TRACE: DELAY-ADAPTIVE TEMPORAL RESIDUALS FOR ASYNCHRONOUS ROBOTIC CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Asynchronous inference supports real-time action execution and improves control responsiveness by allowing the robot to continue acting while the next action sequence is generated. However, overlapping inference with execution cannot eliminate computational latency: the robot and environment may evolve before predicted actions take effect, creating an observation-to-execution mismatch that requires correction. We introduce TRACE (*Temporal Residual Adaptation for Asynchronous Control with Frozen Experts*), a temporal residual injection framework for frozen flow-based action generators. Its central mechanism compresses causal observation–command history and committed actions into delay-conditioned context, then injects this context into internal action representations through gated cross-attention. TRACE preserves the backbone's original modules and pretrained parameters while learning an additional temporal residual pathway. Its injection mechanism is orthogonal in design to delay-aware policy training and sampling-time control, enabling complementary asynchronous enhancements around a frozen backbone. Experiments on Kinetix and LIBERO show aggregate success gains over naive asynchronous execution for the initial adapted configurations. An independent Kinetix evaluation further shows improved aggregate success with the calibrated deployment configuration.

1 INTRODUCTION

Learned policies map sensory observations and task specifications to motor commands (Zhao et al., 2023; Chi et al., 2023; Black et al., 2024; Shukor et al., 2025). Their decisions pass through a control chain comprising sensing, observation transfer, inference, and actuation. Each stage adds latency; for large generative policies, action-chunk inference is a substantial component. Synchronous deployment waits for inference, tying command updates to model latency. Asynchronous deployment overlaps inference with execution of queued commands (Black et al., 2025a), maintaining command delivery while the robot and scene evolve beyond the launch-time observation. Computation delay is therefore a property of the deployed control loop that can misalign predicted commands with the state at execution. This mismatch can introduce discontinuities between action chunks, delay corrective responses to environmental changes, and reduce closed-loop task success, motivating delay compensation during action generation.

Vision-language-action (VLA) models condition motor commands on images and language instructions (Black et al., 2024; Shukor et al., 2025). They commonly predict an *action chunk*, a sequence of future commands that amortizes inference and represents coordinated motion (Zhao et al., 2023; Chi et al., 2023). Real-time chunking (RTC) generates the next chunk while the robot consumes its current queue, conditioning the prediction on actions committed before takeover (Black et al., 2025a;b). The committed prefix identifies forthcoming commands but not their realized effects. Prior work identifies *inter-chunk discontinuity*, an abrupt mismatch between consecutive chunks, and *intra-chunk inconsistency*, a misalignment between executed actions and current perception (Black et al., 2025a; Wang et al., 2026). Both arise within a closed loop that acts on predictions formed under earlier states. Boundary smoothness alone does not ensure task completion; the objective is to preserve task success as the system evolves during delayed takeover. We observe that recent history provides complementary evidence about motion and interaction, helping distinguish the observed launch state from the state in which the generated suffix will execute.

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.

We introduce TRACE, short for *Temporal Residual Adaptation for Asynchronous Control with Frozen Experts*, to condition action generation on evidence available at query launch. Observation–command history records the system's response to recent commands, and the committed prefix specifies commands scheduled during inference. For symbolic control, observed state features supply the historical evidence directly. For visual manipulation, feature differences from the latest frame additionally describe scene changes while the frozen VLA retains current-image conditioning. A delay-conditioned encoder compresses the available streams into temporal context.

The method uses this context to correct internal action features without updating the pretrained parameters. At each generation step, cross-attention injects relevant temporal information with delay-dependent strength, and the backbone continues generating actions from the corrected features. During inference, the controller executes committed commands. At takeover, it discards generated positions whose time slots have elapsed and installs the remaining sequence. Training supervises this executable suffix. At zero delay, bypassing the adapter recovers the native policy under identical inputs and sampling noise. We instantiate TRACE in Kinetix symbolic control and LIBERO visual manipulation.

TRACE is an *injection-based enhancement*, orthogonal in design to the conditioning and execution mechanisms of RTC (Black et al., 2025a), training-time RTC (TT-RTC) (Black et al., 2025b), REMAC (Wang et al., 2026), and VLASH (Tang et al., 2025), reviewed in Section 2. Section 3 contrasts synchronous waiting with asynchronous queue execution, explaining why continuous command delivery still leaves a mismatch between observation and execution. Section 4 develops TRACE through causal context encoding, temporal residual injection, and suffix supervision, then describes the choices for symbolic control and visual manipulation. Section 5 specifies the Kinetix and LIBERO training and evaluation protocols, baselines, and matched comparisons. Section 6 examines success as delay increases and its variation across levels and task suites. Section 7 studies guidance and injection strength with fixed checkpoints, distinguishing development calibration from independent confirmation. Section 8 discusses implications and further evaluation.

2 Related Work

Vision-Language-Action Policies and Action Chunking. Action-chunking policies predict future command sequences to amortize inference and represent coordinated motion. ACT learns action sequences directly, while Diffusion Policy combines generative action prediction with receding-horizon control (Zhao et al., 2023; Chi et al., 2023). Vision-language-action (VLA) policies bring pretrained vision-language representations into instruction-conditioned robot control. OpenVLA adapts a pretrained vision-language model on diverse robot demonstrations to predict actions (Kim et al., 2024). The π family connects this generalist modeling approach with continuous action generation: π_0 couples a pretrained vision-language model with a flow-matching action expert (Lipman et al., 2023; Black et al., 2024), while $\pi_{0.5}$ builds on π_0 through heterogeneous co-training on robot experience, semantic prediction, and web data to improve open-world generalization (Physical Intelligence et al., 2025). SmolVLA provides a compact flow-based VLA with an asynchronous inference stack (Shukor et al., 2025). For LIBERO, we use a task-adapted SmolVLA as the frozen policy and study temporal compensation for delayed execution.

Real-Time Chunking. RTC overlaps chunk generation with execution and uses inference-time inpainting to preserve actions committed before takeover; TT-RTC moves clean-prefix conditioning into training through sampled delays and suffix-only loss (Black et al., 2025a;b). A2C2 applies a lightweight correction at every control step, whereas REMAC combines masked action-chunk training with prefix-preserved sampling (Sendai et al., 2025; Wang et al., 2026). VLASH and FutureRTC condition generation on anticipated execution-time state or visual context (Tang et al., 2025; Jiang et al., 2026). Recent systems explore complementary deployment settings: StreamingVLA pipelines observation, action generation, and execution; FlashVLA streams action decoding; F2F-AP predicts future visual context from object flow; Jetson-PI combines future correction with onboard scheduling; DEFLECT uses latency-derived offline counterfactual tuning; and CloudEdgeVLA combines delayed cloud features with current edge observations (Shi et al., 2026; Li et al., 2026; Wei et al., 2026; Yang et al., 2026; Zhu et al., 2026; Peng et al., 2026). These approaches also differ in how they evaluate delay robustness. RTC, TT-RTC, A2C2, VLASH, and REMAC report closed-loop success across injected delays, but use different benchmarks and grids. A unified study compares

IT-RTC, TT-RTC, VLASH, and A2C2 on Kinetix and LIBERO through $d = 20$ (Agouzoul, 2026). FlashVLA and CloudEdgeVLA extend delay sweeps to streaming and cloud–edge settings; other systems emphasize fewer delay settings, throughput, or aggregate performance.

3 Synchronous and Asynchronous Execution

Figure 1 compares two consecutive action submissions under synchronous and asynchronous control. Let Δ be the control period, H the predicted action horizon, and R the number of predicted actions consumed per replanning cycle. In the illustrated schedules, a query launched at tick t produces a chunk A_t that starts supplying actions at $t + d$, giving a launch-to-execution delay $\delta = d\Delta$.

A synchronous controller acquires an observation, waits for inference, and then executes the returned chunk. In Figure 1(a), the first query launches at tick zero and submits $A^{(1)}$ at d. After executing R actions, the controller launches a second query at $R + d$ and submits $A^{(2)}$ at $R + 2d$. Successive submissions are therefore separated by $R + d$ ticks, with an inference wait in every cycle. Longer or variable response times introduce pauses and irregular command updates. Waiting does not freeze the physical system: the robot or environment may move while a command is held, making the launch observation stale before execution begins. Under high concurrency, batching and contention for a shared inference service can further increase and vary response times, propagating these waits into each synchronous control loop.

