.87
WPFS-Small (Ours)	76.0	**98.0**	**50.0**	**74.7**	**4.72**	**3.85**

or $1.59\times$ on DB-OFT. By bypassing cycles entirely, WPFS replaces the single-stage latency ceiling with a bound dictated by the ratio of correction latency to full planning latency ($\gamma = L_{\text{corr}}/L_{\text{plan}}$). Distillation compresses this ratio from 0.52–0.69 (WPFS-Large) down to 0.028–0.15 (WPFS-Small), amplifying the 1.4–$1.8\times$ teacher speedup into a 2.0–$3.9\times$ student speedup. WPFS-Small maintains parity with WPFS-Large across all backbones ($95.0/97.8/74.0$ vs. $94.9/97.9/74.6$). This validates our premise in Section 3.4: refining a trajectory is a strictly bounded, localized problem that a lightweight 28M student can solve like the massive native VLA.

4.3 Q2: Real-World Deployment in Dynamic Environments

Setup. We deploy WPFS on a physical ROSMASTER X3 Plus manipulator, governed by a 40 Hz control loop on a single NVIDIA RTX 5060 Ti. We benchmark three tasks (50 trials each): a static Pick Cube task, and two dynamic variants where the target cube translates across the workspace at uniform linear velocities of 5 cm/s and 6 cm/s, requiring real-time interception.

Main Results. The performance dichotomy between static and dynamic regimes highlights the core advantage of WPFS. In the static setting, all configurations achieve comparable success (76.0–78.0%); without environmental shift, stale chunks remain kinematically valid, validating that a corrector provides negligible marginal utility when the scene is perfectly stationary. Under dynamic motion, however, the base policy degrades severely, succeeding in only 28% of trials at 5 cm/s

and collapsing to 0% at 6 cm/s. In stark contrast, WPFS-Small maintains robust interception, achieving 98% and 50% success on the respective dynamic tasks. Averaged across all variants, trajectory recycling elevates the success rate from 34.7% to 74.7% using identical frozen policy weights.

The Imperative of Latency on Hardware. On hardware the ordering of our variants reverses—WPFS-Small outperforms WPFS-Large by 16.0 points—because of control-frequency compliance. In simulation, the environment pauses during inference; on hardware, a 40 Hz loop gives each step a 25 ms period. The base policy averages 18.16 ms per step but concentrates compute into blocking multi-step generations, causing actuation to stall for several periods. WPFS-Large reduces average latency to 9.71 ms but still invokes the heavy backbone during corrections. WPFS-Small averages 4.72 ms, under a fifth of the period; its full-plan fallbacks still exceed the period and stall the loop, but far less often than the baseline's replans. We attribute much of the margin between 16% and 50% on the fastest 6 cm/s task to this difference in how often the loop can revise its command.

5 Related Work

Intra-Cycle Optimization. The dominant paradigm in VLA acceleration compresses components within a single planning cycle. One line targets perception through visual token pruning, KV-caching, or dynamic layer routing (Xu et al., 2025; Wang et al., 2026; Yang et al., 2025; Li et al., 2026; Yue et al., 2024; Zhang et al., 2026). Another targets the generative solver, distilling diffusion policies into one- or few-step samplers (Prasad et al., 2024; Wang et al., 2025b; Yan et al., 2025). While these methods reduce stage costs, they remain subject to the Amdahl ceilings discussed in Section 5: optimizing the solver leaves the VLM bottleneck intact, and vice versa. Moreover, distilling the full planner—mapping pure noise to a complete action distribution—still requires much of the original backbone's capacity. WPFS instead bypasses the full cycle: rather than distilling a planner, we distill a localized *corrector*, showing that refining an existing structural prior requires only 28M parameters and avoids single-stage latency ceilings.

Temporal Reuse and Adaptive Execution. A parallel line exploits temporal redundancy to avoid re-planning. Warm-starting from the previous prediction was suggested for Diffusion Policy (Chi et al., 2023); Streaming Diffusion Policy (Høeg et al., 2025) propagates partially denoised chunks across timesteps but requires a bespoke progressive-noise policy trained from scratch. Temporal ensembling (Zhao et al., 2023) reuses overlapping chunks via heuristic averaging, risking out-of-distribution mode collapse. In contrast, our recycling operator requires no policy retraining and re-enters the native solver. Meanwhile, AAC (Liang et al., 2026) and Xu et al. (2026) adapt the execution horizon to decide *when* to replan; real-time chunking (Black et al., 2025) hides inference latency by generating the next chunk asynchronously, freezing committed actions and inpainting a compatible continuation; and parallel (Kim et al., 2025; Song et al., 2025) and speculative (Wang et al., 2025a) decoding shorten autoregressive generation. However, discrete speculative verification struggles with continuous actions, while adaptive-horizon methods still incur a full re-plan latency spike when triggered. Trajectory reuse itself is not new; WPFS contributes a combination of principled re-noising of the clean tail, an agreement test that determines the execution horizon, and a distilled corrector that removes the language backbone from corrections.

6 Conclusion

In this paper, we propose WPFS, a novel framework that circumvents Amdahl's limits of standard intra-cycle acceleration toward real-time, closed-loop deployment of dVLA models. Our framework improves efficiency by recycling the unexecuted tail of the active action chunk as a warm start, realigning it with a single truncated solver step, and using a self-calibrated certificate to dynamically set the execution horizon. Furthermore, by distilling this localized correction task into a lightweight 28M-parameter student, WPFS removes the heavy VLA backbone from every correction, leaving it only in the full plans the certificate still requests. Extensive experiments demonstrate that WPFS achieves up to a $3.91\times$ inference speedup across π_0, $\pi_{0.5}$, and DB-OFT at success rates comparable to the base policy, and more than doubles the success rate on dynamic real-world interception tasks. The physical evaluation is nevertheless narrow—one platform and constant-velocity interception—and robustness to abrupt changes, occlusion or human intervention remains to be established.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems*, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy, 2023.

Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

Sigmund H. Høeg, Yilun Du, and Olav Egeland. Streaming diffusion policy: Fast policy synthesis with variable noise diffusion models. In *IEEE International Conference on Robotics and Automation*, 2025.

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

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: A steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

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

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024a.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, 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 *Conference on Robot Learning (CoRL)*, 2024b.

Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shutao Xia, Zhi Wang, and Wenwu Zhu. SP-VLA: A joint model scheduling and token pruning approach for VLA model acceleration. In *International Conference on Learning Representations*, 2026.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

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, Datasets and Benchmarks Track*, 2023.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Robotics: Science and Systems*, 2024.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Han Zhao, Wei Zhao, Zhide Zhong, Zongyuan Ge, Zhijun Li, Donglin Wang, Jun Ma, Lujia Wang, and Haoang Li. PD-VLA: Accelerating vision-language-action model integrated with action chunking via parallel decoding. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. SpecPrune-VLA: Accelerating vision-language-action models via action-aware self-speculative pruning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Songsheng Wang, Rucheng Yu, Zhihang Yuan, Chao Yu, Feng Gao, Yu Wang, and Derek F. Wong. Spec-VLA: Speculative decoding for vision-language-action models with relaxed acceptance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2025a.

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. In *International Conference on Machine Learning*, 2025b.

Bin Xie, Erjin Zhou, Fan Jia, Hao Shi, Haoqiang Fan, Haowei Zhang, Hebei Li, Jianjian Sun, Jie Bin, Junwen Huang, et al. Dexbotic: Open-source vision-language-action toolbox. *arXiv preprint arXiv:2510.23511*, 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.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-Cache: Efficient vision-language-action manipulation via adaptive token caching. In *Advances in Neural Information Processing Systems*, 2025.

Ge Yan, Jiyue Zhu, Yuquan Deng, Shiqi Yang, Ri-Zhao Qiu, Xuxin Cheng, Marius Memmel, Ranjay Krishna, Ankit Goyal, Xiaolong Wang, and Dieter Fox. ManiFlow: A general robot manipulation policy via consistency flow training. In *Conference on Robot Learning*, 2025.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. EfficientVLA: Training-free acceleration and compression for vision-language-action models. *arXiv preprint arXiv:2506.10100*, 2025.

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

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Yuan Du, and Shanghang Zhang. MoLe-VLA: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.

A Theoretical Analysis and Proofs

Throughout this appendix the forward process is $x_\tau = a_\tau A + b_\tau \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, I)$ drawn independently of $A \sim p_\theta(\cdot \mid o)$, and $r_\tau = b_\tau / a_\tau > 0$ is the noise-to-signal ratio of Eq. (1). We assume the optimal denoiser $D_\theta(x, \tau, o) = \mathbb{E}[A \mid x_\tau = x, o]$, which is what both standard parameterisations compute by Tweedie's formula (Efron, 2011): in the noise parameterisation

$$\frac{x - b_\tau \, \mathbb{E}[\varepsilon \mid x, o]}{a_\tau} \;=\; \mathbb{E}\left[\frac{x - b_\tau \varepsilon}{a_\tau} \;\middle|\; x, o \right] \;=\; \mathbb{E}[A \mid x, o],$$

and in the velocity parameterisation ($a_\tau = 1 - \tau$, $b_\tau = \tau$), since $x - \tau(\varepsilon - A) = (1 - \tau)A + \tau\varepsilon - \tau\varepsilon + \tau A = A$ pointwise,

$$x - \tau \, \mathbb{E}[\varepsilon - A \mid x, o] \;=\; \mathbb{E}[A \mid x, o].$$

Because the action space is normalised, we take $p_\theta(\cdot \mid o)$ to have bounded support, which makes every integral below finite and licenses dominated convergence.

A.1 Proof of Proposition 1

Proof. By Bayes' rule, the posterior over clean chunks given the iterate is

$$p(A \mid x_{\tau_0}{=}x, o) \;\propto\; p_\theta(A \mid o)\, p(x_{\tau_0}{=}x \mid A), \qquad p(x_{\tau_0}{=}x \mid A) \;\propto\; \exp\!\left(-\frac{\|x - a_{\tau_0}A\|^2}{2b_{\tau_0}^2}\right),$$

the second factor being the density of $\mathcal{N}(a_{\tau_0}A,\, b_{\tau_0}^2 I)$. Substituting the warm start $x = a_{\tau_0}\tilde{A} + b_{\tau_0}\varepsilon$ of Eq. (3),

$$x - a_{\tau_0}A \;=\; a_{\tau_0}(\tilde{A} - A) + b_{\tau_0}\varepsilon \;=\; -a_{\tau_0}\big(A - \tilde{A} - r_{\tau_0}\varepsilon\big),$$

so that, using $b_{\tau_0}^2 = a_{\tau_0}^2 r_{\tau_0}^2$,

$$\frac{\|x - a_{\tau_0}A\|^2}{2b_{\tau_0}^2} \;=\; \frac{a_{\tau_0}^2\,\|A - \tilde{A} - r_{\tau_0}\varepsilon\|^2}{2\,a_{\tau_0}^2 r_{\tau_0}^2} \;=\; \frac{\|A - \tilde{A} - r_{\tau_0}\varepsilon\|^2}{2r_{\tau_0}^2}.$$

The likelihood therefore equals $c(x)\,\kappa(A)$ with κ as in Eq. (4) and $c(x)$ independent of A, and $c(x)$ cancels between numerator and denominator when the posterior is normalised. Taking the expectation of A under this posterior and invoking the denoiser identity above gives Eq. (4). □

The kernel κ is (up to normalisation) the density of $\mathcal{N}(\tilde{A} + r_{\tau_0}\varepsilon,\, r_{\tau_0}^2 I)$: the stale chunk enters only through the *location* of the search, and the current observation enters only through the prior $p_\theta(\cdot \mid o)$ being searched. The bandwidth r_{τ_0} is the single quantity that trades the two off, and its two extremes are degenerate.

A.2 The Two Limits of the Re-entry Bandwidth

Corollary 3 (Degenerate bandwidths)**.** *Let $\hat{A} = D_\theta(x, \tau_0, o)$ be obtained from the warm start of Eq. (3) with $n = 1$, and let $p_\theta(\cdot \mid o)$ be continuous and strictly positive at $\tilde{A}$. Then for every fixed ε:* (i) *$\hat{A} \to \tilde{A}$ as $r_{\tau_0} \to 0$, i.e. the correction becomes a no-op; and* (ii) *$\hat{A} \to \mathbb{E}[A \mid o]$ as $r_{\tau_0} \to \infty$, i.e. the correction becomes the conditional mean of the plan distribution, independent of both $\tilde{A}$ and ε.*

Proof. (i) κ is proportional to the density of $\mathcal{N}(\tilde{A} + r_{\tau_0}\varepsilon,\, r_{\tau_0}^2 I)$, whose centre tends to $\tilde{A}$ and whose width tends to 0. It is therefore an approximate identity at $\tilde{A}$, and since $p_\theta(\cdot \mid o)$ is continuous and positive there, the tilted law $p_\theta(A \mid o)\kappa(A)/Z$ converges weakly to $\delta_{\tilde{A}}$; bounded support gives convergence of the mean.

