LEARN THE TASK, NOT THE SCHEDULE: REDUCING SPURIOUS CROSS-ARM COUPLING IN BIMANUAL IMITATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

A bimanual demonstration specifies both *what* the robot should do and one particular schedule for *when* each arm should act. Standard imitation learning does not distinguish task-required cross-arm constraints from incidental scheduling choices, and may therefore exploit opposite-arm progress as a shortcut cue, creating *spurious cross-arm coupling* even where relative timing is irrelevant for the current task phase. We introduce Constraint-preserving Temporal Recombination (CTR), a training-time data intervention that preserves recorded arm-local sensorimotor content while retiming the corresponding segments to realize admissible schedules that preserve task-required cross-arm constraints. We use CTR to train Diffusion Policy, Action Chunking with Transformers, and $\pi_{0.5}$, and evaluate them on four simulated and four real-robot tasks. Across six real-robot bimanual progress-state initializations, CTR raises worst-state success from 56% for the strongest non-CTR schedule mixture to 76%. With $\pi_{0.5}$ on an Independent task, a 4 s pause transfers 2.56 s of delay to the opposite arm under the Concurrent schedule distribution, but only 0.25 s with CTR. On real robots, CTR violates task constraints in only 0–4% of trials, versus 36–98% for narrow-schedule baselines.

1 INTRODUCTION

In a bimanual demonstration, the two arms follow one of possibly many admissible schedules. If each arm independently picks an object and the demonstrator does so sequentially, the recorded order is a choice, not a task requirement; likewise, a drawer must be opened before an object is inserted, but not before the object is grasped. Expressive imitation policies such as Diffusion Policy (DP), Action Chunking with Transformers (ACT), and vision-language-action models including π_0, $\pi_{0.5}$, and OpenVLA (Chi et al., 2025; Zhao et al., 2023; Black et al., 2025b;a; Kim et al., 2025) cannot tell these cases apart from the demonstrations alone. When demonstrations cover only a few schedules, one arm's behavior can come to depend on the other's progress even where their subtasks are independently schedulable; we call this *spurious cross-arm coupling*.

The narrow schedule support that creates this ambiguity appears common in existing datasets. Our audit of four public bimanual datasets—RoboTwin 2.0, RoboCOIN, DexMimicGen, and BiCoord (Chen et al., 2026; Wu et al., 2025; Jiang et al., 2025b; Peng et al., 2026)—finds that 266 of 716 tasks (37.2%) are reschedulable, yet only 3.4% of these exhibit multiple observed admissible schedules (Appendix A). This limited coverage motivates learning cross-arm dependence that reflects task constraints rather than the particular schedules demonstrated. We call this *selective cross-arm dependence*: in independently schedulable phases, delaying one arm should not make the other wait, whereas during insertion, support, or shared-workspace occupancy, the policy must respond to the opposite arm. More demonstrations with the same narrow schedule support cannot resolve this ambiguity (Section 3.3).

We propose **Constraint-preserving Temporal Recombination (CTR)**, a training-time data intervention that broadens schedule support without new demonstrations. CTR segments each arm's recorded stream and retimes the segments into alternative admissible schedules while preserving task-required cross-arm constraints, pairing the same arm-local sensorimotor content with varied opposite-arm progress.

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 evaluate CTR with three policy families (DP, ACT, and $\pi_{0.5}$) on four simulated and four real-robot tasks. CTR-trained policies achieve higher default-condition success rates than the same architectures trained with serial schedules in 19 of 24 model–task comparisons, and CTR raises real-robot worst-case progress-state success from 56% to 76%. Pausing one arm for 4 s delays the other arm's completion by 2.56 s with $\pi_{0.5}$ trained on concurrent-start schedules, but by only 0.25 s with CTR. Required coordination is retained: under real-robot constraint interventions, CTR-trained $\pi_{0.5}$ violates task constraints in 0% and 4% of trials, versus 98% and 36% for narrow-schedule baselines.

Our contributions are: (1) identifying spurious cross-arm coupling under narrow schedule support and formalizing selective cross-arm dependence as the desired behavior; (2) CTR, which broadens admissible schedule support while preserving arm-local sensorimotor content and task-required cross-arm constraints; and (3) simulated and real-robot evidence across three policy families that CTR reduces spurious coupling without sacrificing required coordination.

2 Related work

Bimanual policies and cross-arm dependence. Bimanual imitation policies model cross-arm dependence through joint action chunks in ACT (Zhao et al., 2023) or explicit attention and synchronization in InterACT (Lee et al., 2025). BUDS structures this dependence through acting and stabilizing roles (Grannen et al., 2023). To support independence alongside coordination, the Decoupled Interaction Framework combines arm-specific policies with selective information exchange (Jiang et al., 2025a), while SkillVLA reuses single-arm skills across novel left–right pairings (Zhai et al., 2026). These approaches structure dependence at the model or skill level; our focus is the training distribution, where narrow schedule support can make opposite-arm progress predictive even when it is irrelevant to the current arm-local subtask.

Causal confusion and counterfactual augmentation. This mismatch between predictiveness and task relevance connects schedule-induced coupling to causal confusion in imitation learning (de Haan et al., 2019). A related data-level strategy is counterfactual augmentation: CoDA recombines locally factored transitions (Pitis et al., 2020), and CAIAC exchanges state components estimated to be unaffected by an action (Armengol Urpí et al., 2024). RoCoDA brings this principle to visual robot imitation, including bimanual tasks, by recombining state partitions under phase-specific causal structure (Ameperosa et al., 2025). Here, the intervention of interest is relative arm timing while the recorded arm-local behavior remains fixed. Because relative timing can also encode required coordination, this intervention must vary progress pairings without violating task-required cross-arm constraints.

Constraint-aware generation and temporal recombination. Demonstration generation provides tools for this constrained recombination: MimicGen adapts object-centric segments to new contexts (Mandlekar et al., 2023), and DexMimicGen adds asynchronous per-arm execution with synchronization and ordering constraints (Jiang et al., 2025b). TriManPolicy more directly addresses interface-induced waiting by retiming fixed-duration sensorimotor segments under dependency and arm-resource constraints to minimize demonstration makespan (Zhao et al., 2026). However, selecting an efficient schedule does not by itself establish robustness to other admissible schedules. CTR therefore samples alternative schedules for the same recorded arm-local sensorimotor content under annotated task constraints, broadening progress-state support without changing the policy architecture or deployment procedure. To distinguish selective cross-arm dependence from indiscriminate decoupling, we combine held-out progress tests and paired opposite-arm pauses with task-constraint interventions.

3 Problem formulation: spurious coupling versus selective dependence

3.1 Bimanual imitation learning

We learn a joint bimanual policy from demonstrations $\mathcal{D} = \{\tau_n\}_{n=1}^{N}$ with observations o_t and joint actions $a_t = (a_t^L, a_t^R)$. For a base learner m, let h_t contain the current observation, available history, and task instruction when provided; $\pi_{\theta_m}(A_t \mid h_t)$ predicts an H_m-step action chunk $A_t =$

$(a_t, \ldots, a_{t+H_m-1})$. Let $P_{\mathcal{D}}$ be the empirical trajectory distribution and $\mathcal{W}_m(\tau)$ the learner's distribution over valid training windows $w = (h_t, A_t)$ from τ. We minimize the original training loss:

$$\theta^{\star}_{m,\mathrm{BC}} = \arg\min_{\theta_m} \; \mathbb{E}_{\tau \sim P_{\mathcal{D}}} \, \mathbb{E}_{w \sim \mathcal{W}_m(\tau)} \big[\mathcal{L}_m(\theta_m; w) \big]. \tag{1}$$

Here $\mathcal{L}_m$ is the base learner's per-window loss, including model-specific action and auxiliary terms; the expectations average it over empirical training windows. These windows pair both arms on the recorded timeline, so the training data also encode the demonstrator schedule.

3.2 Spurious cross-arm coupling

A demonstration records one demonstrator schedule although the task may permit other admissible schedules with the same arm-local sensorimotor content. We call this limited support over admissible schedules *narrow schedule support*. Under narrow schedule support, imitation may treat opposite-arm progress as a shortcut, producing *spurious cross-arm coupling* when the policy responds to an incidental timing relation.

For fixed arm-local sensorimotor content in τ, let $\mathcal{S}(\tau)$ contain schedules respecting within-arm order, $V_{\mathrm{task}}(\tau, s)$ encode physical and semantic validity, and $\mathcal{R}(\tau)$ contain candidate cross-arm relations observed in τ. Define

$$\Omega^{\star}(\tau) = \{ s \in \mathcal{S}(\tau) : V_{\mathrm{task}}(\tau, s) = 1 \}, \tag{2}$$

$$\mathcal{R}^{\star}_{\mathrm{req}}(\tau) = \{ r \in \mathcal{R}(\tau) : r(s) = 1 \; \forall s \in \Omega^{\star}(\tau) \}. \tag{3}$$

Relations in $\mathcal{R}^{\star}_{\mathrm{req}}(\tau)$ are *task-required cross-arm constraints*; a candidate relation that changes under some admissible schedule is *reschedulable*. Relations specify properties such as order or non-overlap, not exact delays: a required order can allow timing slack. This is a property of the task, not the learned policy.

Selective cross-arm dependence. Let x_i be arm i's local task state, p_i its within-segment progress, and c the shared conditions relevant to coordination. We call $z = (p_L, p_R)$ the *bimanual progress state*. In an independently schedulable phase, the arm-i marginal should be approximately invariant to an admissible change in opposite-arm progress p_j $(j \neq i)$:

$$\pi^{i}_{\theta_m}(\cdot \mid x_i, p_i, c, p_j) \approx \pi^{i}_{\theta_m}(\cdot \mid x_i, p_i, c, p'_j). \tag{4}$$

In a coupled phase, shared task progress p_C cannot be split into independently schedulable arm-local progress; a change in a prerequisite, support, or workspace condition can instead require a different response. We test both sides through phase-conditioned sensitivity and task-constraint interventions.

3.3 What is and is not identifiable

If every demonstration places one arm's segment before the other's, that order may be a prerequisite or a habit. Under an idealized fixed schedule, $p_j = h(x_i, p_i, c)$ almost surely; dependent and independent policies can therefore agree on observed bimanual progress states but differ at unobserved admissible states. More demonstrations drawn from the same narrow schedule support cannot distinguish them.