An asynchronous controller consumes its existing queue while generating the next chunk (Black et al., 2025a). After initialization with $A^{(0)}$, queries launch at $t_1 = R - d$ and $t_2 = 2R - d$, with takeovers at R and $2R$ (Figure 1(b)). The committed prefix $P_t = (a_t, \ldots, a_{t+d-1})$ consists of the next d actions in the queue at launch: P_{t_1} comes from $A^{(0)}$, and P_{t_2} from the previously submitted $A^{(1)}$. These actions execute while inference uses a fixed snapshot of observation, history, and prefix. At takeover, the controller discards $A_t[0 : d]$ and installs $A_t[d : H]$, since P_t has already supplied the elapsed positions. This overlap maintains command delivery but leaves an observation-to-execution mismatch: the query observes s_t, whereas committed actions and external changes may produce a different state s_{t+d}. Knowing P_t specifies the commands, not their realized effects, so stale predictions can still cause abrupt transitions and ineffective corrections. Under high concurrency, asynchronous execution is needed to sustain continuous command delivery while shared inference requests remain pending, decoupling queue consumption from query completion. This benefit requires adequate queue coverage and timely results; the policy must still account for state evolution, and scheduling alone does not guarantee hardware deadlines.

4 Method

The observation available at query launch generally differs from the state in which the generated actions begin to execute. A current observation describes the scene at one instant, while the committed prefix specifies intended commands; neither alone identifies ongoing motion or the effects of recent interaction. TRACE addresses this information gap through a causal temporal encoder, gated hidden-state residuals within a frozen flow-based action generator, and suffix supervision aligned with asynchronous takeover. The same adaptation mechanism supports a symbolic-observation control policy and a visual manipulation policy, with input branches, insertion points, and regularization matched to each domain.

4.1 Causal Temporal Context

Observation–command history. The same observed pose can occur during approach, retreat, or recovery from contact. To distinguish these situations, we pair historical observed state features with the previously issued command a_{u-1}, its validity indicator, and the observation's time relative to launch. These features are symbolic observations in Kinetix and proprioceptive measurements in visual manipulation. A recurrent encoder processes valid samples chronologically and retains their hidden features, exposing the relationship between commands and observed responses without requiring explicit velocity estimates or a transition model. Issued commands need not equal realized actuation under environment noise.

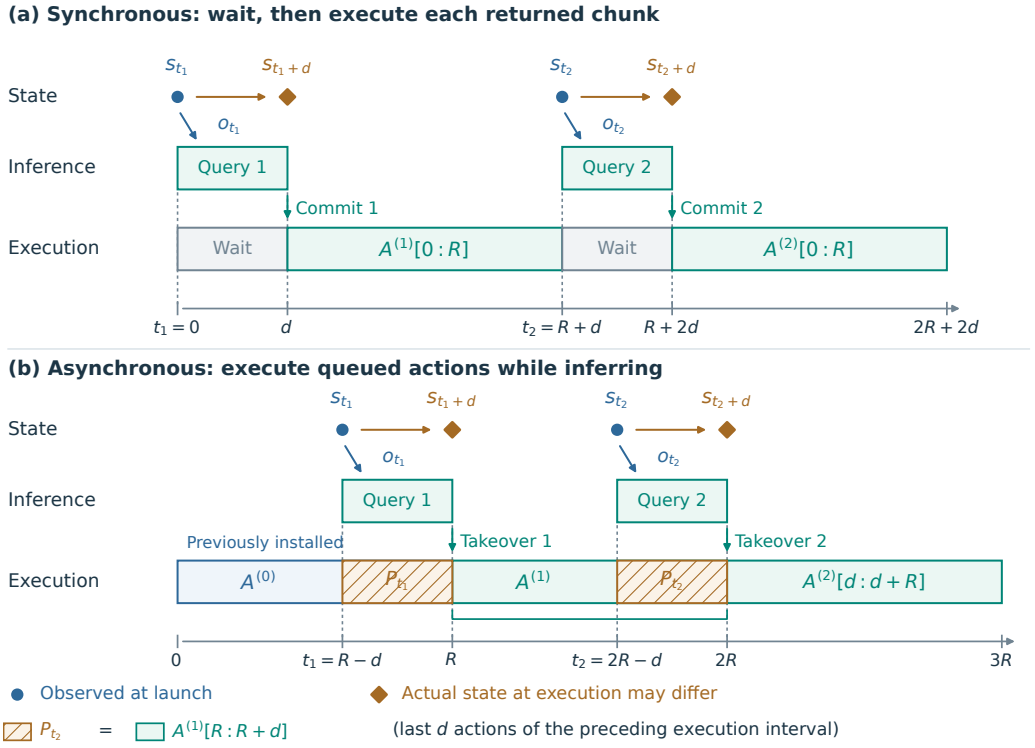

Figure 1: Two consecutive action submissions. **(a)** Synchronous control alternates inference waits and execution, submitting $A^{(1)}$ and $A^{(2)}$ after their respective queries. **(b)** Asynchronous control generates these chunks while consuming the existing queue, with takeovers separated by R ticks. The committed prefixes P_{t_1} and P_{t_2} are portions of the previously installed $A^{(0)}$ and $A^{(1)}$, respectively. Observations describe the state at query launch; the state when each new chunk begins execution may differ.

Visual changes. When images are available, the frozen VLA already represents current appearance. The temporal branch therefore encodes changes relative to the latest available frame. Let $z_{u,c,r}$ be the pooled frozen image feature at capture time u, camera c, and region r, and let $u_c^\star$ index that camera's latest valid frame in the launch snapshot. The temporal visual input is

$$\Delta z_{u,c,r} = z_{u,c,r} - z_{u_c^\star,c,r}. \tag{1}$$

Feature differences emphasize temporal variation without explicitly estimating optical flow or displacement. Projected features receive capture-time, camera, region, and modality embeddings. Only frames captured and received by launch are valid; missing entries are masked. Training sometimes masks the entire visual-history branch to encourage use of other temporal evidence, while retaining the native current-image pathway.

Committed actions and temporal compression. Recent history describes how the system has evolved; the committed prefix describes the commands that will be applied before takeover. A separate Transformer encoder represents P_t, with each command augmented by its execution offset $i\Delta$. All prefix commands are known at launch, so bidirectional attention over this prefix is causally admissible. Denoting the state–command and visual histories by $\mathcal{H}_t^s$ and $\mathcal{H}_t^v$, respectively, we form a masked memory

$$M_t = \left[E_s(\mathcal{H}_t^s);\ E_v(\mathcal{H}_t^v);\ E_p(P_t, \{i\Delta\}_{i=0}^{d-1})\right]. \tag{2}$$

Here E_v includes Equation 1 and is omitted for symbolic observations; modality embeddings distinguish the available streams. Learned queries conditioned on the planned delay compress this memory into a fixed-size representation:

$$Q^{(0)} = Q_{\text{learned}} + e_\delta(\delta), \qquad Q^{(b+1)} = \mathcal{C}_b(Q^{(b)}, M_t), \qquad T_t = Q^{(B)}, \tag{3}$$