(ii) Expanding the square,

$$\kappa(A) \;=\; \exp\!\big(-\tfrac{1}{2}\|\varepsilon\|^2\big) \cdot \exp\!\left(\frac{\langle A - \tilde{A},\, \varepsilon\rangle}{r_{\tau_0}} - \frac{\|A - \tilde{A}\|^2}{2r_{\tau_0}^2}\right).$$

The first factor is constant in A and cancels in the ratio of Eq. (4). The second converges to 1 pointwise as $r_{\tau_0} \to \infty$ and is bounded on the support of $p_\theta(\cdot \mid o)$, so by dominated convergence both integrals converge to their unweighted counterparts and $\hat{A} \to \int A\, p_\theta(A \mid o)\, \mathrm{d}A$. □

Corollary 3 is what makes τ_0 a design variable rather than a tuning knob: it must be large enough to leave the stale chunk and small enough not to average the modes of $p_\theta(\cdot \mid o)$ together. Appendix B.4 measures both failures in the control loop, and Appendix A.4 measures them geometrically.

A.3 Proof of Lemma 2

Proof. Expanding the exponent of the likelihood and discarding the term $\|x\|^2$, which does not depend on A,

$$p(A \mid x_{\tau_0}{=}x, o) \;\propto\; \underbrace{p_\theta(A \mid o) \exp\!\left(-\frac{a_{\tau_0}^2\|A\|^2}{2b_{\tau_0}^2}\right)}_{h(A),\ \text{independent of } x} \cdot\; \exp\big(\langle \eta, A\rangle\big), \qquad \eta \;=\; \frac{a_{\tau_0}}{b_{\tau_0}^2}\, x.$$

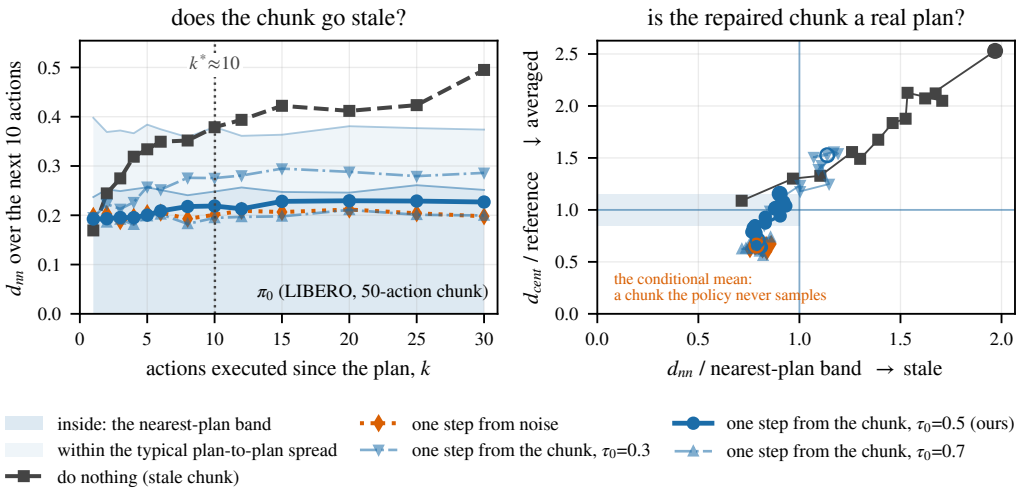


Figure 3: **Offline validation of the trajectory recycling premise.** At a state where the active π_0 chunk was generated k steps ago, we draw 8 fresh full-depth plans conditioned on the current observation. We measure the geometric distance of candidate chunks to this fresh distribution over the next 10 actions: d_{nn} (distance to the nearest fresh plan) and d_{cent} (distance to their centroid), normalized by the reference plans' own internal variance (1,950 probes, 12 LIBERO-Spatial episodes). **Left:** The uncorrected stale chunk drifts outside the nearest-plan band at $k = 3$ and exceeds the typical plan-to-plan spread at $k^* \approx 10$, whereas a one-step correction remains bounded out to $k = 30$. **Right:** Failure modes in the (d_{nn}, d_{cent}) plane. Re-entering at $\tau_0 = 0.7$ (or from pure noise) collapses to the conditional mean ($d_{cent} \approx 0.6$), producing a mode-averaged trajectory the policy would never natively sample. Conversely, $\tau_0 = 0.3$ fails to escape the stale prior. As predicted by Proposition 1, only $\tau_0 = 0.5$ ($r_{\tau_0} = 1$) successfully recovers the true distribution.

This is an exponential family in the natural parameter η with sufficient statistic A, whose log-partition function has gradient $\mathbb{E}[A \mid \eta]$ and Hessian $\mathrm{Cov}[A \mid \eta]$; hence $\partial\,\mathbb{E}[A \mid x]/\partial\eta = \mathrm{Cov}[A \mid x]$. Chaining $\partial\eta/\partial x = (a_{\tau_0}/b_{\tau_0}^2)I$ with $\partial x/\partial\varepsilon = b_{\tau_0} I$ from the warm start gives

$$\frac{\partial \hat{A}}{\partial \varepsilon} \;=\; \mathrm{Cov}[A \mid x_{\tau_0}, o] \cdot \frac{a_{\tau_0}}{b_{\tau_0}^2} \cdot b_{\tau_0} \;=\; \frac{1}{r_{\tau_0}}\,\mathrm{Cov}[A \mid x_{\tau_0}, o],$$

which is Eq. (5). □

Remark 4 (Why the certificate reads a free signal, and why τ_0 cannot be tuned on it). For two drafts generated with $\varepsilon^i, \varepsilon^j$, Lemma 2 gives $\hat{A}^i - \hat{A}^j = r_{\tau_0}^{-1}\,\mathrm{Cov}[A \mid x_{\tau_0}, o]\,(\varepsilon^i - \varepsilon^j) + O(\|\varepsilon^i - \varepsilon^j\|^2)$, so the per-position dispersion σ_j of Eq. (6) is, to first order, a linear image of the local posterior covariance at position j: the drafts coincide exactly where the current observation determines the action and separate where it does not. Two consequences follow. First, the signal costs nothing beyond the K drafts the loop already generates. Second, the same r_{τ_0} that controls fidelity also scales the dispersion as $r_{\tau_0}^{-1}$, so raising τ_0 makes the drafts agree *more* while Corollary 3(ii) makes them *worse*. The certificate therefore cannot be used to select τ_0; Appendix B.4 shows exactly this, with the rejection rate falling monotonically from 30.6% to 8.0% across a sweep in which success does not improve.