Determining which alternatives are safe requires task knowledge beyond the observed timing. CTR supplies it through task-constraint annotations during data construction; the deployed policy does not receive these annotations.

4 Constraint-Preserving Temporal Recombination

Recollecting demonstrations under different schedules could also change the arm-local trajectories. CTR instead preserves recorded arm-local sensorimotor content while retiming segments to realize alternative admissible schedules, making the schedule the intended intervention and broadening schedule support without new demonstrations (Figure 1).

4.1 TASK-SPECIFIC AUTOMATIC SEGMENTATION AND CONSTRAINT EXTRACTION

Let $\tau = \{(o_t, a_t)\}_{t=1}^{T}$ be a recorded demonstration with per-arm streams τ^i, $i \in \{L, R\}$. At task setup time, GPT-6 Astra (OpenAI, 2026) generates a task-specific video-processing pipeline $\mathcal{P}_\xi$. The generated program detects task-relevant interactions from the available visual and robot evidence, decomposes each arm-local stream into K_i ordered segments $\sigma_{i,1} \prec \cdots \prec \sigma_{i,K_i}$ tiling the demonstration timeline, $\sigma_{i,k} = [u_{i,k}, v_{i,k}]$ with duration $\ell_{i,k} = v_{i,k} - u_{i,k}$, and emits the task-required cross-arm relations used for schedule construction and validation. We write $\mathcal{R}_{\text{req}}(\tau)$ for these pipeline-produced constraints on segment pairs; they approximate $\mathcal{R}^\star_{\text{req}}(\tau)$ rather than being inferred from the recorded schedule alone.

A schedule $s = \{s_{i,k}\}$ places each segment at $\Sigma_{i,k}(s) = [s_{i,k},\ s_{i,k} + \ell_{i,k}]$, preserving its content and duration. The implemented set $\Omega(\tau)$ keeps within-arm order and freezes the recorded relative offsets of segment pairs governed by constraints in $\mathcal{R}_{\text{req}}(\tau)$. Intended as a conservative subset of $\Omega^\star(\tau)$, it may exclude admissible timing variations under those constraints while allowing reschedulable blocks to become serial or overlapping.

4.2 RETIME SEGMENTS WHILE PRESERVING ARM-LOCAL SENSORIMOTOR CONTENT

For $s \in \Omega(\tau)$, the source-time map ϕ_i^s advances through arm i's active segments and holds during inserted gaps (Equation 16, Appendix F). The recombination $\widetilde{\tau} = \mathcal{C}(\tau, \phi_L^s, \phi_R^s)$ reads arm-local streams at their respective source times and reconstructs the shared main view (Section 4.3). For every active segment and offset $u \in [0, \ell_{i,k}]$, it preserves recorded arm-local sensorimotor content:

$$(\widetilde{o}^i_{s_{i,k}+u}, \widetilde{a}^i_{s_{i,k}+u}) = (o^i_{u_{i,k}+u}, a^i_{u_{i,k}+u}). \tag{5}$$

CTR keeps the empirical demonstration law $\tau \sim P_{\mathcal{D}}$ but replaces the recorded schedule law δ_{s_τ} with $s \sim q(\cdot \mid \tau)$ supported on $\Omega(\tau)$. It forms $\widetilde{\tau} = \mathcal{C}(\tau, \phi_L^s, \phi_R^s)$ and extracts training windows. The implementation samples admissible reschedulable-block offsets uniformly, pairing the same arm-local sensorimotor content with varied opposite-arm progress.

Let M_s mask only action-supervision coordinates belonging to synthetic waits; model-specific auxiliary terms remain outside the mask normalization. We minimize the CTR training loss:

$$\theta^\star_{m,\text{CTR}} = \arg\min_{\theta_m} \ \mathbb{E}_{\tau\sim P_{\mathcal{D}}}\, \mathbb{E}_{s\sim q(\cdot|\tau)}\, \mathbb{E}_{w\sim \mathcal{W}_m(\mathcal{C}(\tau,\phi_L^s,\phi_R^s))} \big[\mathcal{L}_m(\theta_m; w, M_s)\big], \tag{6}$$

Equation 19 specifies the masked action term. If $q(\cdot \mid \tau) = \delta_{s_\tau}$, $\mathcal{C}(\tau, \phi_L^{s_\tau}, \phi_R^{s_\tau}) = \tau$, and $M_{s_\tau} = 1$, Equation 6 reduces to the original training objective in Equation 1.

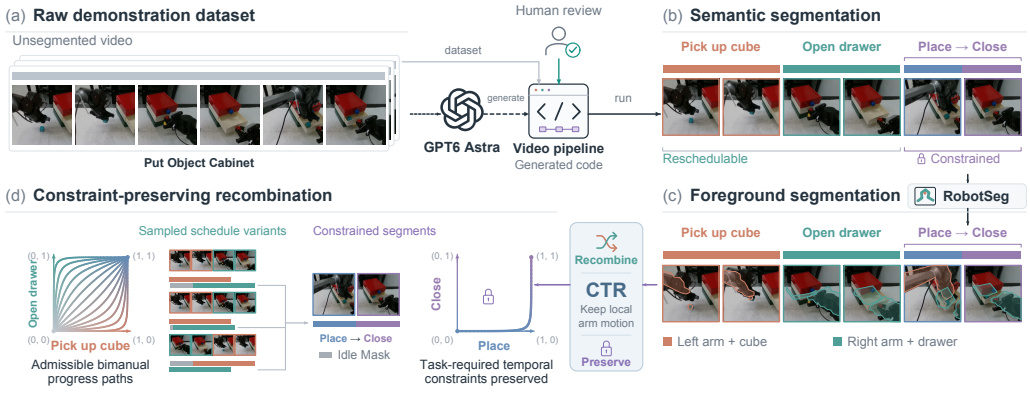


Figure 1: CTR data-processing pipeline. Foreground segmentation supports observation composition, and temporal recombination realizes alternative admissible schedules while preserving arm-local sensorimotor content and task-required cross-arm constraints. The displayed masks and sampled schedules are schematic.

4.3 CTR ON RECORDED REAL-ROBOT DATA: TWO IMPLEMENTATION ISSUES

For the main camera, foreground segmentation extracts the two arms and task-relevant objects, and we composite them at the recombined source times $\phi_i^s(t)$ over the scene background, approximating both arms executing the recombined schedule concurrently (Equation 17, Appendix F). For supervision, an IdleMask zeroes the imitation loss on coordinates and timesteps whose waiting was inserted by recombination, while required holding, active support, and terminal holding remain supervised (Equations 18–19, Appendix F). Preserving arm-local streams does not by itself guarantee that every recombined joint scene is physically executable, nor that a composed observation is identical to a concurrently captured real observation. We therefore apply task-specific geometric, contact, and observation-consistency checks as conservative filters; Appendix G details these audits and their remaining limitations.

5 EXPERIMENTS

We first diagnose spurious cross-arm coupling at the action and rollout levels, then test whether CTR reduces it while preserving task-required dependence, and finally ask whether the resulting robustness extends beyond training support.

5.1 EXPERIMENTAL SETUP

Tasks, models, and evaluation. We evaluate DP, ACT, and $\pi_{0.5}$ on four simulated and four real-robot tasks spanning **Independent**, **Shared-Workspace Exclusion**, **Coordination**, and **Prerequisite** relations (Figure 2). Appendix B specifies the tasks and progress-state initializations, Appendix C gives the training and simulation-evaluation configurations, and Appendix E defines the metrics and aggregation conventions. All success rates reported in this paper are computed from 50 evaluation trials.

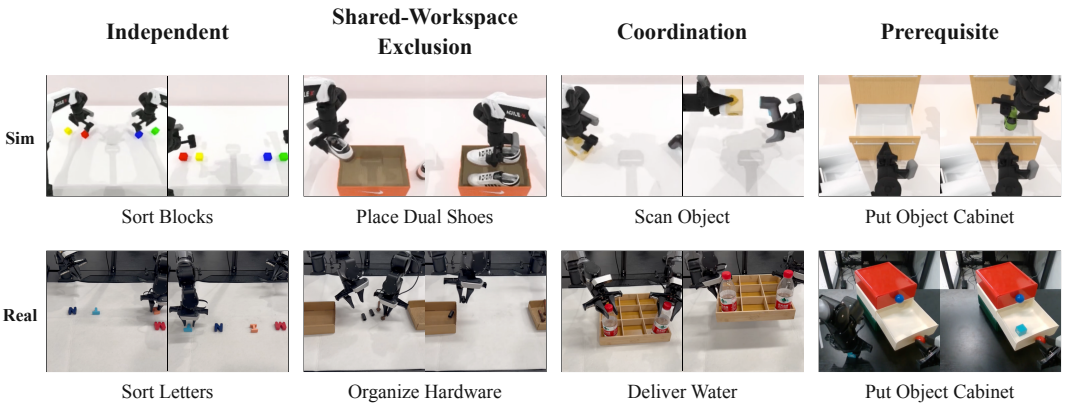


Figure 2: The eight evaluation tasks span Independent, Shared-Workspace Exclusion, Coordination, and Prerequisite relations, testing both invariance where relative timing is irrelevant and responsiveness where cross-arm constraints matter. Columns group tasks by relation type; the top and bottom rows show simulated and real-robot tasks, respectively, with two representative frames per task.

Training schedule distributions. To vary schedule support while preserving arm-local sensorimotor content and annotated task-required constraints, we compare five distributions over independently schedulable regions: $\boldsymbol{L \rightarrow R}$ (one serial order), **Serial** (both serial orders), **Concurrent**, **Mixed** (both serial orders plus Concurrent), and **CTR** (relative timing sampled across the admissible range). Serial, Concurrent, Mixed, and CTR share the main-camera visual-synthesis pipeline in Section 4.3; the original-data $L \rightarrow R$ condition retains recorded main-camera observations. Only CTR uses IdleMask, because its recombination can insert artificial waits. Section 5.3 controls for both visual synthesis and action-supervision masking. Figure 3 shows empirical onset-difference densities, and Appendices D and F detail distribution construction and implementation.