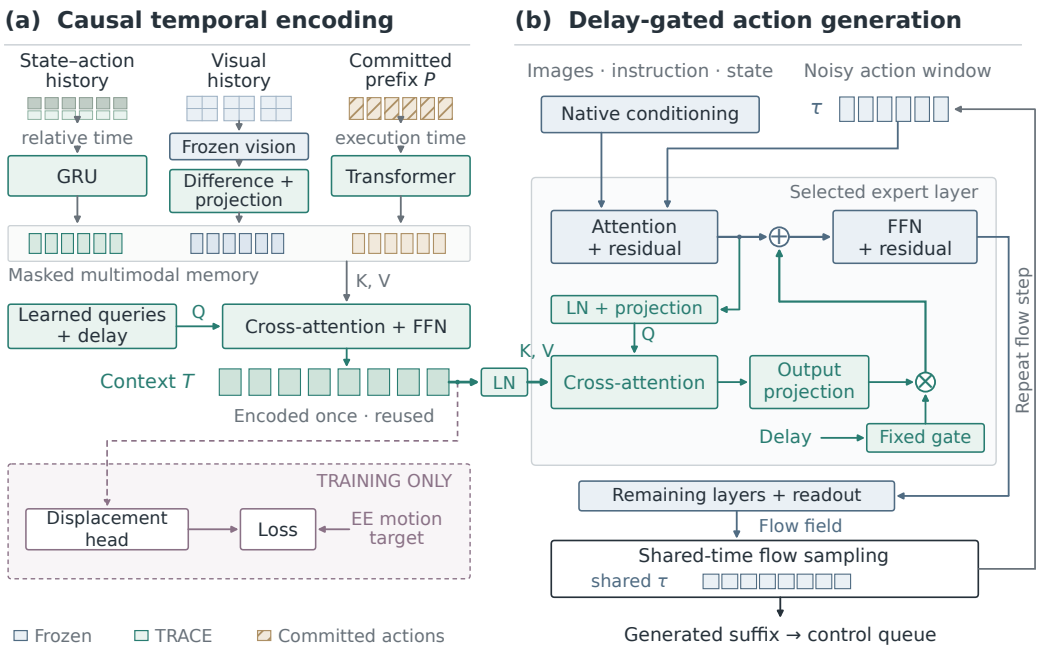

Figure 2: Visual manipulation instantiation of TRACE. **(a)** State–command history, visual feature differences, and committed actions form masked memory, compressed once per launch into delay-conditioned context. **(b)** Selected frozen expert layers attend to this context and add a gated residual before the native feed-forward block. Only the generated suffix enters the queue. Blue-gray denotes frozen modules, teal the adapter, brown hatching committed actions, and the purple dashed branch training-only displacement supervision. The symbolic instantiation omits the visual and displacement branches and adapts mixer blocks.

where $\mathcal{C}_b$ is a residual cross-attention and feed-forward block. Delay conditioning selects evidence relevant to the takeover interval. Time embeddings use control ticks in symbolic control and seconds in manipulation, with offsets and delay expressed consistently within each instance. The tokens T_t are computed once per launch and reused throughout generation.

4.2 Delay-Gated Residual Adaptation

The frozen action generator supplies the native task-conditioned computation, while the adapter learns how recent evolution should modify it. We insert cross-attention residuals at internal block boundaries: after complete mixer blocks in the symbolic policy, and after attention and before the native feed-forward block in the manipulation Transformer. Let h_ℓ denote the action features at such a boundary. Each injection normalizes the temporal tokens before using them as attention memory:

$$\overline{T}_{t,\ell} = \mathrm{LN}_\ell^{\mathrm{mem}}(T_t), \qquad q_\ell = W_\ell^q \,\mathrm{LN}_\ell^{\mathrm{act}}(h_\ell), \tag{4}$$

$$r_\ell = W_\ell^o \,\mathrm{MHA}_\ell\big(q_\ell, \overline{T}_{t,\ell}, \overline{T}_{t,\ell}\big), \tag{5}$$

$$\widetilde{h}_\ell = h_\ell + g(\delta) r_\ell, \qquad g(\delta) = 1 - \exp(-\delta/\delta_{\mathrm{ref}}). \tag{6}$$

The attention module projects the normalized memory into keys and values; subsequent native computations continue from $\widetilde{h}_\ell$. The query depends on evolving action features, so the correction can change across flow steps while T_t remains fixed. This is an internal representation residual, not a separate action-output predictor.

The reference timescale $\delta_{\mathrm{ref}} > 0$ controls gate growth: longer delays permit stronger corrections, without bounding the residual norm. At zero delay, an explicit bypass recovers native computation for identical inputs and noise. Small nonzero output-projection initialization limits the initial perturbation. Backbone parameters remain fixed while gradients propagate through their computations to the adapter.

The injection interface is orthogonal in design to the host's conditioning and execution mechanisms. Composition with TT-RTC, REMAC, or VLASH would require matching adapter training and inference to their temporal inputs, flow-time conventions, and prefix constraints.

Guided sampling also depends on residual strength. We therefore allow a deployment gate $\widehat{g}(\delta) = \alpha(d)g(\delta)$, with $\alpha(d) \in [0, 1]$. Kinetix attenuates shorter-delay injections to temper adaptation when observation staleness is limited; LIBERO retains unit scaling. This scales internal residuals, without implying proportional changes in final actions. Training retains the original gate; calibration leaves all learned parameters unchanged.

4.3 Flow Matching for the Executable Suffix

Both instantiations train with the backbone's shared-time flow over the complete action window, conditioning on P_t through T_t without substituting clean prefix tokens into the flow input. The queue executes committed actions before takeover.

Let $A^\star$ be a demonstration window in the backbone's native action coordinates, including its normalization and padding where applicable, and let $\epsilon \sim \mathcal{N}(0, I)$ have the same shape. With noise-to-data time $\tau \in [0, 1]$, the training input is

$$X_\tau = (1 - \tau)\epsilon + \tau A^\star, \qquad \tau_i = \tau. \tag{7}$$

Flow times follow the native training distribution and are shared by all action positions, including those preceding takeover. Let c_t denote the backbone's current-observation and task conditioning, and let $\mathcal{S}_t$ contain valid physical action coordinates at positions $i \geq d$. The flow objective is

$$\mathcal{L}_{\text{flow}} = \mathbb{E}\left[\frac{1}{|\mathcal{S}_t|}\sum_{(i,j)\in\mathcal{S}_t}\left(v_{\theta_0,\phi}(X_\tau, \tau; c_t, T_t)_{ij} - (A^\star - \epsilon)_{ij}\right)^2\right], \tag{8}$$

where θ_0 denotes the frozen backbone and ϕ the temporal encoder and injection parameters. Sampled training delays vary the takeover position. The loss is normalized per sample and excludes pre-takeover positions, padding, and unused latent coordinates, aligning supervision with actions that can still affect execution. Pre-takeover tokens remain computational context but are not direct targets.

Inference starts from noise and integrates a sampling velocity v^{samp}, expressed here in noise-to-data coordinates:

$$X_{k+1} = X_k + \Delta\tau_k\, v^{\text{samp}}(X_k, \tau_k; c_t, T_t). \tag{9}$$

In our experiments on Kinetix (Matthews et al., 2025), we add RTC-style old-plan consistency guidance (Black et al., 2025a), with full weight on committed positions and decreasing weights on part of the uncommitted plan. A vector–Jacobian product differentiates through the adapted endpoint predictor with temporal context fixed. Injection therefore affects both velocity and guidance direction, motivating joint calibration of residual strength and sampling constraints. Shared flow time is retained without prefix clamping; zero delay uses the native sampler. Appendix A.1 specifies this soft guidance and the binary-prefix reference variant. In our experiments on LIBERO (Liu et al., 2023), $v^{\text{samp}} = v_{\theta_0,\phi}$: the native sampler directly integrates the adapted field.

At takeover, both controllers discard the generated pre-takeover positions and install only the suffix. Committed-command execution is therefore preserved by the queue even when predicted prefix values differ. Offline prefixes come from demonstrations and online prefixes from the policy queue, leaving a distribution shift that suffix imitation alone does not remove.

4.4 Domain-Specific Regularization

Suffix imitation provides an indirect signal for temporal context. Kinetix regularizes the adapted velocity toward the frozen field on valid suffix coordinates, limiting unnecessary changes to an already competent base policy:

$$\mathcal{L}_{\text{res}} = \mathbb{E}_{d>0}\left[\frac{1}{|\mathcal{S}_t|}\sum_{(i,j)\in\mathcal{S}_t}\left(v_{\theta_0,\phi}(X_\tau, \tau; c_t, T_t)_{ij} - v_{\theta_0}(X_\tau, \tau; c_t)_{ij}\right)^2\right]. \tag{10}$$

In visual manipulation, end-effector position supplies a physically meaningful auxiliary target: a training-only head predicts displacement over the planned delay from pooled temporal tokens,

$$\widehat{\Delta p}_t = f_\psi(\mathrm{Pool}(T_t)), \qquad \Delta p_t = p_{t+d} - p_t. \tag{11}$$

If $\mathcal{J}_t$ contains valid displacement coordinates and σ_j their training-set scales, the auxiliary objective is

$$\mathcal{L}_{\mathrm{disp}} = \mathbb{E}_{d>0}\left[\frac{1}{|\mathcal{J}_t|}\sum_{j\in\mathcal{J}_t}\mathrm{Huber}_\kappa\left(\frac{\widehat{\Delta p}_{t,j} - \Delta p_{t,j}}{\sigma_j}\right)\right], \tag{12}$$

Future states supply labels only: the head receives causal context, never conditions action generation, and is removed at deployment. Kinetix omits this head because symbolic control has no shared end-effector target. The objective is $\mathcal{L} = \mathcal{L}_{\mathrm{flow}} + \lambda_{\mathrm{disp}}\mathcal{L}_{\mathrm{disp}} + \lambda_{\mathrm{res}}\mathcal{L}_{\mathrm{res}}$, enabling only the domain-appropriate regularizer. Both variants share causal compression, internal adaptation, and suffix supervision. We evaluate exponential-moving-average weights; dimensions and optimization settings appear in Appendix A.