The discrete channels. The certificate applies two different tests because the two kinds of channel fail differently. On the continuous channels a dispersion threshold is the natural statistic. On a discrete channel it is meaningless: the midpoint between two modes is not a partial command but an inadmissible one, and a draft that sits there is not "half as confident" but wrong. Those channels are therefore compared after projection—q_d maps a value onto the nearest admissible mode of channel d—so that the test asks whether the drafts agree on *which* mode, not on how far apart they are. In the three policies studied here $\mathcal{D}$ contains the gripper channel alone, coded ± 1, for which q_d reduces

to a sign test; the formulation itself does not depend on how many discrete channels an embodiment exposes, nor on how many modes each of them has.

A.4 Empirical Validation of the Recycling Premise

Probe Design. We isolate the core premise of WPFS by directly probing the policy's generative distribution offline, independent of closed-loop rollouts or student distillation. At a state where the active π_0 chunk was generated k actions ago, we draw 8 fresh full-depth plans at the current observation as a reference set. We measure where candidate chunks fall relative to this set over the next $m = 10$ actions using two normalized metrics: d_{nn} (distance to the nearest fresh plan) and d_{cent} (distance to the centroid). A value of 1.00 represents the baseline variance among genuine fresh plans. Figure 3 reports $1{,}950$ probes across 12 LIBERO-Spatial episodes.

Uncorrected Chunks Suffer Measurable Kinematic Drift. Without correction, the stale chunk monotonically drifts away from the true plan distribution. It exits the nearest-plan band by $k = 3$ and exceeds the typical plan-to-plan spread by $k^* \approx 10$. Crucially, its centroid distance d_{cent} increases concurrently, proving the chunk drifts *out* of the valid state space rather than collapsing toward its mean. This degradation is a function of elapsed time rather than position within the chunk—at $k = 10$, the divergence ratio is 1.71 over positions 0–4 and 1.81 over the last five—justifying the use of a uniform re-entry level across the entire horizon.

A Single Localized Step Recovers the Distribution. Re-entering the generative process at $\tau_0 = 0.5$ brings the chunk back within the reference plans' own spread. The corrected chunk maintains a nearest-plan distance $d_{nn} \in [0.78, 0.93]$ at all staleness levels out to $k = 30$, while d_{cent} appropriately rises to ≈ 1.0. This confirms the corrected trajectory is not merely close to an arbitrary plan, but exhibits the exact spatial variance expected of a genuine, high-quality sample.

Empirical Bounds Match Theoretical Predictions. As formalized in Proposition 1, the bandwidth r_{τ_0} must be large enough to escape the stale prior but tight enough to prevent mode-averaging. Our offline probes are consistent with this analysis. At $\tau_0 = 0.3$ ($r = 0.43$), the kernel is too narrow: the correction fails to escape the stale prior, breaching the valid distribution band by $k = 8$. Conversely, at $\tau_0 = 0.7$ ($r = 2.33$) or when predicting from pure noise, the output exhibits the *lowest* d_{nn} but collapses to $d_{cent} \in [0.53, 0.71]$. This is the mathematical signature of the conditional mean $\mathbb{E}[A \mid o_t]$—a mode-averaged trajectory that falls into out-of-distribution whitespace and would never be natively sampled by the policy. A naive nearest-plan metric would have falsely ranked this mode-averaged output first, necessitating our dual-statistic approach. Only $\tau_0 = 0.5$ ($r_{\tau_0} = 1$) correctly satisfies both criteria. Synthesized with our real-world findings (Section 4.3), these diagnostics confirm the mechanical foundation of WPFS: trajectory recycling pays massive computational dividends when chunks age against a dynamic scene, but correctly converges to a zero-margin operation when the environment remains perfectly static.

B Ablation Studies

B.1 Is the Speedup Solely Driven by Reduced Replanning?

The certificate raises the average number of actions executed per full-depth plan from 9.6 to 36.9 on π_0/LIBERO-Spatial. This substantial increase invites a natural methodological question regarding our results: could any policy that replans this infrequently achieve identical speedups, implying the gain stems merely from a sparse schedule rather than our proposed mechanism? Table 3 addresses this directly by sweeping the fixed execution horizon N_{exec} on the same backbone, suite, random seed, and 500 initial states, ensuring that every evaluated row is paired episode-by-episode.

Results demonstrate that the fixed schedule maintains baseline performance only up to 14.1 executed steps per plan—neither $N_{\text{exec}} = 5$ nor 15 statistically differs from the 10-step baseline ($p = 0.87$ in both directions). Beyond this threshold, performance degrades precipitously and significantly: dropping to 91.6% at 18.6 steps per plan ($p = 0.005$) and further to 89.0% at 26.6 steps ($p < 10^{-4}$), representing a severe loss of seven absolute points. In stark contrast, WPFS executes an average of 36.9 actions per plan—39% longer than the most aggressive static schedule tested—with no significant difference from the 10-step baseline ($18/16$, $p = 0.86$). This demonstrates that the extreme cadence achieved by WPFS cannot be replicated by any static hyperparameter tuning at

Table 3: **Static schedules fail to replicate the dynamic cadence of WPFS.** π_0 evaluated on LIBERO-Spatial (seed 7, 500 episodes per row, unified harness). "Steps per plan" denotes the realized number of actions executed per full-depth generation. The final column reports the paired McNemar comparison against the N_{exec}=10 baseline, formatted as (episodes the baseline wins / episodes it loses) alongside the exact two-sided p-value.