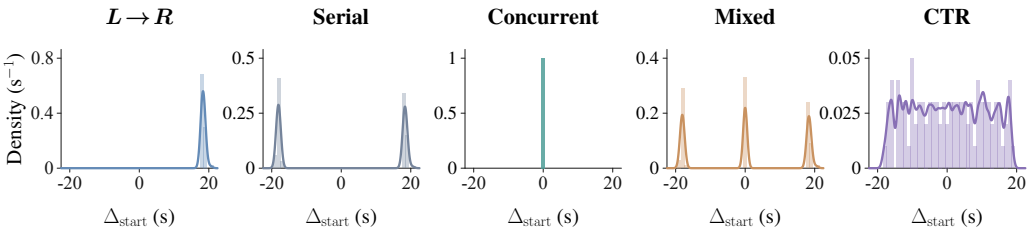


Figure 3: The five training distributions vary relative arm timing: the baselines concentrate on one or more canonical schedules, whereas CTR covers a broader range of admissible timings. Histograms show onset-difference densities from Sort Blocks simulation trajectories, with $\Delta_{\text{start}} = t_R^{\text{start}} - t_L^{\text{start}} > 0$ indicating that the left arm starts first; vertical density scales differ across panels. Curves show smoothed density estimates except for Concurrent; smoothing-induced tails should not be interpreted as additional observed support. Appendix D gives construction and measurement details.

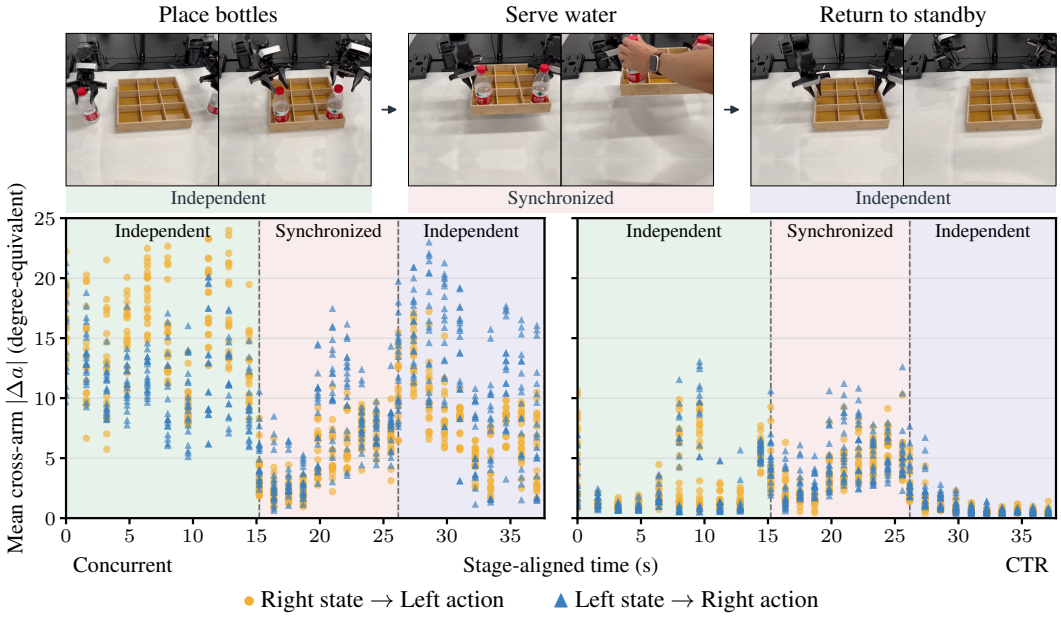


Figure 4: Phase-conditioned action sensitivity of ACT on Deliver Water: the change in one arm's predicted actions when the opposite-arm state changes while its own local state is held fixed. CTR (right) reduces sensitivity during *Place Bottles* and *Return to Standby* while retaining responsiveness during *Serve Water*; Concurrent (left) remains strongly sensitive even in the independent phases.

5.2 Diagnosing spurious cross-arm coupling

Action-level sensitivity in independently schedulable phases. The invariance in Equation 4 motivates a phase-conditioned action-sensitivity probe (Appendix E.3) on Deliver Water, whose independent *Place Bottles* and *Return to Standby* phases surround the synchronized *Serve Water* phase. Averaging matched interventions in both directions, Concurrent ACT yields $(S_{\text{place}}, S_{\text{serve}}, S_{\text{return}}) = (13.28, 5.82, 9.77)$, with greater sensitivity in both independent phases than during synchronization (Figure 4, left); CTR ACT yields $(2.79, 4.19, 1.07)$, suppressing independent-phase sensitivity while retaining a *Serve Water* response larger than its own independent-phase responses (right). CTR therefore reduces sensitivity selectively in the independent phases while retaining a stronger response during synchronization.

From action dependence to progress-state brittleness. If a policy uses opposite-arm progress as a shortcut, a familiar local state paired with a different admissible opposite-arm state can elicit

Table 1: Bimanual progress-state robustness for $\pi_{0.5}$. Each measured-state cell reports SR (%) / T_{Pen} (s). When SR=0, T_{Pen} is undefined and shown as "–" because its failure penalty depends on the cell's mean successful completion time. Cell tint is determined solely by SR, with darker blue indicating higher success rates. Worst SR is the minimum over the six states. Appendix E specifies the penalty.

Task	Data	C00	Cmid	L-lead	R-lead	CL	CR	Worst SR
Put Object Cabinet (sim)	$L \to R$	40 / 16.4	56 / 3.8	74 / 6.4	0 / –	**60** / 9.1	2 / 8.1	0
	Serial	36 / 16.9	90 / 3.5	**76** / 6.4	30 / 8.8	52 / 9.3	44 / 10.6	30
	Concurrent	22 / 12.1	98 / 2.9	0 / –	0 / –	0 / –	0 / –	0
	Mixed	54 / 10.1	**100** / 3.0	64 / 7.0	50 / 7.4	48 / 10.1	80 / 7.8	48
	CTR (ours)	**66** / 11.0	**100** / 3.2	72 / 6.1	**62** / 11.3	56 / 8.7	**96** / 6.6	**56**
Sort Letters (real)	$L \to R$	90 / 32.1	82 / 29.1	36 / 48.7	72 / 39.3	92 / 22.2	98 / 18.0	36
	Serial	92 / 32.7	0 / –	18 / 54.6	50 / 46.8	96 / 21.0	88 / 25.8	0
	Concurrent	92 / 19.2	92 / 12.6	54 / 34.9	52 / 35.9	42 / 42.4	28 / 48.6	28
	Mixed	74 / 39.0	**94** / 13.5	58 / 35.6	56 / 34.3	96 / 20.4	74 / 29.3	56
	CTR (ours)	**94** / 21.1	**94** / 17.4	**76** / 26.6	**96** / 18.9	**100** / 16.4	**100** / 16.2	**76**

Table 2: Implementation-control SRs (%) on real-robot Sort Letters and simulated Scan Object, testing masking and visual synthesis.

Training variant	Sort Letters	Scan Object
CTR	**94**	**70**
CTR w/o IdleMask	92	56
$L \to R$	90	68
$L \to R$ + visual synthesis	90	62

Table 5: Sort Letters SRs (%) for $\pi_{0.5}$ at initial states and in a region withheld from CTR-hole training.

Data	Initial-state SR (%)	Unseen SR (%)
Mixed	74	58
CTR-hole	92	74
CTR-full	94	76

an inappropriate action and compromise task completion. We test this behavioral prediction with $\pi_{0.5}$ on Put Object Cabinet in simulation and Sort Letters on the real robot. Each task uses six task-defined initial states in independently schedulable regions: nominal (C00), matched-progress (Cmid), left-leading, right-leading, and two task-specific ready/completion-side states (CL/CR); these are not interpolated robot joint configurations. Table 1 reports success rate (SR) and failure-penalized completion time T_{Pen}. On Sort Letters, $L \to R$, Serial, and Concurrent achieve 90–92% SR from C00, yet their worst-state SRs fall to 36%, 0%, and 28%, respectively. On Put Object Cabinet, Concurrent reaches 98% at Cmid but 0% at all four asymmetric states. Narrow serial and concurrent support can thus both leave policies brittle across admissible bimanual progress states. These rollout results extend the action-level probe to closed-loop behavior across tasks and models.

5.3 CTR RESTORES ROBUSTNESS WITH SELECTIVE CROSS-ARM DEPENDENCE

Recovery under broader schedule support. Because brittleness appears under multiple narrow schedules, we next broaden their support rather than select another fixed order. In the same Sort Letters stress test, Mixed raises worst-state SR to 56%, and CTR raises it to 76% (Table 1). Relative to Mixed, CTR improves R-lead from 56% to 96% and CR from 74% to 100%, while Cmid remains at 94%. On Put Object Cabinet, worst-state SR similarly increases from 48% with Mixed to 56% with CTR. Broader admissible support thus improves worst-case robustness over a mixture limited to canonical schedules.

Implementation controls. Since CTR also introduces main-camera visual synthesis and action-supervision masking for synthetic waits, we test these implementation choices separately (Table 2). Applying CTR's main-camera segmentation and compositing to $L \to R$ without retiming leaves Sort Letters SR at 90% and changes Scan Object SR from 68% to 62%, so visual synthesis alone does not explain an improvement on these tasks. Removing only IdleMask from CTR reduces SR from 94% to 92% and from 70% to 56%, respectively, showing that action-supervision masking for synthetic waits also contributes, particularly on Scan Object.

Table 3: Task-constraint interventions with $\pi_{0.5}$ on two real-robot tasks. Put Object Cabinet compares $L \rightarrow R$ with CTR; Organize Hardware compares Concurrent with CTR. Lower is better for both violation rate and release latency; CTR lowers violations but has longer release latencies.

Task	Baseline	Violation (%)		Release latency (s)	
		Baseline	CTR	Baseline	CTR
Put Object Cabinet (real)	$L \rightarrow R$	98	0	0.01	0.27
Organize Hardware	Concurrent	36	4	0.09	0.15

Behavior under task-required constraints. We directly test required cross-arm dependence with $\pi_{0.5}$ on two real-robot tasks. In Put Object Cabinet, we close the drawer just before the left arm inserts the object; attempting insertion while the drawer is closed is a violation, and the correct response is to wait until it is reopened. CTR waits in every trial, whereas the $L \rightarrow R$ baseline, which never observes waiting in front of the drawer, violates the constraint in 98% of trials. In Organize Hardware, we define a shared workspace, where entry by one arm while the other occupies the region is a violation; CTR reduces violations from 36% to 4% against the Concurrent baseline (Table 3). Release latency increases from 0.01 to 0.27 s and from 0.09 to 0.15 s, respectively. CTR reduces constraint violations at the cost of longer release latency. Together, the Deliver Water probe, where CTR retains greater sensitivity during *Serve Water* than in the independent phases, and these constraint interventions show phase-dependent sensitivity: reduced response when relative timing is irrelevant and retained response when coordination is required.