5 Experimental Protocol

Within each cohort, comparisons match the frozen backbone, episodes, sampling noise, and asynchronous schedule. Naive Async uses only the current observation and task input. Kinetix combines adaptation with guidance and independently confirms deployment calibration against Naive Async, RTC, and the preceding adapted configuration; LIBERO adds the adapter to the native sampler.

5.1 Kinetix: Symbolic Control

We use 12 Kinetix physics-control levels (Matthews et al., 2025) and released per-level RTC flow policies (Black et al., 2025a), each mapping 679-dimensional symbolic observations to six-dimensional actions through a frozen MLP-Mixer. Per-level compact adapters have approximately 387.7K trainable parameters, trained on public demonstrations for 3,000 updates with batch size 128 and seed 42. EMA selection minimizes fixed-validation suffix flow loss, excluding regularization. Episode-level splits hold out data only from adapter training: the released backbones have already used the public corpus.

The horizon is $H = 8$, history contains four consecutive observations, and replanning uses $R = 4$. Training samples $d = 1$–4 uniformly. Evaluation uses five flow steps, 256 episodes per condition, a 256-step limit, and actuator-noise standard deviation 0.1, with CPU physics and CUDA 13 inference. Paired seeds and initial-observation hashes agree. Success uses the first episode's solved flag, averaged equally across levels; delays are simulated control ticks.

Figure 3 separates the initial five-method evaluation, following selection on three development levels, from independent calibration confirmation. Calibration screens eight configurations on all twelve levels using a separate development cohort and 64 episodes per condition. The selected soft guidance and residual attenuation reuse existing checkpoints and are fixed before confirmation. Appendices A.1 and B.1 detail selection, evaluation seeds, and numerical results.

5.2 LIBERO: Visual Manipulation

We use 40 tasks across LIBERO-Spatial, Object, Goal, and Long (Liu et al., 2023). The 1,693 demonstrations are split by episode within task into 1,357/168/168 adapter training/validation/test episodes, yielding 218,971 training anchors. The released SmolVLA backbone has already seen the full source dataset. We train approximately 16.88M adapter and auxiliary parameters for 10,000 updates with batch size 256 and seed 42, selecting EMA weights at update 6,000 by fixed-validation action loss. Training samples zero delay with probability 0.1 and otherwise uniformly samples legal delays through ten ticks, respecting demonstration boundaries. Appendix A gives architecture and optimization details.

Evaluation uses control period $\Delta = 0.05$ seconds, horizon $H = 50$, takeover stride $R = 5$, and ten Euler steps for both methods. We report simulated delays $d = 0$–3 on 50 official initial states

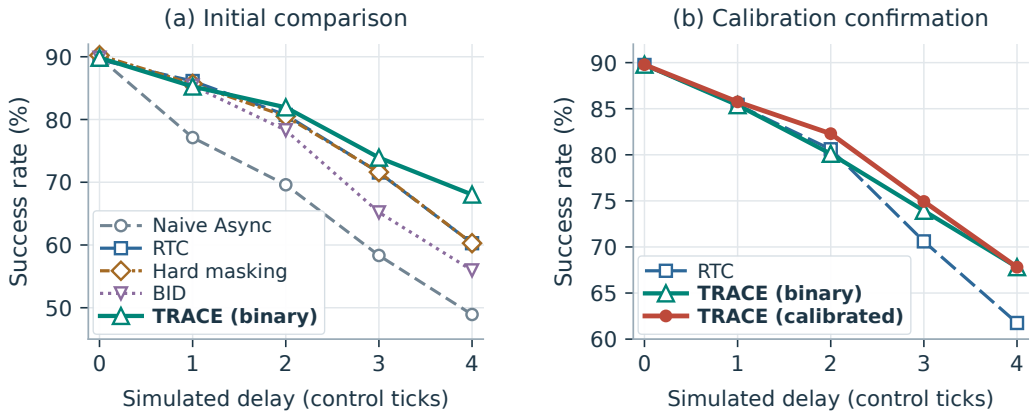

Figure 3: Kinetix success: **(a)** initial comparison; **(b)** independent calibration confirmation. Panels use different vertical scales; points average 12 levels with 256 episodes per condition. Calibration changes guidance and residual strength with fixed weights. Appendix B.1 provides all rates, including confirmation Naive Async, and evaluation seeds.

per task, giving 2,000 episodes per method and delay. Each suite uses its official success criterion and control-step limit; the zero-delay native policy is displayed once. Both arms share system-default EGL, backbone and processor hashes, runtime configuration, batch size ten, episode identities, initial-observation hashes, and sampling-noise seeds. Two anomalous initial observations were repaired using hash agreement, independently of success (Appendix D).

The reference recipe was selected over two variants on initial states 0–4 and fixed before evaluation on all states 0–49. The full cohort therefore includes states used for candidate selection. These delays lie within training support and specify simulated action availability, independently of measured inference time.

6 Results

Kinetix success across delays and levels. Calibrated TRACE averages 77.69% success over $d = 1$–4, versus 76.81% for its binary-guided predecessor and 74.59% for RTC (Figure 3(b)), exceeding RTC on ten of twelve levels. Same-cohort Naive Async achieves 63.35% (Appendix B.1). The $d = 1$ separation among guided methods is small; both adapted variants coincide at $d = 4$, where injection strength and guidance weights match. In the earlier cohort (Figure 3(a)), binary-guided TRACE reaches 77.29%, versus 63.49% for Naive Async and 74.52% for the same binary guidance without adaptation. The Naive Async comparison includes adaptation and sampling effects; the binary-guidance control supports an additional gain from temporal injection.

LIBERO success across delays and task suites. Table 1 reports success on all 50 initial states (0–49) per task for delays $d = 0$–3. Averaged over positive delays $d = 1$–3, TRACE achieves 82.85% success, compared with 82.47% for Naive Async. The corresponding suite averages are 86.27% versus 85.27% on Spatial, 91.20% for both methods on Object, 87.87% versus 87.73% on Goal, and 66.07% versus 65.67% on Long. Thus the aggregate success rates remain close under the tested schedule, with the largest separation on Spatial and the lowest absolute success on Long.

At zero delay both policy branches reduce to the native computation; the table therefore reports one native reference, with the original rollout records documented in Appendix D. Appendix B.2 provides further suite-level results.

Table 1: Success rates (%) on each LIBERO suite at takeover stride $R = 5$. Each suite contributes 500 episodes per method and delay; Overall averages the four suites. One delay tick is 0.05 seconds. Since the zero-delay adapter is bypassed, $d = 0$ is shown once as a native reference using the recorded Naive Async run. Bold indicates the higher success rate within each delay and metric; ties are bold for both methods.

Delay d	Method	Spatial	Object	Goal	Long	Overall
0	Native reference	85.20	90.20	87.20	67.80	82.60
1	Naive Async	85.20	90.60	**87.80**	**65.80**	82.35
	TRACE	**87.20**	**91.00**	87.60	65.20	**82.75**
2	Naive Async	85.00	**91.80**	87.60	64.40	82.20
	TRACE	**86.00**	90.40	**87.80**	**66.40**	**82.65**
3	Naive Async	**85.60**	91.20	87.80	**66.80**	82.85
	TRACE	**85.60**	**92.20**	**88.20**	66.60	**83.15**

Table 2: Kinetix deployment calibration with fixed adapter weights. Success rates (%) use the same development cohort; all rows retain temporal injection. Bold marks each column's highest rate.