Execution schedule	Steps per plan	Success (%)	McNemar vs. N_{exec}=10
Fixed, $N_{\text{exec}} = 5$	4.9	95.6	$19/17, p = 0.87$
Fixed, $N_{\text{exec}} = 10$	9.6	96.0	—
Fixed, $N_{\text{exec}} = 15$	14.1	96.4	$17/19, p = 0.87$
Fixed, $N_{\text{exec}} = 20$	18.6	91.6	$39/17, p = 0.005$
Fixed, $N_{\text{exec}} = 25$	22.8	92.6	$34/17, p = 0.024$
Fixed, $N_{\text{exec}} = 30$	26.6	89.0	$51/16, p < 10^{-4}$
WPFS-Large (adaptive ℓ_t)	**36.9**	95.6	$18/16, p = 0.86$
WPFS-Small (adaptive ℓ_t)	**36.7**	95.6	$18/16, p = 0.86$

Table 4: **Ablating the contributions of recycling and verification.** π_0 evaluated on LIBERO-Spatial (seed 7, 500 episodes per row, unified harness); every configuration differs from **WPFS-Large** by exactly one parameter. Cost per action is calculated as the measured per-call latency multiplied by the total calls each run executed, divided by the total executed actions. Speedups are normalized specifically within this suite and therefore differ from the cross-suite averages in Table 1. The final column reports the paired McNemar comparison against WPFS-Large, formatted as (WPFS-Large wins / WPFS-Large loses).

Configuration	Success (%)	Steps per plan	ms per action	Speedup	McNemar vs. ours
Baseline, $N_{\text{exec}} = 10$	96.0	9.6	6.00	1.00×	$18/16, p = 0.86$
Recycling without the certificate: fixed segment of c actions					
$c = 5$	**98.2**	109.9	8.22	0.73×	$9/22, p = 0.029$
$c = 10$	95.6	114.0	4.27	1.41×	$17/17, p = 1.00$
$c = 25$	82.0	130.0	1.86	3.22×	$82/14, p < 10^{-4}$
Recycling with the certificate: agreement multiplier α					
$\alpha = 0.5$	95.6	18.7	6.46	0.93×	$21/21, p = 1.00$
$\alpha = 1.0$ (**WPFS-Large**)	95.6	36.9	3.73	1.61×	—
$\alpha = 1.5$	94.8	40.9	3.25	1.84×	$18/14, p = 0.60$
$\alpha = 2.0$	95.0	40.5	3.20	1.87×	$19/16, p = 0.74$

equivalent success levels. Consequently, the two imperatives—*minimizing planning frequency* and *ensuring trajectory validity*—cannot be decoupled by a simple schedule. Furthermore, the static schedule begins to fail at 18.6 executed actions per plan, well past the point at which the uncorrected chunk has diverged from the policy's valid plan distribution (Appendix A.4). The geometric horizon is therefore a conservative early indicator of a behavioral cliff rather than a precise predictor of its exact location.

B.2 Separating the Two Contributions: Recycling and Verification

WPFS integrates a trajectory recycling operator with an agreement-based verification certificate. Because the results reported thus far evaluate these components jointly, Table 4 isolates their respective contributions on π_0/LIBERO-Spatial. We achieve this by ablating the certificate—executing a strictly fixed segment of c actions following every correction without any acceptance test—and by systematically sweeping the certificate's single agreement threshold.

Recycling sustains long-horizon execution. Without the certificate there is no fallback, so each episode relies on exactly one initial full-depth plan and realigns it through corrections for its entire length—110 to 130 actions from a single plan, a lineage tens of corrections deep. At $c = 5$ this configuration reaches 98.2%, above both the unmodified baseline (96.0%) and WPFS-Large ($9/22$ paired, $p = 0.029$). We read this as strong evidence for the premise of Section 3.2: the structure a policy encodes into a chunk survives dozens of successive re-anchorings, suggesting that replanning mostly supplies realignment rather than new intent.

Table 5: **The realized execution horizon and certificate rejection statistics.** π_0, seed 7, 500 episodes per row. ℓ is the raw accepted-prefix length of each correction before clipping; a correction with $\ell < \ell_{\min}$ is rejected and triggers a full plan, so rejected corrections enter the quantiles with their own $\ell \in [0, \ell_{\min} - 1]$. The horizon cap is $\ell_{\max} = 25$ except on Object, where the per-suite setting is $\ell_{\max} = 10$. Baseline rows execute a fixed $N_{\text{exec}} = 10$ and lack the verification mechanism.

			Execution horizon ℓ					
Suite	Method	Success (%)	mean	p10	median	p90	Steps per plan	Rejected (%)
Object	Baseline	98.4	—	—	—	—	9.7	—
	WPFS-Large	99.4	9.03	6	10	10	36.0	8.9
	WPFS-Small	99.0	9.02	5	10	10	35.9	9.0
Spatial	Baseline	96.0	—	—	—	—	9.6	—
	WPFS-Large	95.6	14.49	1	13	25	36.9	18.5
	WPFS-Small	95.6	14.15	1	12	25	36.7	18.7
Goal	Baseline	95.6	—	—	—	—	9.6	—
	WPFS-Large	94.6	15.76	0	21	25	35.5	22.4
	WPFS-Small	95.6	16.12	0	22	25	37.0	20.3
Long	Baseline	86.2	—	—	—	—	9.9	—
	WPFS-Large	89.8	16.69	0	25	25	38.3	23.4
	WPFS-Small	89.8	16.27	0	25	25	37.2	25.3

Recycling alone buys accuracy, not speed. That same $c = 5$ configuration runs at $0.73\times$—*slower than the policy it wraps*—because correcting every five actions costs more than replanning every ten. Speed appears only when the segment lengthens, and lengthening it blindly costs accuracy: at $c = 25$, the cap WPFS itself uses, success falls to 82.0% ($p < 10^{-4}$).

The certificate makes an extended cap usable. With the agreement test, WPFS-Large runs under the same cap ($\alpha = 1$, $\ell_{\max} = 25$) and keeps 95.6% at $1.61\times$. Against the fixed rule the comparison runs in both directions: at matched success ($c = 10$, $17/17$, $p = 1.00$) the certificate is 14% faster, and at matched or greater speed ($c = 25$) the fixed rule gives up 13.6 points; only the slow $c = 5$ rule is more accurate. One caveat bounds this attribution. Because the fixed-segment arm has no fallback, it also lacks the lineage bound Λ that WPFS enforces, so these rows measure the agreement test *together with* the re-planning it triggers rather than the test in isolation. A fixed segment with a forced re-plan every Λ steps is the control that separates the two.