Default-condition task performance. We next check whether the benefit extends to ordinary task execution across all eight tasks and three policy families. Table 4 compares CTR with the Serial baseline and shows higher task-averaged SR for each model family, in both simulation and real-robot evaluation. CTR improves success rates in 19 of 24 task–model comparisons, but does not improve performance on Place Dual Shoes.

Table 4: Default-condition SRs (%) for DP, ACT, and $\pi_{0.5}$ trained on Serial and CTR data. The first four rows are simulated tasks and the last four are real-robot tasks. Each entry is task-level SR.

Task	DP		ACT		$\pi_{0.5}$	
	Serial	CTR	Serial	CTR	Serial	CTR
Sort Blocks	70	72	90	92	94	94
Place Dual Shoes	0	0	26	20	38	38
Scan Object	0	2	4	10	62	70
Put Object Cabinet (sim)	2	4	8	24	36	66
Sort Letters	0	2	12	18	92	94
Organize Hardware	0	2	0	0	8	16
Deliver Water	20	26	14	34	78	92
Put Object Cabinet (real)	32	38	54	68	92	98

5.4 Generalization beyond training support

To distinguish generalization from coverage of the tested bimanual progress states, we test an execution-time intervention absent from training and then explicitly withhold a region of the bimanual progress-state space.

Unseen execution delays. On the Independent Sort Letters task, we externally pause only the opposite arm for $d \in \{1, 2, 3, 4\}$ seconds, matching the evaluated arm's local initial state between control and intervention branches. Since this pause does not change the evaluated arm's task requirements, selective dependence predicts little transfer to its completion time.

We measure $\Delta T_{\text{other}}(d) = T^{\text{pause}}_{\text{other}}(d) - T^{\text{control}}_{\text{other}}$ and fit its slope β_{delay} across pause durations; slopes near one and zero indicate near-complete transfer and approximate invariance, respectively. For $\pi_{0.5}$, a 4 s pause transfers 5.12 s under $L \to R$, 2.56 s under Concurrent, and 0.25 s under CTR, with fitted slopes of 1.232, 0.758, and -0.042 (Figure 5). CTR's weak response to irrelevant opposite-arm progress thus persists under an external execution delay.

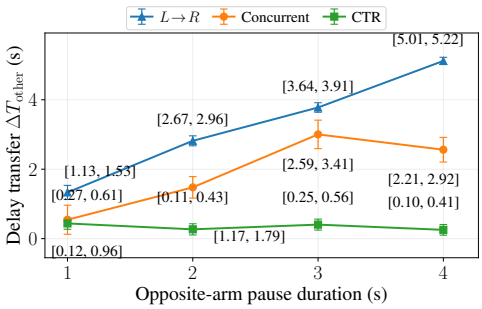


Figure 5: On the Independent Sort Letters task, we externally pause one arm and measure the unpaused arm's additional completion time relative to a matched no-pause control, testing whether an irrelevant execution delay propagates across arms. With $\pi_{0.5}$, CTR keeps delay transfer low, most clearly for pauses of 2 s or longer, whereas $L \to R$ and Concurrent transfer substantially more delay; error bars and brackets show 95% confidence intervals from 50 trials per condition and duration.

Held-out progress-state region. Because an unseen pause can still visit progress states covered by CTR, we also withhold progress states from training. On Sort Letters, we normalize each arm's progress within a reschedulable block to $[0, 1]$ and fix one held-out region, $H = [0.20, 0.30] \times [0.70, 0.80]$, leaving both arm-local progress values individually familiar. Mixed retains the three-schedule mixture, CTR-hole removes training samples whose bimanual progress states lie in H, and CTR-full retains full admissible support; training budgets are otherwise matched. We evaluate $\pi_{0.5}$ from both the standard initial state and states inside H. CTR-hole achieves 74% unseen-state SR versus 58% for Mixed and 76% for CTR-full; their standard initial-state SRs are 92%, 74%, and 94%, respectively (Table 5). The improvement over Mixed despite withholding H supports generalization to unobserved bimanual progress states.

6 Limitations

CTR relies on the task-specific processing pipelines; errors in interaction detection, segment boundaries, or task-relation logic can create invalid recombinations. Main-view composition only approximates a physically realized concurrent scene, so contact-sensitive and coupled segments remain protected rather than independently recombined. Finally, our interventions and diagnostics cover specific forms of cross-arm dependence rather than every possible bimanual interaction.

7 Conclusion

A bimanual policy should learn the task, not the demonstrator's schedule: it should remain invariant to the relative timing of independently schedulable arm-local subtasks while preserving dependence imposed by task-required cross-arm constraints. CTR promotes this selective cross-arm dependence by exposing the same arm-local sensorimotor content to diverse admissible schedules. Across our diagnostics and interventions, broader admissible schedule support suppresses spurious cross-arm coupling while retaining task-required coordination.

Reproducibility Statement

We submit an anonymized code package and the datasets generated with CTR as supplementary material with this paper. The code release provides the implementation needed to construct the CTR training data and reproduce the experimental workflow, while the accompanying generated datasets make the outputs of our data-construction procedure directly available for inspection and use without requiring reviewers to rerun the full generation pipeline. Appendix C reports the matched training and simulation-evaluation configurations, Appendix E defines the reported metrics and aggregation rules, and Appendix F specifies the CTR data-construction procedure, including segmentation and constraint extraction, admissible-schedule construction, temporal recombination, IdleMask supervision, qualification checks, and provenance/rejection handling. The supplementary code package is anonymized for double-blind review.

ETHICS STATEMENT

Temporal recombination is applied only within task-annotated admissible regions; prerequisites, sustained support, shared-object interactions, and workspace exclusions remain protected. Real-robot deployment uses the same independent safety procedures as the baseline policies. Any release of recorded camera streams should respect consent, privacy, and applicable data-use permissions.

AI USE STATEMENT

GPT-6 Astra was used at task setup time to generate task-specific video-processing pipelines. Those programs—not GPT-6 Astra itself—process the recorded demonstrations to detect task-relevant interactions, segment arm-local trajectories, produce the temporal dependency metadata that separates reschedulable from protected relations, and execute the task-specific recombination and validation logic. GPT-6 Astra is therefore not used as a per-video annotator and is not queried for episode-level labels. Generative AI was also used to discuss hypotheses and experimental design, assist literature discovery and comparison, formalize the method, analyze aggregate results, and draft and edit the manuscript and LaTeX. The robot experiments and reported empirical outcomes are produced by the experimental pipeline rather than by the writing model.

REFERENCES

Ezra Ameperosa, Jeremy A. Collins, Mrinal Jain, and Animesh Garg. RoCoDA: Counterfactual data augmentation for data-efficient robot learning from demonstrations. In *IEEE International Conference on Robotics and Automation*, 2025. URL https://arxiv.org/abs/2411.16959.

Núria Armengol Urpí, Marco Bagatella, Marin Vlastelica, and Georg Martius. Causal action influence aware counterfactual data augmentation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 1709–1729. PMLR, 2024. URL https://proceedings.mlr.press/v235/armengol-urpi-24a.html.

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

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025b. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=itonej9GIV.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10–11):1684–1704, 2025. doi: 10.1177/02783649241273668. URL https://doi.org/10.1177/02783649241273668.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Advances in Neural Information Processing Systems*, 2019. URL https://arxiv.org/abs/1905.11979.

Jennifer Grannen, Yilin Wu, Brandon Vu, and Dorsa Sadigh. Stabilize to act: Learning to coordinate for bimanual manipulation. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 563–576. PMLR, 2023. URL https://proceedings.mlr.press/v229/grannen23a.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Jian-Jian Jiang, Xiao-Ming Wu, Yi-Xiang He, Ling-An Zeng, Yi-Lin Wei, Dandan Zhang, and Wei-Shi Zheng. Rethinking bimanual robotic manipulation: Learning with decoupled interaction framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12427–12437, 2025a. URL https://openaccess.thecvf.com/content/ICCV2025/html/Jiang_Rethinking_Bimanual_Robotic_Manipulation_Learning_with_Decoupled_Interaction_Framework_ICCV_2025_paper.html.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. DexMimicGen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *IEEE International Conference on Robotics and Automation*, 2025b. URL https://arxiv.org/abs/2410.24185.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015–4026, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Kirillov_Segment_Anything_ICCV_2023_paper.html.

Andrew Choong-Won Lee, Ian Chuang, Ling-Yuan Chen, and Iman Soltani. InterACT: Interdependency aware action chunking with hierarchical attention transformers for bimanual manipulation. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 1730–1743. PMLR, 2025. URL https://proceedings.mlr.press/v270/lee25a.html.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *Computer Vision – ECCV 2024*, volume 15105 of *Lecture Notes in Computer Science*, pp. 38–55. Springer, 2024. doi: 10.1007/978-3-031-72970-6_3.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1820–1864. PMLR, 2023. URL https://proceedings.mlr.press/v229/mandlekar23a.html.

Haiyang Mei, Qiming Huang, Hai Ci, and Mike Zheng Shou. RobotSeg: A model and dataset for segmenting robots in image and video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14346–14356, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Mei_RobotSeg_A_Model_and_Dataset_for_Segmenting_Robots_in_Image_CVPR_2026_paper.html.

OpenAI. GPT-6 Astra: A new generation of intelligence, 2026. URL https://openai.com/index/gpt-6-astra/. OpenAI product announcement, accessed 2026-09-26.

Xingyu Peng, Chen Gao, Liankai Jin, Annan Li, and Si Liu. BiCoord: A bimanual manipulation benchmark towards long-horizon spatial-temporal coordination. *arXiv preprint arXiv:2604.05831*, 2026. URL https://arxiv.org/abs/2604.05831.

Physical Intelligence. openpi: Open-source models and packages for robotics, 2025. URL https://github.com/Physical-Intelligence/openpi. Software repository, accessed 2026-09-26.

Silviu Pitis, Elliot Creager, and Animesh Garg. Counterfactual data augmentation using locally factored dynamics. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2007.02863.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *International Conference on Learning Representations*, pp. 28085–28128, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/45c1f6a8cbf2da59ebf2c802b4f742cd-Abstract-Conference.html.

Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, Xinghang Li, et al. RoboCOIN: An open-sourced bimanual robotic data collection for integrated manipulation. *arXiv preprint arXiv:2511.17441*, 2025. URL https://arxiv.org/abs/2511.17441.

Xuanran Zhai, Zekai Huang, Longyan Wu, Qianyou Zhao, Qiaojun Yu, Jieji Ren, Ce Hao, and Harold Soh. SkillVLA: Tackling combinatorial diversity in dual-arm manipulation via skill reuse. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026. doi: 10.15607/RSS.2026.XXII.082. URL https://www.roboticsproceedings.org/rss22/p082.html.

James Zhao, Mingyuan Ba, and Weiming Zhi. Tri-manual visuomotor imitation learning of robot policies. arXiv:2607.25731, 2026. URL https://arxiv.org/abs/2607.25731.

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. URL https://arxiv.org/abs/2304.13705.

A SURVEY OF SCHEDULE COVERAGE IN EXISTING BIMANUAL DATASETS

We survey four publicly available bimanual manipulation datasets: RoboTwin 2.0 (Chen et al., 2026), RoboCOIN (Wu et al., 2025), DexMimicGen (Jiang et al., 2025b), and BiCoord (Peng et al., 2026) (Table 6). A task is *reschedulable* when the relative timing or ordering of its two arms can change without altering task semantics or violating a task-required dependency. We call a reschedulable task *diversity-covered* when its released demonstrations contain at least two materially different schedules; otherwise it is *locked*. Different schedules differ in subtask order or in a material inter-arm timing offset.

Table 6: Dataset size and schedule coverage for the surveyed bimanual datasets. Locked is the percentage of reschedulable tasks demonstrated under a single observed schedule.

Dataset	Domain	Released tasks	Reschedulable tasks	Diversity-covered tasks	Locked (%)
RoboTwin 2.0	Simulation	50	12	0	100.0
RoboCOIN	Real	639	231	0	100.0
DexMimicGen	Simulation	9	8	6	25.0
BiCoord	Simulation	18	15	3	80.0
Pooled		716	266	9	96.6

Across the four datasets, the survey covers 716 released tasks, of which 266 (37.2%) are reschedulable. Only 9 of those tasks (3.4%) are diversity-covered, leaving 96.6% locked to a single observed schedule. The pattern is present in every dataset: RoboTwin 2.0 and RoboCOIN are 100% locked, BiCoord is 80% locked, and DexMimicGen is 25% locked. The equal-weighted mean of the four per-dataset locked fractions is 76.3%, showing that the conclusion is not driven only by the larger RoboCOIN collection. Overall, observed schedule coverage is narrow across existing bimanual datasets.

B TASK AND EVALUATION INVENTORY

The evaluation covers four simulated and four real-robot tasks spanning Independent, Shared-Workspace Exclusion, Coordination, and Prerequisite settings. Independent relations permit either arm's local progress to vary without affecting the other; Shared-Workspace Exclusion requires only non-overlap in a conflict region; Coordination requires a coupled or synchronized bimanual phase; and Prerequisite requires one arm to establish a condition before the other may proceed.

Table 7: Task inventory used in the experiments.

ID	Domain	Task	Class
S1	Sim.	Sort Blocks	Independent
S2	Sim.	Place Dual Shoes	Shared-Workspace Exclusion
S3	Sim.	Scan Object	Coordination
S4	Sim.	Put Object Cabinet	Prerequisite
R1	Real	Sort Letters	Independent
R2	Real	Organize Hardware	Shared-Workspace Exclusion
R3	Real	Deliver Water	Coordination
R4	Real	Put Object Cabinet	Prerequisite

Why each task has this relation. **S1, Sort Blocks (Independent):** this task is adapted from the Blocks Ranking task in RoboTwin 2.0 (Chen et al., 2026). We modify the setup so that each side contains two blocks to be manipulated, giving the two arms the same number of arm-local subtasks. The arms operate on disjoint block sets, and either arm can start, progress, or finish its local sorting without changing the validity of the other arm's actions. **R1, Sort Letters (Independent):** the two arms sort distinct letters into arm-local destinations, so one arm's progress is neither a precondition nor a synchronization signal for the other.

S2, Place Dual Shoes (Shared-Workspace Exclusion): the two placement motions are locally executable by either arm, but they use an overlapping placement/workspace region; simultaneous occupancy is therefore disallowed even though the motions outside that conflict region can be rescheduled. **R2, Organize Hardware (Shared-Workspace Exclusion):** the arms manipulate separate hardware items but share a conflict region, so one arm must wait while the other occupies that region and may proceed once it is released.

S3, Scan Object (Coordination): successful scanning contains a coupled phase in which the two arms must jointly establish and maintain the required scan configuration, so that phase cannot be independently retimed across arms. **R3, Deliver Water (Coordination):** the *Serve Water* phase requires mutually timed bimanual actions, while the surrounding *Place Bottles* and *Return to Standby* phases remain independently schedulable.

S4, Put Object Cabinet (Prerequisite): one arm must establish cabinet access before the other arm can validly insert or place the object, imposing a required order rather than a mere scheduling preference. **R4, Put Object Cabinet (Prerequisite):** the dependent placement must likewise wait for the cabinet-ready condition; withholding that condition should block the dependent arm until the prerequisite is released.

Progress-state interventions. For the progress-state interventions, each arm's local progress is normalized to $[0, 1]$. On Sort Letters, the six evaluated states are C00 $(0, 0)$, Cmid $(0.5, 0.5)$, L-lead $(0.75, 0.25)$, R-lead $(0.25, 0.75)$, CL $(1, 0)$, and CR $(0, 1)$. On Put Object Cabinet, the matched suite is C00 $(0, 0)$, Cmid $(0.5, 0.5)$, L-lead $(0.75, 0.25)$, R-lead $(0.25, 0.5)$, CL $(1, 0)$, and CR $(0, 0.5)$. C00 is the nominal start, Cmid is matched progress, L-lead and R-lead are interior off-canonical progress states, and CL/CR are task-specific ready or completion-side states.

Figure 6 shows the frozen initial scenes used for these two six-state evaluations.

C Training and simulation-evaluation configurations

Demonstrations and frames. In the 50-demonstration setting, each task has 50 independent source episodes. Temporal recombination can produce multiple derived episodes from one source. Their durations and recorded frame counts vary with the task and schedule; training uses the valid, unpadded frames rather than forcing equal frame counts.

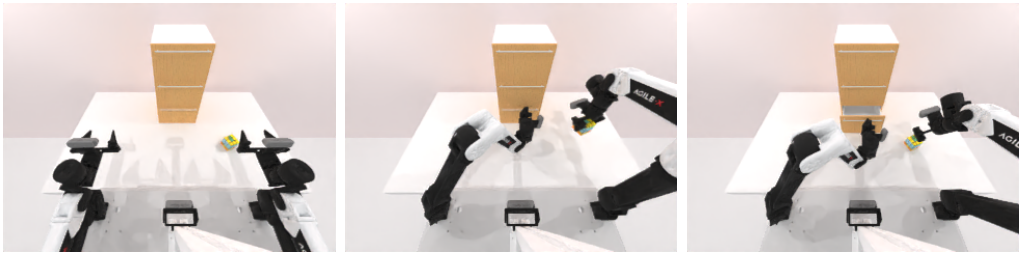
Unified training protocol. For the checkpoints reported in this paper, all $\pi_{0.5}$ models use the OpenPI implementation (Physical Intelligence, 2025) with LoRA (Hu et al., 2022): simulation models use 10,000 optimizer steps and real-robot models use 20,000 steps. All ACT models in both domains use batch size 8 and 100,000 optimizer steps. All DP models in both domains use 600 effective epochs, with each epoch defined over eligible training frame-window anchors.

Simulation evaluation. Within each task, all compared policies use the same frozen initial-scene suite, camera calibration, control clock, native action limit, success predicate, and rollout timeout. The common rollout timeout is used only to terminate unsuccessful evaluations; it is not the failure penalty used in T_{Pen}. Model-specific action queues and inference procedures are retained; parallel workers change only execution speed.

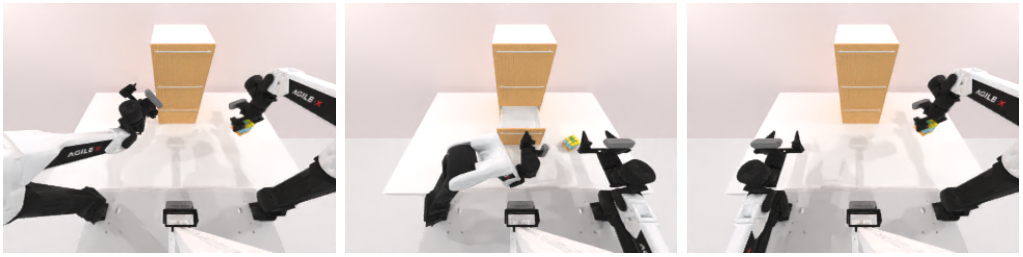
D Controlled schedule distributions: schematic illustrations

Figure 7 illustrates the geometry of the five controlled schedule distributions used in the main text. $L \to R$ contains only left-first serial execution; Serial combines $L \to R$ and $R \to L$; Concurrent starts both arms together; Mixed combines both serial orders and Concurrent; and CTR spans intermediate admissible schedules. For the real-robot tasks, the $L \to R$ distribution corresponds to the original, unretimed demonstrations; in simulation, we construct the same $L \to R$ schedule as the matched reference distribution. The curves are schematic, do not represent a particular physical task, and do not generate the empirical observations in Figure 3. Each schematic trajectory has independently normalized time; the number of displayed curves does not encode the distribution's mixture weights.