Guidance	Extra attenuation	$d = 1$	Mean $d = 1$–4
Binary	None	84.38	75.94
Soft	None	83.20	75.78
Binary	Delay-dependent	84.11	76.76
Soft	Delay-dependent	**87.37**	**78.06**

7 Temporal Injection and Sampling Calibration

Temporal injection changes both sampling velocity and the endpoint predictor used for guidance. Table 2 crosses binary versus soft guidance with additional delay-dependent attenuation, using fixed Kinetix checkpoints and the same development episodes (12 levels, 64 episodes per condition). Every row retains the adapter and its exponential gate; "None" omits only extra attenuation.

Soft guidance alone does not improve mean success, while attenuation improves it under either guidance rule. Their combination performs best overall and at $d = 1$, motivating joint calibration without retraining. The selected configuration also improves mean success in independent confirmation (Section 6). That evaluation confirms the selected recipe without repeating all four controls or isolating encoder components. Appendix C reports all candidates and LIBERO training diagnostics.

8 Conclusion

TRACE offers an injection-based approach to improving policy robustness under asynchronous inference. A learned temporal pathway corrects internal action representations for delayed execution while preserving the original policy parameters. Both benchmarks show aggregate gains, with modest improvements in LIBERO. TRACE is thus orthogonal in design to RTC, TT-RTC, REMAC, and VLASH, offering an additional injection pathway for previously fine-tuned policies or training-free inference enhancements. Kinetix demonstrates composition with RTC-style guidance; integrations with the other methods require matching their conditioning and sampling conventions and remain to be evaluated. Evidence is limited to one training seed and takeover stride per benchmark, simulated delays within training support, and development-based selection. Further component studies and hardware evaluations should address retraining variability, larger delays, timing jitter, and sensor transport; simulation does not establish hardware deadline feasibility or physical stability.

AI USE STATEMENT

This work was completed through extensive collaboration between the human authors and generative AI tools. AI assistance covered: (1) gathering materials for the initial literature review; (2) assessing and selecting the initial methodological approach; (3) modifying training and experimental code after the core design had been established; (4) automating experiments, including exploration of parameter settings and ablation studies; and (5) polishing and assisting with manuscript drafting. All AI-assisted work was conducted under close supervision by the human authors. The paper's central claims, methodology, and experimental results have all undergone a second review by the human authors, who take responsibility for the final content.

REFERENCES

Ayoub Agouzoul. Understanding asynchronous inference methods for vision-language-action models. *arXiv preprint arXiv:2605.08168*, 2026. URL https://arxiv.org/abs/2605.08168.

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. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164v1.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. URL https://papers.nips.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. *arXiv preprint arXiv:2512.05964*, 2025b. URL https://arxiv.org/abs/2512.05964.

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 *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://roboticsproceedings.org/rss19/p026.html.

Hai Jiang, Yixian Zou, Binbin Liang, Boqian Liu, Fanman Meng, and Shuaicheng Liu. FuturERTC: Real-time robot execution with anticipatory-conditioned action chunking. *arXiv preprint arXiv:2607.24008*, 2026. URL https://arxiv.org/abs/2607.24008v1. Version 1.

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. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

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

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Michael Matthews, Michael Beukman, Chris Lu, and Jakob Foerster. Kinetix: Investigating the training of general agents through open-ended physics-based control tasks. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/936ce22b767cf1a1496083e4725d3b21-Abstract-Conference.html.

Daojie Peng, Fulong Ma, Bingtao Wang, Sheng Wang, and Jun Ma. Latency-tolerant cloud-edge collaborative vision-language-action models via emergent representational specialization. *arXiv preprint arXiv:2608.00569*, 2026. URL https://arxiv.org/abs/2608.00569.

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. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for VLA action chunks. *arXiv preprint arXiv:2509.23224*, 2025. URL https://arxiv.org/abs/2509.23224.

Yiran Shi, Dongqi Guo, Tianchen Zhao, Feng Gao, Liangzhi Shi, Chao Yu, Zhijian Mo, Qihua Xiao, Xiaoshuai Peng, Qingmin Liao, and Yu Wang. StreamingVLA: Streaming vision-language-action model with action flow matching and adaptive early observation. *arXiv preprint arXiv:2603.28565*, 2026. URL https://arxiv.org/abs/2603.28565.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025. URL https://arxiv.org/abs/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. *arXiv preprint arXiv:2512.01031*, 2025. URL https://arxiv.org/abs/2512.01031v2. Version 2, revised July 2026.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Yuzhang Shang, Ramana Rao Kompella, and Gaowen Liu. Real-time robot execution with masked action chunking. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2601.20130.

Haoyu Wei, Xiuwei Xu, Ziyang Cheng, Hang Yin, Angyuan Ma, Bingyao Yu, Jie Zhou, and Jiwen Lu. F2F-AP: Flow-to-future asynchronous policy for real-time dynamic manipulation. *arXiv preprint arXiv:2604.02408*, 2026. URL https://arxiv.org/abs/2604.02408v2. Version 2.

Zebin Yang, Qi Wang, Yunhe Wang, Xiurui Guo, Bo Yu, Shaoshan Liu, Jiafeng Xu, Hao Dong, and Meng Li. Jetson-PI: Towards onboard real-time robot control via foresight-aligned asynchronous inference. *arXiv preprint arXiv:2607.12659*, 2026. URL https://arxiv.org/abs/2607.12659v5. Version 5.

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. doi: 10.15607/RSS.2023.XIX.016. URL https://roboticsproceedings.org/rss19/p016.html.

Yixiang Zhu, Yonghao Chen, Rui Meng, Jingyu Guo, Jiaxiang Zou, Zijie Yang, Taowen Wang, and Xinyu Chen. DEFLECT: Delay-robust execution via flow-matching likelihood-estimated counterfactual tuning for VLA policies. *arXiv preprint arXiv:2605.19294*, 2026. URL https://arxiv.org/abs/2605.19294.

A Implementation and Training Details

A.1 Kinetix Configuration and Guided Sampling

Both Kinetix configurations in Table 5 reuse the same trained compact adapters. The initial configuration uses binary-prefix guidance; calibrated TRACE changes only the deployment sampler and residual strength. Table 3 records the shared training settings and final deployment settings. Adapter inputs are standardized using training-episode statistics and clipped, while the frozen policy retains its native symbolic observations and raw action coordinates. Compression-memory normalization is enabled in Kinetix; injection-memory normalization is present in both domains. Each mixer block receives a hidden-state residual, and the native final normalization and velocity-output projection remain frozen. There is no visual branch, displacement head, or future-symbolic prediction target.

Table 3: Shared Kinetix training settings and calibrated deployment configuration.

Component	Setting
Frozen base	Per-level released step-24 MLP-Mixer flow policy
Temporal width / tokens / attention heads	64 / 2 / 4
History / prefix / compression	One GRU / Transformer / cross-attention block each
History observations	Four consecutive control-step observations
Normalization	Adapter branches clipped to $[-5, 5]$; compression K/V norm
Injection	After all four mixer blocks; memory LayerNorm
Training gate / output initialization	$1 - \exp(-d/2)$ / 10^{-3}
Deployment scale $\alpha(d)$, $d = 0, \ldots, 4$	$[0, 0.25, 0.5, 0.75, 1]$
Flow / guidance	Five Euler steps; soft RTC VJP guidance, cap 5
Objective	Suffix flow MSE; velocity-residual weight 0.002
Optimizer	AdamW; LR 10^{-4}; weight decay 0.05; gradient clip 1
Updates / batch / training seed / EMA	3,000 / 128 / 42 / 0.999
Validation	512 fixed windows; every 100 updates; flow loss only
Trainable parameters per level (approx.)	387.7K

For the guided sampler, write the adapted velocity, including any deployment attenuation, as v_ϕ and the endpoint prediction as $D_\tau(X) = X + (1 - \tau)v_\phi(X, \tau)$. If A_{old} is the remaining plan and W_d contains its consistency weights, the sampling field is

$$v^{\text{samp}}(X, \tau) = v_\phi(X, \tau) + \omega(\tau) J_{D_\tau}(X)^\top W_d \big(A_{\text{old}} - D_\tau(X)\big), \tag{13}$$

with the native RTC guidance schedule capped at five. This vector–Jacobian product differentiates through the adapted endpoint predictor with temporal context fixed. Let $C = H - R$ be the available old-plan horizon. Binary-prefix guidance uses weights $w_i = \mathbf{1}[i < d]$. Soft RTC guidance retains $w_i = 1$ for $i < d$, sets $w_i = 0$ for $i \geq C$, and uses $w_i = u_i(e^{u_i} - 1)/(e - 1)$ for $d \leq i < C$, where $u_i = (C - i)/(C - d + 1)$. These weights apply equally to all action coordinates. Soft guidance therefore also constrains part of the uncommitted old plan, whereas the temporal encoder still receives only P_t. Neither weighting scheme enforces exact prefix equality by projection.