Threshold robustness. Across a fourfold range, $\alpha \in [0.5, 2.0]$, no setting differs significantly from WPFS-Large in success (all $p \geq 0.60$) while the speedup rises from $0.93\times$ to $1.87\times$; non-significance at 500 episodes bounds these differences rather than establishing equivalence. Below $\alpha = 1$ the certificate discards 48.5% of its corrections and the loop runs slower than the baseline; above $\alpha \approx 1.5$ the gain saturates. We keep $\alpha = 1$ because it has a parameter-free reading—accept a correction exactly as far as its dispersion matches that of a fresh plan—and because Remark 4 identifies the permissive direction as the one in which the dispersion signal becomes less informative; on this suite, $\alpha = 2$ would be an equally defensible operating point.

B.3 The Certificate in Operation

The execution horizon ℓ_t is the dynamic output generated by the certificate, and Section 3.3 establishes it as a state-dependent random variable rather than a disguised static hyperparameter. Table 5 reports its realized distribution across all four LIBERO suites, alongside the fraction of corrections refused by the certificate. These statistics are derived directly from the primary evaluation rollouts, requiring no auxiliary or isolated runs.

The execution horizon exhibits a heavy-tailed distribution. On the Spatial, Goal, and Long suites, the accepted prefix spans the full admissible range within a single continuous run: $p_{90} = 25$, strictly hitting the cap, while p_{10} remains at 0 or 1. Consequently, approximately ten percent of all corrections are aggressively truncated, and on the Goal and Long suites, that fraction is rejected entirely. No static N_{exec} can replicate this adaptive behavior, explaining why fixed schedules consistently fail to achieve comparable cadences without sacrificing task success (Appendix B.1).

Table 6: **Empirical manifestation of re-entry level failure modes during closed-loop execution.** $\pi_{0.5}$ on LIBERO-Goal, seed 7, 500 episodes per row. The rejection rate denotes the fraction of corrections refused by the certificate, each triggering a full plan. The final column reports the paired McNemar comparison against τ_0=0.5, formatted as (episodes τ_0=0.5 wins / episodes it loses) alongside the exact two-sided p-value.

τ_0	r_{τ_0}	Steps per plan	Rejection rate (%)	Success (%)	McNemar vs. τ_0=0.5
0.3	0.43	20.0	30.6	96.2	$17/4$, $p = 0.007$
0.5	1.00	28.6	16.5	**98.8**	—
0.7	2.33	35.1	8.0	98.4	$5/3$, $p = 0.73$

Rejection tracks task difficulty. The rejection rate rises monotonically with increasing suite difficulty—8.9% on Object, 18.5% on Spatial, 22.4% on Goal, and 23.4% on Long—in the exact corresponding order as the baseline success rates (98.4, 96.0, 95.6, 86.2). This mechanism does not enforce a rigid replanning rhythm; rather, it strategically allocates computationally heavy full plans to states where the environmental observation leaves the subsequent actions genuinely undetermined.

The student reproduces the teacher's certificate statistics. Section 3.4 argues that a student trained on a first-moment objective alone may under-disperse and silently disable the certificate, which the second term of Eq. (9) is designed to prevent. The student's certificate behaviour closely matches the teacher's: mean ℓ agrees to within 0.4 actions on every suite (14.15 vs. 14.49 on Spatial, 9.02 vs. 9.03 on Object) and the rejection rate to within 2 points (18.7% vs. 18.5%; 9.0% vs. 8.9%). This is consistent with the regularizer doing its job, but it is evidence of the outcome rather than of the cause: without a $\lambda = 0$ student trained under the same protocol, the match cannot be attributed to the dispersion term.

B.4 The Re-entry Level τ_0

Proposition 1 mathematically constrains the re-entry bandwidth r_{τ_0} from both sides: excessively small values trap the kernel at the stale prior, whereas excessively large values collapse the one-step evaluation to the conditional mean. Table 6 evaluates this dynamic by sweeping τ_0 in the rollout process on the $\pi_{0.5}$ backbone within the LIBERO-Goal suite at 500 episodes per configuration.

These theoretical bounds manifest as divergent empirical failure modes, cleanly exposed by the certificate's rejection rate. At $\tau_0 = 0.3$, the correction fails to escape the stale prior it originated from, causing the K candidate draws to spatially disagree. Consequently, the certificate refuses 30.6% of all corrections, degrading both inference speed (20.0 steps per plan) and task success (-2.6 absolute points). Conversely, the failure mode at $\tau_0 = 0.7$ is far more insidious. The output distribution collapses toward the conditional mean, which is inherently smooth, forcing the candidate draws to *artificially agree*: rejections artificially halve to 8.0% and the certificate accepts dangerously long prefixes, yielding the highest apparent cadence in our sweep. Because the baseline ceiling on this suite is highly saturated (98.8%), this artificial agreement does not immediately register as a measurable drop in success, potentially misleading an observer evaluating solely the success metric into concluding that $\tau_0 = 0.7$ represents the optimal operating point. What this metric conceals is that the acceptance test has quietly disengaged. Geometric analysis resolves this ambiguity where raw success rates fail: at $\tau_0 = 0.7$, the corrected trajectory exhibits a centroid distance of $d_{cent} = 0.70$ relative to the reference plans' intrinsic dispersion, compared to 1.02 at $\tau_0 = 0.5$ (Appendix A.4). This spatial collapse is the mathematical signature of a mode-averaged trajectory rather than a valid posterior sample. This exact pathology motivates the dispersion regularization utilized in our student distillation objective (Eq. (9)). Its emergence here within the core operator underscores the critical necessity of reporting spatial dispersion statistics alongside standard success rates throughout the evaluation.

C Experimental Settings

Benchmarks and Metrics. We evaluate on two simulated benchmarks and one real platform. *LIBERO* (Liu et al., 2023) provides four suites—Spatial, Object, Goal and Long—that isolate spatial

Table 7: **Comprehensive hyperparameter configurations.** The architecture, training dynamics, and structural hyperparameters are unified across all settings; variations exist solely in sequence coordinates and data scale. All students are supervised by the teacher's localized denoiser employing the dispersion regularization (Eq. (9), $\lambda = 1.0$), and are strictly never trained on human demonstrations.