Put Object Cabinet (simulation)

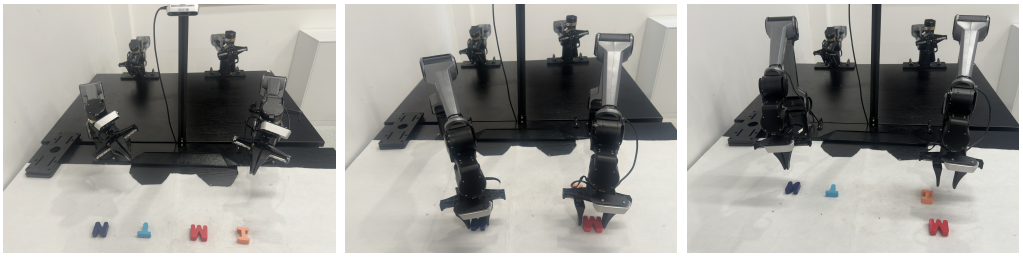


C00 (0, 0) | Cmid (0.5, 0.5) | L-lead (0.75, 0.25)

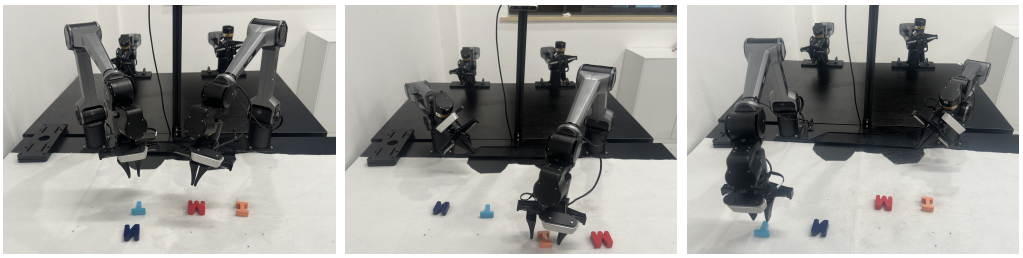


R-lead (0.25, 0.5) | CL (1, 0) | CR (0, 0.5)

Sort Letters (real robot)



C00 (0, 0) | Cmid (0.5, 0.5) | L-lead (0.75, 0.25)



R-lead (0.25, 0.75) | CL (1, 0) | CR (0, 1)

Figure 6: Initial-scene setups for the six progress-state evaluations. The first two rows show Put Object Cabinet in simulation; the last two rows show Sort Letters on the real robot. Each panel is labeled by its setup name and normalized bimanual progress state.

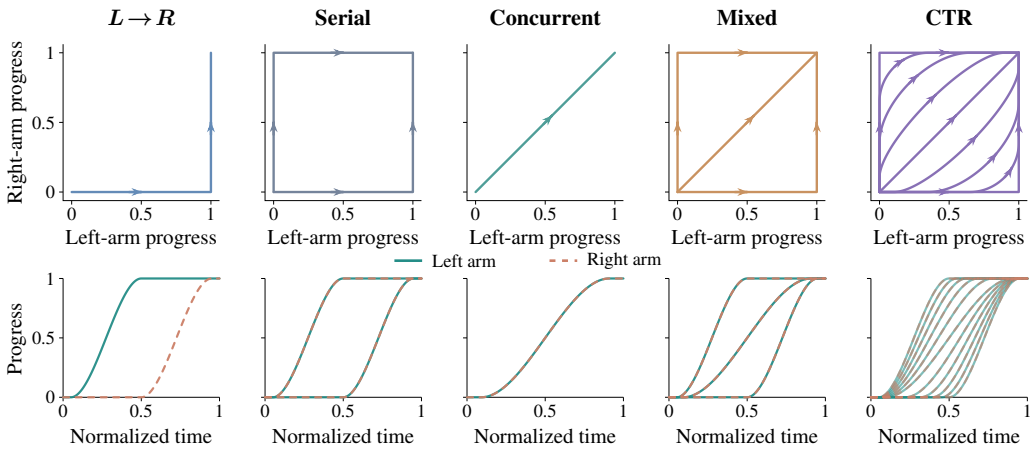


Figure 7: Idealized schedule geometry. Top: idealized bimanual progress-state support with left-arm progress on the horizontal axis and right-arm progress on the vertical axis. Arrows indicate increasing progress. Axis padding separates the trajectories from the axes without changing their shapes; in the top-row CTR panel, the two interior paths nearest the serial envelope are omitted for clarity. Bottom: the corresponding schedule families as arm progress over independently normalized time; solid teal and dashed orange denote the left and right arms, respectively. Both rows are schematic.

Empirical onset measurement. The densities in Figure 3 use Sort Blocks simulation trajectories. An arm's onset is the first of three consecutive recorded frames whose end-effector translation exceeds 2 mm from its initial position. Thus $\Delta_{\text{start}} = t_R^{\text{start}} - t_L^{\text{start}} > 0$ means that the left arm starts first; a negative value means that the right arm starts first. $L \to R$ contains 50 left-first episodes; each other group contains 100 episodes. Serial contains 50 episodes per order, and Mixed contains 34 left-first, 33 right-first, and 33 Concurrent episodes. Groups share source scenes and some episodes. Histograms use shared 1 s bins, with density equal to count divided by sample size and bin width, so each histogram has unit area. Gaussian kernel density estimates use a fixed 0.6 s bandwidth; their tails reflect smoothing rather than additional observed support. Concurrent remains unsmoothed because all observed offsets are zero at the approximately 0.04 s recording resolution. Vertical scales differ across panels.

E Metric definitions and aggregation

E.1 Success and failure-penalized completion time

For each policy–state cell c, let N_c be the number of evaluations, $S_{cj} \in \{0, 1\}$ indicate success, and t_{cj}^{goal} be the completion time for successful evaluation j. When $n_c^+ = \sum_j S_{cj} > 0$, define the cell's mean successful completion time and failure penalty as

$$\bar{t}_c^+ = \frac{1}{n_c^+} \sum_{j=1}^{N_c} S_{cj} t_{cj}^{\text{goal}}, \qquad P_c = 2\bar{t}_c^+. \tag{7}$$

We then report

$$\text{SR}_c = 100 \frac{n_c^+}{N_c}, \qquad T_{\text{Pen},c} = \frac{1}{N_c} \sum_{j=1}^{N_c} \left[S_{cj} t_{cj}^{\text{goal}} + (1 - S_{cj}) P_c \right]. \tag{8}$$

Thus each failed evaluation receives a penalty equal to twice the mean successful completion time of that same cell, rather than the common rollout timeout. Equivalently, with success proportion $s_c = n_c^+ / N_c$, $T_{\text{Pen},c} = (2 - s_c)\bar{t}_c^+$. If $n_c^+ = 0$, $\bar{t}_c^+$ and $T_{\text{Pen},c}$ are undefined and we report "–".

Undefined completion-time entries. For cells with SR=0, Table 1 reports "–" because the cell has no successful completion times from which to define $\bar{t}_c^+$ and hence $T_{\text{Pen},c}$.

Aggregation convention. Unless otherwise stated, summaries across tasks, model families, or conditions are equal-weighted macro-averages of the reported condition-level values, computed before display rounding. In particular, the default-condition "overall," simulation/real, and model-family summaries use equal-weighted means of task-level SRs rather than pooling episodes across heterogeneous tasks.

E.2 Relative start time

Schedule inheritance is measured by

$$\Delta_{\text{start}} = t_R^{\text{start}} - t_L^{\text{start}}. \tag{9}$$

Condition-level entries report the arithmetic mean of Δ_{start} over evaluations in that condition.

Table 8: Schedule inheritance on Sort Letters. Policy entries are mean $\Delta_{\text{start}} = t_R^{\text{start}} - t_L^{\text{start}}$ in seconds. The Demos average row reports dataset-average demonstration references: $+17.10$ s for $L \to R$ (mean left-arm task duration), 0.00 s for Concurrent, and -17.22 s for $R \to L$ (negative mean right-arm task duration).

Source	$L \to R$	Concurrent	$R \to L$
Demos average	17.10	0.00	-17.22
DP	14.74	0.05	-15.15
ACT	20.31	-0.04	-19.78
$\pi_{0.5}$	14.52	0.02	-14.43

E.3 Phase-conditioned cross-arm sensitivity

We use the 20-episode StagePair dataset for Deliver Water, selecting 600 matched groups: 300 with the left arm fixed and 300 with the right arm fixed. The groups are balanced across *Place Bottles*, *Serve Water*, and *Return to Standby*, with 100 groups per phase and intervention direction. Each group contributes two counterfactual points. In the $L \leftarrow R$ direction, the right-arm state is changed while the left-arm local state is preserved, and we measure the resulting change in the predicted left-arm Action. In the $R \leftarrow L$ direction, the left-arm state is changed while the right-arm local state is preserved, and we measure the resulting change in the predicted right-arm Action. Each phase therefore has 200 plotted points per direction. We compare matched CTR and Concurrent checkpoints for ACT, DP, and $\pi_{0.5}$; ACT is shown in the main text, and DP and $\pi_{0.5}$ are shown here. All composite scores use the shared gripper factor 0.7608885 degree-equivalent/mm described below.

For evaluated arm i and intervened opposite arm j, let $a_{k,h,d}^{i,\text{ref}}$ and $a_{k,h,d}^{i,\text{cf}(j)}$ be the reference and counterfactual predictions for matched group k, prediction step h, and joint Action dimension d. The six joint Action coordinates are measured in degrees and the gripper coordinate in millimeters. To avoid averaging these unlike units directly, we map gripper changes to a shared degree-equivalent scale using robust spans from the StagePair state distribution. Let $Q_{p,d}$ be percentile p of state coordinate d. Across the twelve joint coordinates in both arms, the median $Q_{90,d} - Q_{10,d}$ is $s_J = 42.9902°$; across the two gripper coordinates, the median span is $s_g = 56.5000$ mm. We use the same factor for every checkpoint and direction:

$$c_g = \frac{s_J}{s_g} = 0.7608885 \text{ degree-equivalent/mm}, \qquad |\Delta g|_{\text{eq}} = c_g |\Delta g_{\text{mm}}|. \tag{10}$$

This is a shared scale convention based on the input-state ranges, not a claim that gripper travel is physically an angle. The factor and source quantiles are recorded in the probe's normalization metadata.

Using six joint dimensions and one normalized gripper dimension, the point-level directional sensitivity is

$$d_{k,m}^{i \leftarrow j} = \frac{1}{7H_m} \sum_{h=1}^{H_m} \left[\sum_{d=1}^{6} \left| a_{k,h,d}^{i,\text{cf}(j)} - a_{k,h,d}^{i,\text{ref}} \right| + c_g \left| g_{k,h}^{i,\text{cf}(j)} - g_{k,h}^{i,\text{ref}} \right| \right], \qquad H_m \in \{8, 50\}. \tag{11}$$

Table 9: Normalized phase-selectivity gap G_{sel} in degree-equivalent units. Gripper differences are scaled by 0.7609 degree-equivalent/mm using median $q_{90} - q_{10}$ spans of $42.9902°$ for joints and 56.5000 mm for grippers. A positive value means sensitivity is higher during *Serve Water* than the mean of *Place Bottles* and *Return to Standby*. The phase-contained column excludes chunks that extend beyond the sampled phase boundary; the first-step and first-eight-step columns include all sampled groups.