The calibrated sampler multiplies each internal residual by $\alpha(d)$ from Table 3, in addition to the exponential gate. Training retains the original gate and suffix velocity regularizer; no prefix-specific penalty or new checkpoint is used by the selected configuration. At $d = 0$, both adapted variants explicitly use the native sampler and match Naive Async in all 3,072 confirmation pairs on success, return, length, and truncation. RTC retains soft old-plan guidance at zero delay and is reported separately. At $d = 4 = C$, the two guidance masks coincide and $\alpha = 1$; both adapted variants therefore produce identical confirmation records.

After warm start, each replanning cycle executes the previous plan's first d actions followed by the new plan's positions d through $R - 1$; positions from R onward remain queued for the next cycle.

This implements the committed-prefix principle with a benchmark-specific bootstrap phase. Frame skip is two and only the first episode is scored, masking subsequent automatic resets.

Initial selection compared the original, regularized, and compact adapters with plain flow, soft RTC guidance, and binary-prefix guidance on Grasp, Cartpole, and Walker. Screening used seed 424241 and 64 episodes per condition; the two leading candidates were rechecked with seed 424242 and 128 episodes. The selected compact configuration was fixed before evaluation on seed 271829. All twelve selected EMA checkpoints occur at update 3,000, and tensor audits record unchanged backbone weights. The initial grid contains 76,800 episodes across the five methods in Table 5a. RTC uses soft old-plan guidance; Hard masking uses binary-prefix guidance without the adapter. BID uses the recorded 16-candidate strong-policy selection configuration, without a separate weak-policy checkpoint.

The refinement uses the initial results for diagnosis and screens eight configurations on all twelve levels with development seed 424341. Candidates vary guidance, residual attenuation, and a separately trained prefix-regularized alternative. Eligibility requires no decrease at $d = 1$ or in the mean over Catapult, Swimmer, and Half-cheetah, and at most a 0.5-percentage-point loss in the overall positive-delay mean relative to the binary-guided reference. Eligible candidates are ranked by a weighted sum of overall positive-delay success (0.5), $d = 1$ success (0.25), and the three-level mean (0.25). Selection fixes soft guidance with delay-dependent residual attenuation before confirmation on seed 271930, without further candidate selection. The development and confirmation grids contain 42,240 and 61,440 episodes, respectively. The latter includes Naive Async, RTC, and both adapted variants; Table 5b reports all four methods from this confirmation cohort. Confirmation uses independent episode seeds on the same levels, rather than unseen tasks.

A.2 LIBERO Configuration

Table 4 specifies the standard TRACE training recipe for LIBERO. The selected weights are the EMA parameters at update 6,000; the run completed 10,000 updates. Subsequent evaluations retain these weights and use shared-time flow sampling without action-prefix clamping or clean-prefix flow-time overrides. The recipe was selected among fixed candidates using an earlier small closed-loop comparison, so those selection results are exploratory.

Table 4: Implementation settings for the LIBERO instantiation.

Component	Setting
Backbone	Released task-adapted SmolVLA; all weights frozen
State / physical action dimensions	8 / 7
Temporal width / tokens / attention heads	384 / 8 / 8
State–command encoder	Two-layer GRU; up to 16 samples
Visual history	Two cameras; four frames each; 2×2 regions
Visual representation / dropout	Differences from latest frame / 0.25 per sample
Prefix / compression encoders	Two Transformer / two cross-attention blocks
Injection layers	3, 5, 8, 11, 13, 16; per-layer memory LayerNorm
Gate	Fixed $1 - \exp(-d\Delta/0.16)$
Flow training / sampling	Shared time; no prefix clamp; 10 Euler updates
Action objective	Mean MSE over each sample's valid physical suffix
Auxiliary objective	3D displacement; weight 0.1; Huber threshold 1
Trainable parameters (approx.)	16.88M, including auxiliary head
Training delays	10% zero; otherwise uniform legal 1–10
Optimizer	AdamW; LR 1.4142×10^{-4}; decay 0.1
Schedule / EMA	27,371-update schedule; EMA cap 0.999
Batch / training seed	256 / 42
Validation	Every 500 updates; 1,536 fixed task-balanced windows

Optimization uses BF16 AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, gradient clipping at 1.0, and 5% linear warmup followed by cosine decay. The schedule horizon is 27,371 updates, although this run stops

at 10,000. Frozen image features are cached during training; evaluation encodes live images. EMA updates use $\beta_n = \min(0.999, (1+n)/(10+n))$. Validation uses six batches of 256 fixed task-balanced windows with sampling seed 42 and repeated noise draws. Weight selection minimizes validation action MSE, excluding the auxiliary displacement loss.

The model retains native observation normalization, causal action attention, the 32-dimensional action latent, and the native flow-time distribution. SmolVLA uses noise at native time one; our equations use $\tau = 1 - t_{\text{native}}$, with the corresponding velocity sign reversal. Only the seven physical action coordinates contribute to the action loss. State and visual histories retain their physical timestamps; visual frames are separated by at least 0.1 seconds. Compression queries share access to the masked memory without semantic-slot routing or compression-memory LayerNorm. The auxiliary displacement scales are $(0.0404854, 0.0544757, 0.0573618)$ meters, inherited from training-episode statistics. The output projections use a nonzero initialization scale of 10^{-3}. The control period is 0.05 seconds and the horizon is 50 actions; takeover stride and evaluated delay belong to the rollout protocol, not the learned architecture.

Asynchronous queue implementation. LIBERO evaluation uses ten settling steps and control-step limits of 220/280/300/520 for Spatial/Object/Goal/Long. Source videos provide 10-fps media timestamps; the data pipeline reconstructs 20-Hz control ticks from recorded actions. These preserve simulator ordering rather than measuring sensor transport. Both evaluated methods use ten native Euler steps; Naive Async omits history and prefix conditioning.

A zero-delay bootstrap query initializes the queue. Subsequent queries use immutable launch snapshots and are scheduled d ticks before each takeover. The single-worker schedule requires $d \leq R$ and $R+d \leq H$ for nonoverlapping queries and adequate queue coverage. Early results wait for their scheduled takeover; late results and results from outdated queue versions are rejected. At takeover, the generated suffix atomically replaces the uncommitted queue tail. The simulated delay controls availability independently of measured inference time and therefore does not establish hardware deadline feasibility.

B Additional Success Results

B.1 Kinetix: Initial Comparison and Calibration Confirmation

Table 5 gives the complete rates underlying Figure 3 and the Naive Async confirmation results omitted from the calibration-focused panel, retaining the two evaluation cohorts separately.

Table 5: Kinetix success (%) across 12 levels, 256 episodes per condition. Each panel uses a separate cohort. Binary-guided TRACE and calibrated TRACE share weights; calibration uses soft guidance and residual attenuation. Bold marks each panel's best positive-delay rates, including ties; zero-delay values are sampler-specific.

Method	$d=0$	$d=1$	$d=2$	$d=3$	$d=4$	Mean 1–4
(a) Initial comparison (seed 271829)						
Naive Async	89.84	77.12	69.60	58.33	48.93	63.49
RTC	89.52	**86.13**	80.73	71.55	60.29	74.67
Hard masking	90.23	85.68	80.50	71.61	60.29	74.52
BID	89.84	85.38	78.29	65.20	55.96	71.21
TRACE (binary)	89.84	85.25	**81.93**	**73.93**	**68.03**	**77.29**
(b) Independent confirmation (seed 271930)						
Naive Async	89.78	78.35	69.14	56.93	48.99	63.35
RTC	89.75	85.42	80.60	70.61	61.75	74.59
TRACE (binary)	89.78	85.38	80.11	73.93	**67.81**	76.81
TRACE (calibrated)	89.78	**85.74**	**82.29**	**74.93**	**67.81**	**77.69**

Table 6 expands the initial Naive Async and binary-guided TRACE comparison from Table 5a (seed 271829), including level-specific averages over $d = 1$–4. These rates describe the configuration preceding deployment calibration.