	π_0 / LIBERO	$\pi_{0.5}$ / LIBERO	DB-OFT / Bridge	$\pi_{0.5}$ / X3 (Real)
Shared Configurations (All Models)				
Student Architecture	6 pre-norm blocks, $d = 512$, 8 attention heads, FFN multiplier 4			
Optimization	Learning rate: 3×10^{-4} — Training length: 60 epochs			
Framework Parameters	Drafts $K = 4$ — Dispersion $\lambda = 1.0$			
Collection Strategy	DAgger (Round 0: teacher-driven; Subsequent: student-driven with teacher labels)			
Model-Specific Configurations				
Student Parameters	27.8M	28.7M	28.4M	28.7M
Re-entry level	$\tau_0 = 0.5$	$\tau_0 = 0.5$	$\bar{\alpha} = 0.967$	$\tau_0 = 0.5$
Solver steps n	1	1	2	1
Multiplier α	1.0	1.0	1.5	1.0
Training Scenes	`env.reset()`	`env.reset()`	48 generated layouts	Hand-set layouts
Data Volume	20–60 scenes / task / suite / round		3 rounds $\times$ 4 tasks	2,588 corrections

reasoning, object variation, goal variation and long-horizon composition; each cell of Table 1 represents one suite evaluated at 10 tasks $\times$ 50 initial states (500 episodes). *SimplerEnv* (Li et al., 2024b) is a real-to-sim benchmark whose WidowX (Bridge) split contains four table-top tasks—PutCarrot, PutSpoon, StackCube and PutEggplant—and is evaluated at 24 initial layouts $\times$ 5 seeds per task (120 episodes per task, 480 per row). The primary metrics are task success rate (%) and normalized inference speedup. Speedup is computed from per-call latencies measured on an idle GPU multiplied by the exact number of calls each run made, rather than relying on wall-clock time, ensuring that hardware contention or throttling cannot artificially inflate the results. For real-robot deployments, we additionally report latency as the GPU compute time per *executed action*, averaged over an episode: it measures the average load the control loop must absorb, not the worst-case response time, which is bounded by a full plan whenever the certificate falls back.

Backbones. WPFS wraps a frozen policy: WPFS-Large requires no training, and WPFS-Small trains only its 28M-parameter student. We instantiate it on three structurally diverse dVLAs spanning different generative families, chunk geometries, and backbone computational costs: π_0 (Black et al., 2024), a flow-matching policy that denoises a 50-action chunk within a narrow action expert; $\pi_{0.5}$ (Intelligence et al., 2025), which predicts a 10-action chunk directly in the native action space; and DB-OFT (Xie et al., 2025), a DDIM policy where a 16-action chunk is denoised through a massive 7B language backbone. All baseline evaluations execute the unmodified policies at their published, default replanning cadences (10 out of 50 actions for π_0, 5 out of 10 for $\pi_{0.5}$, and 5 out of 16 for DB-OFT).

Horizon Guards and Training. The boundary guards $(\ell_{\min}, \ell_{\max}, \ell_{\text{safe}})$ are configured to $(5, 25, 10)$ for π_0, $(5, 10, 5)$ for $\pi_{0.5}$, and $(2, 10, 5)$ for DB-OFT, paired with continuous lineage limits Λ of one chunk capacity, 50, and 15 environment steps, respectively. WPFS-Small is distilled using the objective in Eq. (9) during student-driven DAgger rounds. Through this distillation, the relative correction cost γ is compressed from 0.52–0.69 (WPFS-Large) down to 0.028–0.15 (WPFS-Small).

C.1 Configurations and Training Data

What the student sees at each correction. In every setting the student's visual input is recomputed from the *current* observation at every correction: the policy's own frozen vision encoder (SigLIP for π_0 and $\pi_{0.5}$, the CLIP features before the projector for DB-OFT) encodes the current camera images, and the student cross-attends to these tokens together with the instruction embedding and, for π_0, the current proprioceptive state. Only two inputs come from the last full plan: the plan context z (the action expert's final hidden states, for $\pi_{0.5}$ and DB-OFT) and the plan age u. What the student skips is the language backbone, not the observation. Of the K drafts, the first is executed and the remaining $K - 1$ serve only the agreement test.

DAgger Rounds and Training Cost. Data collection proceeds in rounds, and which policy *drives* the robot changes between them. In round 0 the teacher drives: it plans, corrects and executes, and every correction it makes is stored together with the denoiser target that produced it. In every later round the student currently being trained drives instead, while the teacher labels the states it reaches. The asymmetry is deliberate. Corrections compound—each one is applied to the output of the last—so a student trained only on teacher-visited states would be evaluated on a state distribution it never saw. Letting the student drive moves the training distribution onto the states it will actually meet at deployment, and letting the teacher label them keeps the supervision exact. Training a single student costs approximately three hours on one RTX 5090.

Strict Isolation of Training and Evaluation States. To prevent data leakage that would artificially inflate the reported student performance, we enforce strict separation between training and evaluation environments. For LIBERO, which provides exactly 50 fixed initial states per task for benchmark evaluation, we explicitly bypass these instances during data collection. Instead, all training episodes are initialized via independent `env.reset()` calls utilizing disjoint seed blocks (enforced via `--random-scenes` and `--scene-seed`).

Conversely, SimplerEnv lacks a designated training split, defaulting entirely to 24 fixed evaluation layouts. To resolve this, we procedurally generate a disjoint training set. Our custom pipeline (`scripts/dboft_new_scenes.py`) applies spatial translations and rotations to task-specific objects within the workspace, filtering for kinematic validity. A layout is retained strictly if objects settle without drifting, remain within camera view, and maintain a minimum 2.5 cm clearance from all original layouts (dynamic objects, such as the Eggplant rolling down the sink, are geometrically validated post-settling). The formal evaluation strictly utilizes the unmodified 24 layouts to ensure rigorous parity with upstream baselines. Finally, for real-world physical deployment where algorithmic seeding is inapplicable, we manually construct a diverse distribution of object layouts deliberately distinct from the evaluation setups. The identical DAgger pipeline is executed across these configurations.

Reproducibility. The anonymized repository linked in the abstract (`https://anonymous.4open.science/r/WPFS-FFDD`) provides the model checkpoints for every student evaluated in this manuscript. Furthermore, it includes the complete data harvest scripts, training and evaluation pipelines, generated SimplerEnv training scenes, requisite upstream environment patches, and raw per-episode outcome logs. This ensures every metric reported in this paper can be reproduced end-to-end rather than merely re-derived.

C.2 Real-Robot Platform and Tasks

Hardware Platform. Physical evaluations detailed in Section 4.3 are conducted on a ROSMASTER X3 Plus mobile manipulator equipped with a 5-DOF robotic arm and a parallel gripper. The sensory suite comprises two RGB streams: a chassis-mounted overview camera and a wrist-mounted camera. The policy natively ingests both visual inputs at every step; consequently, the dual perspectives illustrated in Figure 4 represent the exact perceptual inputs processed by the model, rather than auxiliary visualization feeds. Inference is executed on a single NVIDIA RTX 5060 Ti GPU, communicating with the robotic platform via a local network interface. The closed-loop control frequency is strictly enforced at 40 Hz, corresponding to the maximum reliable read/write bandwidth of the underlying servo bus.