Model	Training data	Full chunk	First step	First 8 steps	Phase-contained full chunk
ACT	CTR	$+2.26$	$+3.32$	$+3.16$	$+2.22$
ACT	Concurrent	-5.71	-5.12	-5.62	-6.35
DP	CTR	$+1.60$	$+1.14$	$+1.60$	$+1.60$
DP	Concurrent	-3.22	-3.06	-3.22	-3.22
$\pi_{0.5}$ 20k	CTR	$+1.11$	$+0.33$	$+0.61$	$+1.05$
$\pi_{0.5}$ 20k	Concurrent	-1.22	$+0.11$	0.00	-1.63

The resulting metric is in degree-equivalent units. ACT and $\pi_{0.5}$ use 50-step predicted chunks; DP uses the eight executed steps from its 16-step sampling horizon. Within each checkpoint pair, both panels use a shared y-axis. For visualization, source episodes are aligned within each phase by linearly scaling each phase duration to the mean duration of that phase; this changes only the horizontal coordinate of a point, not its sensitivity value. In the plots, *Place Bottles* and *Return to Standby* are labeled *Independent*; *Serve Water* is labeled *Synchronized*.

Let $\mathcal{K}_p^{L\leftarrow R}$ and $\mathcal{K}_p^{R\leftarrow L}$ denote the points in phase $p \in \{\text{place}, \text{serve}, \text{return}\}$ for the two intervention directions. We first average within each direction and then average the two directions:

$$S_p = \frac{1}{2}\left[\frac{1}{|\mathcal{K}_p^{L\leftarrow R}|}\sum_{k\in\mathcal{K}_p^{L\leftarrow R}} d_{k,m}^{L\leftarrow R} + \frac{1}{|\mathcal{K}_p^{R\leftarrow L}|}\sum_{k\in\mathcal{K}_p^{R\leftarrow L}} d_{k,m}^{R\leftarrow L}\right]. \tag{12}$$

Thus S_{place}, S_{serve}, and S_{return} are simply the phase-wise means of the two directional point clouds, with the two directions weighted equally even if their sample counts differ. The selectivity gap is

$$G_{\text{sel}} = S_{\text{serve}} - \frac{1}{2}(S_{\text{place}} + S_{\text{return}}). \tag{13}$$

When the two directions contain the same number of points in a phase, Equation 12 is equivalent to taking the mean y-value of all yellow and blue points in that phase. The probe is diagnostic rather than a task-success test.

Table 9 checks the selectivity gap after shortening the prediction horizon and after excluding chunks that reach beyond their annotated phase. The one-step and first-eight-step scores use the first 1 and 8 predicted Actions, respectively; the phase-contained score uses the full model chunk only when every predicted step remains within the sampled phase.

At 30 Hz, a 50-step ACT or $\pi_{0.5}$ chunk spans about 1.67 s. Of the 100 groups per direction and phase, 10 *Place Bottles* groups per direction reach into *Serve Water*, and 12 *Serve Water* groups per direction reach into *Return to Standby*. At the end of *Return to Standby*, 17 groups in $L \leftarrow R$ and 14 in $R \leftarrow L$ extend beyond the labeled segment end; the strict phase-contained column excludes these as well. DP's eight-step chunk spans 0.27 s, and none of its sampled chunks crosses a labeled segment boundary.

The ACT and DP gaps retain their signs at one step, over eight steps, and on phase-contained full chunks. For $\pi_{0.5}$, the CTR gap remains positive across these checks, while Concurrent is near zero at the first step and over eight steps but negative for the full phase-contained chunk. This suggests the $\pi_{0.5}$ difference develops later in the predicted horizon. Removing boundary-reaching chunks changes the full-chunk gaps only modestly, so phase crossing alone does not explain the full-horizon pattern.

E.4 Constraint violations and release latency

For Prerequisite tasks, a violation is an attempted dependent action before the required condition is available; for Shared-Workspace Exclusion tasks, it is forbidden concurrent occupancy of the

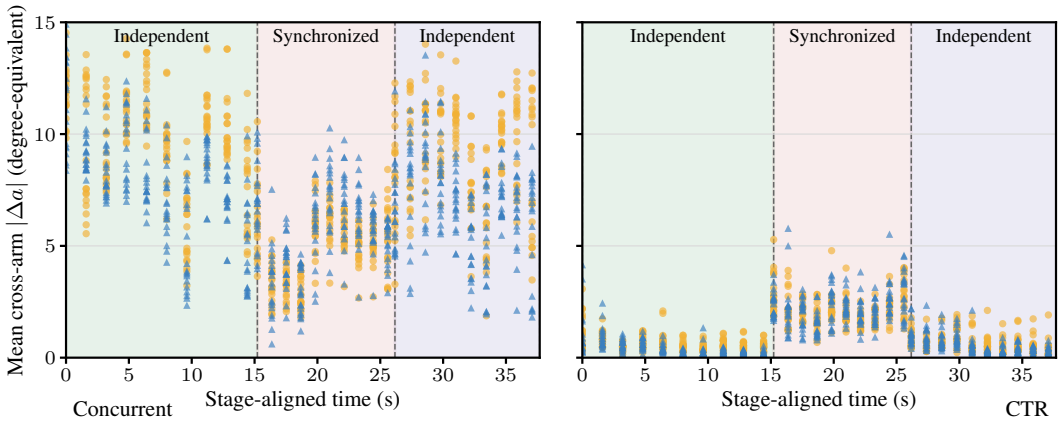


Figure 8: Phase-conditioned cross-arm sensitivity for DP checkpoints trained on Concurrent data (left) and CTR data (right). Yellow circles measure right-state-to-left-Action sensitivity; blue triangles measure left-state-to-right-Action sensitivity. The two outer regions are labeled Independent and the middle region is Synchronized.

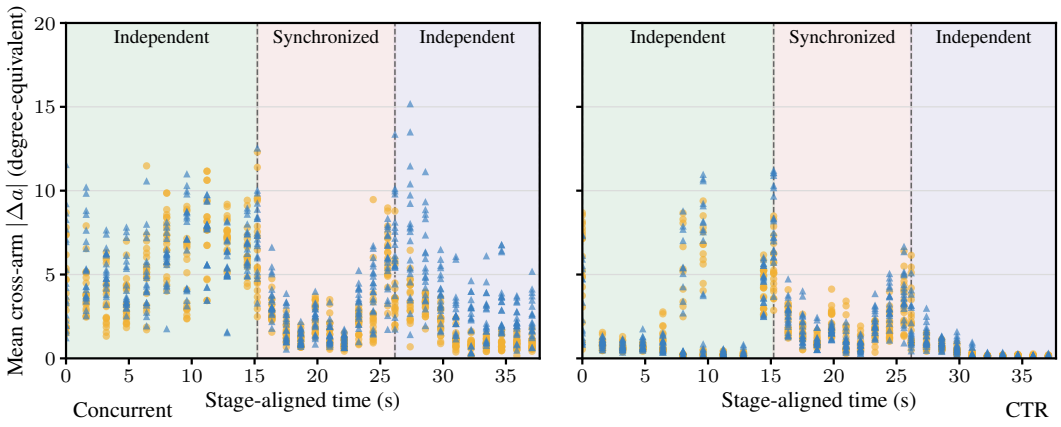


Figure 9: Phase-conditioned cross-arm sensitivity for $\pi_{0.5}$ 20k checkpoints trained on Concurrent data (left) and CTR data (right). Yellow circles measure right-state-to-left-Action sensitivity; blue triangles measure left-state-to-right-Action sensitivity. The two outer regions are labeled Independent and the middle region is Synchronized.

conflict region. Violation rate is the fraction of evaluations containing the task-specific violation. For a release event e, with t_e^{ready} the first time the blocked condition becomes available and t_e^{act} the first appropriate progress action, release latency is

$$L_e = t_e^{\text{act}} - t_e^{\text{ready}}. \tag{14}$$

Condition-level release latency is the arithmetic mean of L_e over the evaluated release events in that condition.

E.5 Paired pause effects

For the pause intervention, a control branch and a matched branch share the same local initial state and differ only in an externally imposed pause of the opposite arm for duration d. For the unfrozen arm,

$$\Delta T_{\text{other}}(d) = T_{\text{other}}^{\text{pause}}(d) - T_{\text{other}}^{\text{control}}. \tag{15}$$

At each duration $d \in \{1, 2, 3, 4\}$ seconds, the reported $\Delta T_{\text{other}}(d)$ is the arithmetic mean of the matched-pair differences, and $\text{SR}_{\text{other}}(d)$ is the unfrozen arm's local success rate at that duration.

We report β_{delay} as the fitted slope of $\Delta T_{\text{other}}(d)$ versus d over the four durations. A value near one indicates near-complete transfer of the imposed delay; a value near zero indicates invariance. For Figure 5, the confidence intervals are computed directly from the 50 measured delay values recorded for each condition and duration; these values are raw experimental trial records.

E.6 PROGRESS-STATE SUCCESS SUMMARIES

For the six-state progress-state robustness evaluation, Worst SR is the minimum of the six state-level success rates for that task and training distribution. For the held-out progress-state experiment on Sort Letters, the two arm-local progress coordinates are normalized to $[0, 1]$, and we use the fixed hyperparameter choice $H = [0.20, 0.30] \times [0.70, 0.80]$ for the held-out region. Unseen evaluations initialize inside H; initial-state evaluations start from the task's standard initial state. We report success rate only for $\pi_{0.5}$ under Mixed, CTR-hole, and CTR-full.