Table 6: Initial Kinetix success rates (%), seed 271829. Left: mean across 12 levels at each delay. Right: each level's mean over $d = 1$–4. Each condition has 256 episodes; the shared Naive Async/binary-guided TRACE native reference at $d = 0$ is shown once. Bold marks the higher paired rate.

Delay d	Naive Async	TRACE (binary)
0	89.84 (shared)	
1	77.12	**85.25**
2	69.60	**81.93**
3	58.33	**73.93**
4	48.93	**68.03**
Mean 1–4	63.49	**77.29**

Level	Naive Async	TRACE (binary)
Grasp	74.02	**91.41**
Cartpole	73.44	**86.23**
Walker	23.34	**50.88**
Catapult	41.89	**47.56**
Lunar lander	74.61	**90.53**
Half-cheetah	80.18	**85.84**
Swimmer	39.36	**54.79**
Unicycle	74.51	**87.50**
Chain lander	**95.02**	94.43
Catcher	38.38	**73.34**
Trampoline	74.61	**80.37**
Car launch	72.56	**84.57**
Mean	63.49	**77.29**

Table 7 details each level at $d = 4$ and over all positive delays in the initial cohort. At $d = 4$, binary-guided TRACE attains 68.03% success versus 48.93% for Naive Async. Their $d = 1$–4 means are 77.29% and 63.49%, with higher adapted success on eleven of twelve levels; Chain Lander is the exception. Each level contributes the same number of episodes at every delay. Zero-delay Naive Async and binary-guided TRACE records agree on success, return, length, and truncation across all 3,072 pairs.

Table 7: Additional initial Kinetix success rates (%), seed 271829, comparing Naive Async and binary-guided TRACE. The native reference is shown once per level. Bold marks the higher rate in each paired comparison.

	$d = 0$	$d = 4$		Mean $d = 1$–4	
Level	Native	Naive Async	TRACE (binary)	Naive Async	TRACE (binary)
Grasp	96.88	72.66	**91.02**	74.02	**91.41**
Cartpole	100.00	44.92	**60.16**	73.44	**86.23**
Walker	67.97	3.52	**39.06**	23.34	**50.88**
Catapult	69.53	37.50	**41.41**	41.89	**47.56**
Lunar lander	94.53	50.78	**83.20**	74.61	**90.53**
Half-cheetah	87.89	67.58	**84.77**	80.18	**85.84**
Swimmer	91.41	9.38	**25.39**	39.36	**54.79**
Unicycle	98.83	65.63	**80.47**	74.51	**87.50**
Chain lander	92.97	92.58	**96.48**	**95.02**	94.43
Catcher	98.44	20.70	**57.81**	38.38	**73.34**
Trampoline	83.98	68.75	**80.86**	74.61	**80.37**
Car launch	95.70	53.13	**75.78**	72.56	**84.57**
Mean	89.84	48.93	**68.03**	63.49	**77.29**

Table 8 reports per-level results for the independent calibration confirmation (seed 271930). Calibrated TRACE reaches 77.69% mean success, compared with 76.81% for the binary-guided variant, 74.59% for RTC, and 63.35% for Naive Async. Seven levels improve over the preceding adapted configuration, including Catapult (51.95% versus 45.70%) and Walker (53.61% versus 48.34%); Cartpole decreases from 84.28% to 82.23%. The calibration therefore improves the aggregate while retaining level-specific tradeoffs. Calibrated TRACE exceeds Naive Async on all twelve levels in this same confirmation cohort.

Table 8: Kinetix calibration confirmation by level (seed 271930). Rates (%) average $d = 1$–4 with 256 episodes per level and delay. Mean weights all levels equally. All methods use the same confirmation cohort as Table 5b; bold marks the best rate, including ties.

Level	Naive Async	RTC	TRACE (binary)	TRACE (calibrated)
Grasp	77.54	90.92	92.87	**93.75**
Cartpole	68.46	76.46	**84.28**	82.23
Walker	25.68	50.20	48.34	**53.61**
Catapult	39.06	50.88	45.70	**51.95**
Lunar lander	75.10	85.16	**89.16**	88.28
Half-cheetah	80.86	85.06	**85.84**	84.77
Swimmer	38.96	54.10	53.81	**56.93**
Unicycle	76.17	79.69	89.36	**89.84**
Chain lander	94.63	94.53	**95.61**	95.12
Catcher	41.50	70.31	75.88	**76.46**
Trampoline	73.24	77.44	**78.61**	76.86
Car launch	69.04	80.37	82.23	**82.52**
Mean	63.35	74.59	76.81	**77.69**

B.2 LIBERO: Success across Task Suites

Table 9 summarizes success across all 50 official initial states (0–49) per task, averaged over positive delays. Table 1 reports the corresponding rates at each delay.

Table 9: Mean success rates (%) over $d = 1$–3 for all 50 initial states (0–49) per task. Each suite contributes 500 initial states evaluated at all three positive delays, giving 1,500 episodes per method. Overall averages the four suites. Bold marks the higher success rate in each row, including ties.

Initial states	Suite	Naive Async	TRACE
0–49	Spatial	85.27	**86.27**
	Object	**91.20**	**91.20**
	Goal	87.73	**87.87**
	Long	65.67	**66.07**
	Overall	82.47	**82.85**

For each method, suite-level success is the fraction of successful episodes under the official task criterion. All suites contain the same number of tasks and initial states. Aggregate rates average the three positive delays equally and then average across suites. All methods and delays use the same complete set of episode identities, including states 0–4 used in the earlier candidate comparison.

C Additional Ablation Details

C.1 Kinetix: Calibration and Additional Controls

Table 10 expands the four configurations in Table 2 to all eight development candidates, evaluated with seed 424341 and 64 episodes per level and delay. The first six rows share the selected compact checkpoints. Every row retains the temporal adapter and its exponential gate; attenuation additionally applies $\alpha(d) = [0, 0.25, 0.5, 0.75, 1]$ for $d = 0, \ldots, 4$. The soft-guidance row without attenuation retains the adapter, distinguishing it from the unadapted RTC baseline. Development success is used for selection; the complete candidate grid is not repeated on the confirmation cohort.

Table 10: Kinetix development candidates. Success rates (%) use twelve levels and 64 episodes per condition. The first four rows form the deployment comparison in Table 2; the final two use separately trained prefix-regularized adapters. Bold marks the highest rate, including ties.

Variant	$d = 1$	Mean $d = 1$–4
Binary guidance	84.38	75.94
Soft guidance	83.20	75.78
Binary guidance + attenuation	84.11	76.76
Soft guidance + attenuation	**87.37**	**78.06**
Delay-adaptive guidance + attenuation	**87.37**	77.54
Frozen-guide adaptive variant + attenuation	87.24	77.18
Prefix regularization + binary guidance	84.11	75.39
Prefix regularization + adaptive guidance + attenuation	85.55	76.14

Delay-adaptive guidance uses soft weights at $d = 1, 2$ and binary weights at $d = 3, 4$, together with residual attenuation. Its mean success is below that of soft guidance with attenuation throughout. The frozen-guide variant preserves this adaptive schedule and adapted sampling velocity, but computes the guidance endpoint and its Jacobian from the frozen backbone. Comparing it with the delay-adaptive variant changes the guidance source alone; their mean rates are 77.18% and 77.54%, respectively. This development result favors differentiating through the adapted predictor, without establishing a separately confirmed component effect.

The prefix-regularized candidates add a penalty on the adapted velocity's deviation from the frozen field at committed positions. They retain the same training data, architecture, update budget, and training seed as the reference, but require new checkpoints. The binary-guided candidate is compared with the first row, and the attenuated adaptive candidate with its counterpart in the fifth row. Neither improves mean success in these matched comparisons. The selected configuration therefore retains suffix regularization without this additional prefix penalty. None of these controls removes history or isolates the temporal encoder's architecture.