Policy and Data Collection. We fine-tuned the $\pi_{0.5}$ backbone on approximately 600 task-specific demonstration episodes collected on this physical platform. To ensure high-fidelity kinematic demonstrations, data collection was conducted via a VR teleoperation pipeline utilizing a Meta Quest 2 interface, Quest2ROS, and a RelaxedIK solver for precise end-effector mapping. The entire fine-tuning process required approximately 6 GPU-hours. Crucially, these identical fine-tuned weights remain strictly frozen across all evaluations in Table 2: the baseline executes the native model directly, whereas the two WPFS variants wrap it without altering a single parameter of the policy.

Task 1: Static Grasping in Clutter. The manipulator is tasked with grasping a specific red cube and depositing it into an adjacent receptacle. The core challenge is purely perceptual rather than kinematic: the workspace is populated with multi-colored distractor cubes. Across evaluation trials, the target is systematically positioned adjacent to distractors, placed directly in front of them

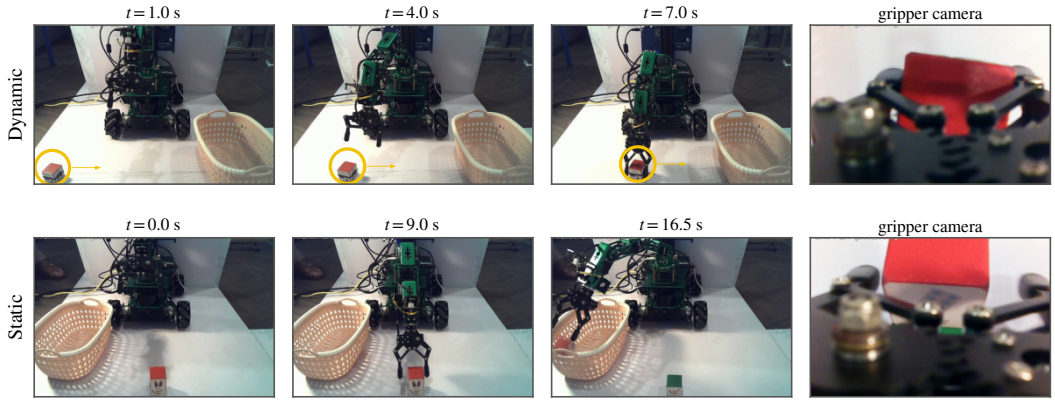


Figure 4: **Overview and wrist-camera perspectives of the physical evaluation tasks. Top (Dynamic Interception):** The target cube translates uniformly across the workspace (circled, arrow indicates velocity vector). The manipulator must track, intercept, and deposit the object into the receptacle. The sequence depicts a single trial at $t = 1$, 4, and 7 s, culminating in grasp execution. **Bottom (Static Clutter):** The manipulator targets the red cube, shown here occluding a green distractor that only becomes observable post-lift ($t = 16.5$ s). Both tasks culminate in identical placement objectives, isolating environmental dynamism during the planning phase as the sole independent variable.

(causing total occlusion until the target is displaced), or partially occluded itself. A trial is strictly recorded as a success only if the target object is stably released within the receptacle.

Task 2: Dynamic Target Interception. In this variant, the target cube translates across the workspace undergoing uniform linear motion throughout the episode. The manipulator must dynamically track, intercept, and deposit the moving object. We evaluate performance at two target velocities, 5 cm/s and 6 cm/s. These speeds were deliberately selected to bracket the kinematic failure point of the unmodified baseline policy: an action chunk generated at the onset of the reaching phase becomes critically misaligned by the time its terminal actions execute. This temporal distribution shift isolates the exact operational regime our trajectory recycling framework is engineered to resolve. Figure 4 (top) visualizes a successful dynamic interception sequence.

For both physical tasks, we conduct 50 independent evaluation trials per configuration. The target's initial spatial coordinates and, for Task 2, its specific directional velocity vector, are heavily randomized across all episodes to ensure rigorous evaluation.

D BASELINES

We benchmark WPFS against four state-of-the-art accelerators that collectively address the two orthogonal axes of the computational bottleneck defined in Section 2. The first three methods focus on reducing the intra-cycle cost *of* individual planning stages. *SpecPrune-VLA* (Wang et al., 2026) employs dual-granularity visual token pruning, combining a static per-layer budget with an adaptive controller that modulates pruning aggressiveness based on recent kinematic history. *VLA-Cache* (Xu et al., 2025) leverages temporal redundancy across consecutive visual observations; tokens classified as visually static are bypassed during recomputation and retrieved directly from the preceding KV cache, safeguarded by a task-relevance filter. *EfficientVLA* (Yang et al., 2025) implements a comprehensive, training-free architectural compression, simultaneously pruning redundant backbone layers, filtering for diverse, task-critical visual tokens, and caching intermediate representations within the action expert. Conversely, the fourth baseline, *AAC* (Liang et al., 2026), reduces the *frequency* of full-scale planning by dynamically adjusting the executed chunk length during inference. Because AAC operates on the same scheduling axis as WPFS, it serves as the closest conceptual baseline

to our self-verification certificate. To ensure equitable comparisons, four baselines are applied to identical frozen checkpoints and evaluated within our unified experimental harness.

Scope: why real-time chunking is not a baseline. Real-time chunking (RTC) (Black et al., 2025) is the best-known inference-time method for latency-sensitive action chunking, but it answers a different question from the one our tables measure. RTC hides inference latency by overlapping it with execution: the next chunk is generated asynchronously while the current one runs, with the committed actions frozen and the remainder inpainted. Every chunk is still a full-depth plan—with inpainting guidance added—so RTC does not reduce the computation spent per executed action; under our metric, measured call costs times the calls each run makes at matched success, it would sit at or below $1.00\times$ by construction. The two methods are therefore complementary rather than competing: WPFS reduces what each re-plan costs, RTC changes when that cost is paid, and nothing in either prevents executing WPFS's corrections asynchronously with RTC's inpainting. We keep every row of Tables 1 and 2 synchronous so that each speedup is attributable to computation alone, and leave the evaluation of WPFS under an asynchronous executor to future work.