F CTR IMPLEMENTATION DETAILS

Algorithm 1: Constraint-preserving temporal recombination

Input: Demonstrations $\mathcal{D}$, task description ξ, schedule law $q(\cdot \mid \tau)$, draws $J \in \mathbb{N}$, learner m
Output: trained joint policy π_{θ_m}

$\widetilde{\mathcal{D}} \leftarrow \varnothing$
$\mathcal{P}_\xi \leftarrow \text{GeneratePipeline}_{\text{GPT6}}(\xi)$ ▷ *once per task*
for $\tau \in \mathcal{D}$ **do**
  $(\Sigma_\tau, \mathcal{R}_{\text{req}}(\tau)) \leftarrow \mathcal{P}_\xi(\tau)$ ▷ *automatic processing; no per-video GPT query*
  $\Omega(\tau) \leftarrow \text{Admissible}(\Sigma_\tau, \mathcal{R}_{\text{req}}(\tau))$
  for $j = 1, \ldots, J$ **do**
    $s_j \leftarrow \text{Sample}(q(\cdot \mid \tau))$
    $(\phi_L^{s_j}, \phi_R^{s_j}) \leftarrow \text{Retime}(\tau, s_j)$ ▷ *preserve local motion*
    $\widetilde{\tau}_j \leftarrow \mathcal{C}(\tau, \phi_L^{s_j}, \phi_R^{s_j})$ ▷ *compose real main view*
    $M_{s_j} \leftarrow \text{IdleMask}(s_j, \Sigma_\tau)$ ▷ *omit synthetic waits*
    $\widetilde{\mathcal{D}} \leftarrow \widetilde{\mathcal{D}} \cup \{(\widetilde{\tau}_j, M_{s_j})\}$
$\theta_m \leftarrow \text{Train}_m(\widetilde{\mathcal{D}}; \mathcal{L}_m)$

Operators and sampling. Here ξ is the task description, J is a finite number of schedule draws per demonstration, and $\Sigma_\tau = \{\sigma_{i,k}\}$ collects its ordered arm-local segments. GPT-6 Astra is invoked at task setup to generate the task-specific processing program $\mathcal{P}_\xi$; the resulting program, not GPT-6 Astra, processes each demonstration. $\mathcal{P}_\xi$ returns the detected segments and task-required relations $\mathcal{R}_{\text{req}}(\tau)$; Admissible builds the conservative schedule set $\Omega(\tau)$ described in Section 4.1. Each call to Sample draws one s_j from $q(\cdot \mid \tau)$, whose support lies in $\Omega(\tau)$; it does not enumerate a continuous schedule space. Retime constructs the source-time maps in Equation 16; $\mathcal{C}$ aligns the arm-local streams and composes the real-data main view; IdleMask constructs M_{s_j} using Equation 18; and Train_m optimizes the empirical version of the model-specific objective in Equation 6. In our experiments, $J \in \{1, 2\}$. All conditions are trained with the matched update budget in Appendix C; the deployed policy receives ordinary observations, and neither GPT-6 Astra nor the generated scheduler is queried online.

F.1 GPT-6 ASTRA AND THE GENERATED TASK PIPELINE

GPT-6 Astra is used at task setup time to generate the task-specific video-processing program $\mathcal{P}_\xi$; it is not the per-episode video annotator. The generated program encodes the task-specific logic for interaction detection, arm-local segmentation, protected-versus-reschedulable temporal relations, recombination and scheduling, and validation. It then runs automatically on the recorded videos or datasets to produce the segment and constraint metadata consumed by CTR. Task-required cross-arm constraints include ordering, synchronized execution, sustained support, and shared-region exclusion. The resulting pipeline outputs are used only during data construction.

F.2 ADMISSIBLE SCHEDULE CONSTRUCTION

A schedule is rejected when it violates an annotated task-required cross-arm constraint. Relative timing between segments governed by such a constraint is conservatively held fixed. Elsewhere, segment starts may be shifted so that independent work appears in serial, overlapping, and partially overlapping bimanual progress states.

F.3 SOURCE-TIME MAP AND RECOMBINATION

Given $s \in \Omega(\tau)$, arm i is active on $\mathcal{A}_i(s) = \bigcup_k \Sigma_{i,k}(s)$, and the source-time map $\phi_i^s : [0, \widetilde{T}] \to [0, T]$ reads

$$\phi_i^s(t) = \begin{cases} u_{i,1}, & 0 \le t < s_{i,1}; \\ u_{i,k} + (t - s_{i,k}), & t \in \Sigma_{i,k}(s) \text{ for some } k; \\ v_{i,k}, & t \in \big(s_{i,k} + \ell_{i,k},\ s_{i,k+1}\big),\ k < K_i; \\ v_{i,K_i}, & t > s_{i,K_i} + \ell_{i,K_i}. \end{cases} \tag{16}$$

On active segments, this map gives the content-preservation invariant in Equation 5 of the main text.

F.4 ROBOTSEG-BASED FOREGROUND EXTRACTION AND MAIN-VIEW COMPOSITION

For each real demonstration, arm-local wrist streams remain aligned to their own source times. General-purpose segmentation and open-vocabulary detection provide relevant foreground-extraction primitives (Kirillov et al., 2023; Ravi et al., 2025; Liu et al., 2024). RobotSeg-based arm segmentation (Mei et al., 2026) and the task-object annotation pipeline provide foregrounds and masks, which we select at $\phi_L^s(t)$ and $\phi_R^s(t)$ and composite over the scene background. Composition is limited to admissible schedules under the annotated task-required cross-arm constraints. Writing $F_i^s(t) = F_i(\phi_i^s(t))$ and $M_i^s(t) = M_i(\phi_i^s(t))$ for the foreground and mask of arm i and scene background B, the recombined main view is

$$\begin{aligned} \widetilde{I}_t^{\text{main}} &= \text{Compose}(B,\ M_L^s(t) \odot F_L^s(t),\ M_R^s(t) \odot F_R^s(t)) \\ &= \Big(1 - \textstyle\sum_i M_i^s(t)\Big) \odot B + \sum_i M_i^s(t) \odot F_i^s(t), \end{aligned} \tag{17}$$

the alpha-compositing form for non-overlapping masks, with occasional overlaps resolved by the annotated layering order.

F.5 IDLEMASK ROUTING

IdleMask is applied only to action coordinates and timesteps whose waiting was introduced by temporal recombination. The opposite arm's valid target at the same prediction step remains supervised. Required support and post-completion holding are not masked. Formally, output time t is an artificial wait for arm i on an inserted pre-segment or inter-segment gap, $t \in [0, s_{i,1}) \cup \bigcup_{k<K_i} \big(s_{i,k} + \ell_{i,k},\ s_{i,k+1}\big)$. For batch element b with sampled schedule s_b, prediction step k at output time t_{bk}, arm i, and action coordinate d,

$$m_{bkid} = \mathbb{I}[t_{bk} \in \mathcal{A}_i(s_b) \text{ or } t_{bk} > s_{i,K_i} + \ell_{i,K_i}]\,. \tag{18}$$

The masked action-supervision component is

$$\mathcal{L}_{\text{action}} = \frac{\sum_{b,k,i,d} m_{bkid}\, \ell_{bkid}}{\max(1, \sum_{b,k,i,d} m_{bkid})}, \tag{19}$$

Here ℓ is the base learner's coordinatewise action, denoising, or velocity-prediction loss; model-specific auxiliary terms in $\mathcal{L}_m$ remain outside this mask normalization.

F.6 HELD-OUT PROGRESS-STATE CONSTRUCTION

The held-out progress-state experiment on Sort Letters uses one reschedulable block with independently normalized left- and right-arm progress. We use the fixed hyperparameter choice

$H = [0.20, 0.30] \times [0.70, 0.80]$ for the single held-out region. No mirrored hole is used. Mixed is left unchanged and we verify that H is absent from its realized bimanual progress-state support. CTR-hole explicitly removes training samples in H, whereas CTR-full retains full admissible support. Initial-state evaluations start from the task's standard initial state, and unseen evaluations initialize inside H. All three conditions use the matched real-robot $\pi_{0.5}$ training configuration of 20,000 optimizer steps described in Appendix C; only the schedule-support construction and the stated CTR-hole filtering differ.

G PHYSICAL-VALIDITY AND OBSERVATION-CONSISTENCY AUDIT

For recorded real-robot demonstrations, preserving each arm's local trajectory does not imply that an arbitrary recombination of the two trajectories is physically realizable. Likewise, a recomposed main-camera image is not identical to a concurrently captured real observation. We therefore treat physical validity and observation consistency as *qualification problems*: CTR first proposes a schedule that satisfies the annotated task constraints, and the data pipeline then rejects candidates that fail geometric, contact, or observation-consistency checks. These checks make the operational schedule set deliberately conservative, but they are not a proof that every accepted trajectory would execute without deviation under closed-loop robot dynamics.

Conservative geometric qualification. For recorded real-robot data, we do not replay every counterfactual schedule on hardware during dataset construction. Instead, each arm advances monotonically along its recorded source clock; waits hold a recorded pose, and the scheduler does not invent arbitrary robot configurations. Newly paired left/right configurations and the motion between them are checked with the robot URDF, forward kinematics, and mesh-distance tests, together with task-specific prerequisite or shared-workspace gates. If the pipeline cannot certify the configured clearance for a newly created pairing, that schedule is rejected rather than accepted by default. These checks certify the declared geometric model and recorded-state interpolation, not unmodeled compliance, controller transients, calibration error, or contact dynamics.

Recorded contact phases. Some demonstrations contain intentional or unavoidable contact during a coupled manipulation. Requiring positive separation everywhere would therefore reject the original demonstration itself. We handle this case narrowly: a contact edge may be retained only when the two arm clocks reproduce the *same adjacent paired source frames* from the original demonstration. Such an edge is recorded explicitly in the contact ledger. It cannot be combined with a different opposite-arm source time, skipped into, or artificially prolonged. In contrast, every newly created cross-arm edge must pass the ordinary geometric qualification. This preserves observed coupled contact without treating it as permission to synthesize new contact configurations.

Observation composition and object state. Only the shared main view is spatially recomposed. The compositor maintains an explicit ownership model for task objects: an object is scene content before pickup, joins the corresponding arm foreground while it is carried, and returns to shared-scene ownership after release. The pipeline also reconstructs the object's source region after pickup and checks for duplicated objects at their original location. Where registered metric depth is available, overlapping arm foregrounds use depth for per-pixel occlusion ordering; otherwise the fallback ordering is recorded explicitly. Candidates can therefore be rejected even when timing is geometrically admissible—for example, when no safe schedule is certified or when the main-view composition audit indicates an implausible duplicate or otherwise unreliable composition.

The wrist cameras have a narrower guarantee. Each wrist stream follows its own arm's recorded source clock and is checked against the corresponding source video, but we currently do *not* recomposite the opposite arm when it happens to be visible in that wrist camera. Consequently, wrist-local content is source-consistent, while cross-arm appearance inside a wrist image can differ from the fully realized counterfactual scene. We treat this as a known approximation rather than as a physically exact observation model.