C.2 LIBERO: Training Sensitivity and Offline Diagnostics

Table 11 compares standard TRACE training with two variants at matched update budgets, using five initial states over $d = 1$–4. Window filtering requires support for the complete delay range and a minimum executable suffix; takeover weighting emphasizes the first actions after takeover. Standard training retains the original windows and uniform suffix weighting. Neither variant improves mean success; takeover weighting slightly improves smaller-delay results while reducing larger-delay results.

The standard-training results were reused from an earlier evaluation with different source revisions and batch size, and these scores subsequently selected that recipe. This is an exploratory sensitivity study rather than an isolated component-effect estimate.

Current training-sensitivity settings. The standard configuration and both variants use EMA weights at update 6,000, a 27,371-update learning-rate schedule, batch size 256, visual differences, visual-history dropout 0.25, weight decay 0.1, and the displacement objective. The reference checkpoint was selected by validation loss; variant checkpoints are fixed final-update weights. The

Table 11: LIBERO training sensitivity at takeover stride $R = 5$. Entries are mean success rates (%) over the indicated delays, using five initial states per task and 200 episodes per delay. The reused standard-training evaluation differs in recorded source revision and batch size.

Training variant	$d = 1$–2	$d = 3$–4	$d = 1$–4
TRACE (standard training)	82.25	84.25	83.25
Window filtering	82.00	81.75	81.88
Takeover weighting	83.00	82.75	82.88

window-filtering variant requires at least five valid actions after takeover for every training delay $d = 0, \ldots, 10$, reducing anchors from 218,971 to 199,973. This changes both delay coverage and representation of demonstration endings. The takeover-weighted objective assigns half its weight to the first five suffix actions and half to the remaining valid suffix, with the available segment receiving full weight when the other is empty.

Each arm has 1,000 completed rollouts including the zero-delay diagnostic. All paired initial observations and noise seeds match. Evaluation uses the system-default EGL protocol and the same frozen backbone and processor, but the reused reference uses batch size eight and the new variants use five; recorded source revisions also differ. A single anomalous initial observation was repaired under the fixed episode identity and archived before the completed comparison. Table 12 gives the reference and variant success rates for each delay group. These evaluation states subsequently selected the standard TRACE recipe among the fixed candidates and therefore do not constitute independent post-selection evidence.

Table 12: Standard TRACE and variant success rates (%) for the training-sensitivity comparison. Each entry averages the same initial states over the indicated delays.

Training variant	Delays	Standard TRACE	Variant
Window filtering	1–2	82.25	82.00
Window filtering	3–4	84.25	81.75
Window filtering	1–4	83.25	81.88
Takeover weighting	1–2	82.25	83.00
Takeover weighting	3–4	84.25	82.75
Takeover weighting	1–4	83.25	82.88

Capacity and regularization are validation diagnostics. Table 13 reports best and terminal EMA validation flow-MSE for runs observed through 10,000 updates. The compact model reduces both width and depth. The regularized model changes both visual-history dropout and weight decay. Neither intervention isolates a single factor, and neither passed the predeclared validation stability screen. The regularized checkpoint was later used in separate closed-loop selection; these validation curves alone do not establish improved task success.

Table 13: Exploratory validation diagnostics. The last column is terminal minus best validation MSE, not a paired test effect.

Configuration	Best MSE	Update	MSE at 10,000	Increase
Absolute features	0.07764	1,500	0.08587	+0.00823
Visual differences	0.07604	5,000	0.07836	+0.00232
Reduced width + depth	0.07671	6,000	0.07830	+0.00159
Dropout + weight decay	0.07639	6,000	0.07764	+0.00124

Visual representation at matched training budgets. We compare absolute visual-history features with differences from each camera's latest frame, holding the adapter architecture and optimization settings fixed. The recorded diagnostics use EMA checkpoints at matched update counts and the same 1,536 test windows, delays, and noise draws. Table 14 reports lower suffix flow-MSE for visual differences at both budgets, with the reduction concentrated in LIBERO-Goal. These are offline prediction diagnostics; their effect on closed-loop success is not isolated by this comparison.

Table 14: Visual representation at matched training budgets. Values are normalized suffix flow-MSE on the recorded paired test windows; lower is better. The MSE difference is visual differences minus absolute features.

Updates	Absolute	Differences	Δ MSE
5,000	0.08806	0.07635	-0.01171
10,000	0.08998	0.07905	-0.01093

Both representation arms use no visual-history dropout and weight decay 0.01, differing from the standard TRACE training recipe. Checkpoint budgets are matched rather than independently selected for each architecture. The test episodes are held out from adapter training but were revisited during development, including continuation decisions, and the released backbone has seen the source dataset. Saved reports retain aggregate statistics but omit individual window losses, so these results are audited from the recorded reports and configuration provenance. The visual-difference test MSE rises from 0.07635 at update 5,000 to 0.07905 at update 10,000; temporal differencing therefore does not eliminate late-training degradation in this diagnostic.

Evidence not used for component claims. The history-content controls were paused before training, and raw-versus-EMA closed-loop evidence is incomplete. Semantic query routing, compression-memory normalization, and inference-time residual scaling supply exploratory diagnostics with differing budgets or interventions; they are not substituted for matched retrained component ablations. These incomplete or unmatched comparisons are excluded from the reported component claims.

D Evidence and Reproduction

The Kinetix results separate initial evaluation, calibration development, and independent confirmation, with their settings and selection procedure specified in Appendix A.1 and numerical results in Appendices B.1 and C. The report-level audit verifies recorded hashes and weight identities, reconstructs integer success counts from the reported rates and episode denominators, and checks aggregate consistency. Original episode journals were not reprocessed for this manuscript audit; its checks use the completed experiment reports and recorded independent recalculations. Initial evaluation and refinement used distinct source revisions. The confirmation results include Naive Async and all three guided methods from the same cohort; no baseline rates are borrowed from the initial evaluation.

LIBERO uses the settings in Table 4 and the training procedure in Appendix A.2. The selected EMA weights achieve validation action MSE 0.07639175 at update 6,000, before training ends at update 10,000. Their serialized SHA256 is

```
9b8c9661899586a5b2ad4fdd163a485b12
c35267a7826370d163e6f391485cd1
```

The frozen-backbone tensor audit records identical initial and final hashes,

```
caa3211efeb010599043f8de3b82d6caa
047b949fe4b0abef8d0ecd7f4ccbe20
```

with zero backbone updates and no backbone gradients. Tensor audit hashes describe parameter values and are distinct from serialized-file hashes.

The complete LIBERO evaluation contains 20,000 episodes: two methods (TRACE and Naive Async), five delays ($d = 0$–4), 40 tasks, and 50 initial states per task. An independent CPU analysis recomputes success counts and verifies all 10,000 method pairs by task, initial-state identity, initial-observation hash, and sampling-noise seed. The manuscript reports delays $d = 0$–3 from this cohort. Both methods share the frozen model and processor, evaluation source revision, runtime configuration, and batch size ten. Rendering uses system-default EGL with `MUJOCO_GL=egl`; all paired initial-image hashes match. Observation-hash agreement complements the renderer check and does not imply identical subsequent trajectories.

Before final pairing, two TRACE records had initial observations inconsistent with the common image hash under their fixed task and state identities. The affected conditions were Spatial task 3, initial state 5, delay 4, and Spatial task 9, initial state 17, delay 3. Each was rerun once under the original configuration and source snapshot. Acceptance used initial-observation and noise agreement, independently of success. Original and replacement records remain archived in the evaluation provenance. At zero delay, the recorded Naive Async and TRACE runs succeed in 1,652 and 1,650 of 2,000 episodes, respectively; full action hashes agree in 1,920 pairs. The single native-reference row reports the former run and does not replace the latter measurements. We therefore distinguish the policy's exact bypass property from reproducibility of a full simulator trajectory.

Naive Async comparisons use matched cohorts within each benchmark; Kinetix confirmation also compares the calibrated configuration with RTC and its binary-guided predecessor. Section 7 distinguishes fixed-checkpoint development calibration from independent confirmation, and Appendix C qualifies the additional Kinetix controls and LIBERO training diagnostics. Incomplete history, EMA, and additional-stride comparisons are excluded.