LIFT-WAM: FROM COUNTERFACTUAL ADAPTATION TO GROUNDED GOAL EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

In language-guided manipulation, the same scene can support multiple feasible goals, with the current instruction specifying the intended outcome. World-action models (WAMs) may continue executing scene-associated behaviors after an instruction changes, and correct target interaction can still fall short of establishing the requested final relation. We characterize this gap at three levels: alternative-goal responsiveness, goal grounding, and complete goal realization. Our counterfactual post-training framework, LIFT-WAM, learns alternative-goal behavior from native and counterfactual supervision. A Grounded Intent Extractor and an Intent-to-Action Adapter convert scene-grounded entity roles and target relations into explicit action conditions, while replay-verified corrective continuations supervise execution through goal completion. Under same-scene counterfactual evaluation with source initial states held fixed, LIFT-WAM achieves 54%/42% success on LIBERO-10/LIBERO-Object and 36%/28% on RoboTwin Clean/Random, compared with 44%/38% and 26%/18.5%, respectively, for counterfactual adaptation alone. Controlled Video Expert interventions further show that, after counterfactual adaptation, language-conditioned representations of current observations can redirect closed-loop control.

1 INTRODUCTION

Language-guided robotic manipulation requires realizing the goal specified by the *current instruction*. World–action models (WAMs) jointly model visual dynamics and robot actions and have shown strong manipulation capabilities (Ye et al., 2026; Yuan et al., 2026). Yet a familiar scene can support multiple goals involving different entity roles, spatial relations, or destinations. Success under familiar instructions therefore does not establish faithful instruction following: a policy may exploit scene–behavior associations without changing its goal when the instruction changes. For WAMs, this raises the question of how language shapes the representations used for control and redirects execution toward the requested goal.

Standard task success can obscure this distinction because visual generalization, manipulation difficulty, and goal selection are entangled when scenes and instructions vary together. Recent counterfactual evaluations begin to isolate language dependence through feasible alternative instructions (Fei et al., 2025; Fang et al., 2026; Zhao et al., 2026). We study WAMs through *same-scene counterfactual goal interventions*: the environment, objects, and initial robot state remain fixed while the instruction specifies a competing outcome, such as placing A to the right rather than the left of B, or exchanging their roles. The required behavioral change thus follows from the requested goal under fixed visual conditions.

Our analysis identifies three failure modes. **Behavioral persistence** occurs when the policy continues executing a scene associated goal despite a competing instruction. **Goal grounding error** occurs when it responds to the instruction but binds the goal to the wrong entity, role, or relation. **Goal realization failure** occurs when it interacts with an appropriate target but fails to establish the requested terminal relation. These failures define increasingly strong capabilities: responding to an alternative goal, grounding it correctly, and realizing it through complete execution. Faithful instruction following is therefore *hierarchical rather than binary*.

We present LIFT-WAM, a counterfactual post-training framework that connects language-specified intent to complete goal execution. Counterfactual adaptation provides the behavioral foundation by

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

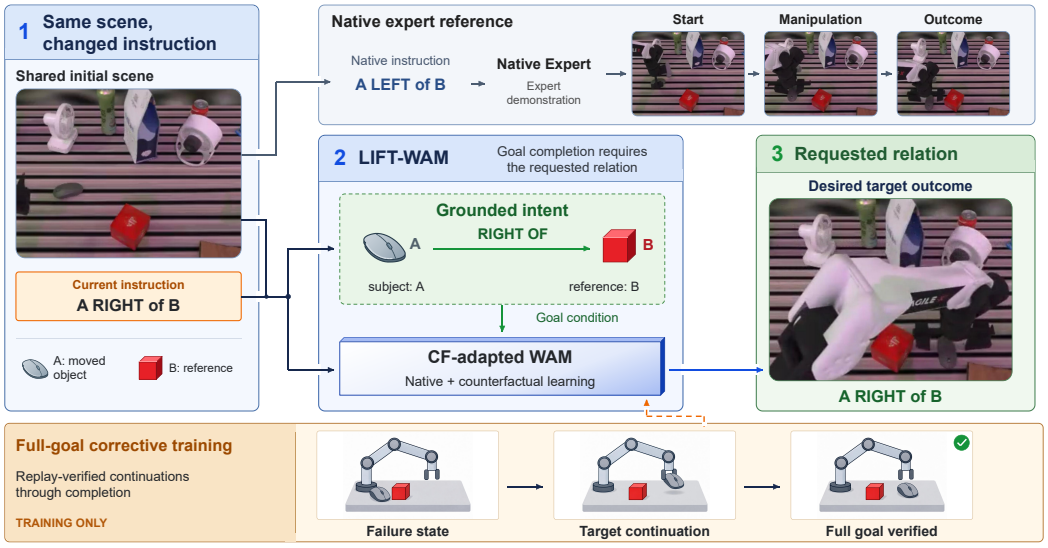

Figure 1: **LIFT-WAM: grounded intent for faithful goal execution.** Native and counterfactual expert demonstrations from the same RoboTwin initial scene illustrate how a changed instruction requires a different final relation between the manipulated object A and reference object B. LIFT-WAM builds on a counterfactually adapted WAM and exposes requested entity roles and relations as scene-grounded action conditions through GIE and the Intent-to-Action Adapter. Full-goal corrective training supervises execution through replay-verified goal completion. Video frames show expert references, not LIFT-WAM policy rollouts. The lower sequence schematically depicts corrective training, which operates only during training.

learning native and competing-goal trajectories in shared visual contexts. Building on this foundation, GIE and the Intent-to-Action Adapter expose the requested entity roles and relations as scene-grounded conditions for the action expert. Because competing goals can share grasping and lifting while requiring different continuations, full-goal corrective training supervises replay-verified continuations through achievement of the requested final relation.

Experiments on LIBERO and RoboTwin support this progression. Counterfactual adaptation establishes a strong foundation for alternative-goal execution, on which grounded intent and full-goal supervision further improve target-directed manipulation and goal completion. CF success rises from $18\%/26\%$ to $44\%/38\%$ on LIBERO-10/Object and from $3\%/0\%$ to $26\%/18.5\%$ on RoboTwin Clean/Random. Grounded goal conditioning further raises these rates to $46\%/40\%$ and $33.5\%/27.5\%$, respectively. Full-goal training particularly improves completion: with grounded intent on LIBERO-10, target lift rises from 62% to 64%, while complete CF success increases from 46% to 54%. Complete LIFT-WAM reaches $54\%/42\%$ on LIBERO-10/Object and $36\%/28\%$ on RoboTwin Clean/Random. Controlled Video Expert interventions further identify a language-conditioned representation pathway capable of redirecting closed-loop control after adaptation. Together, these results distinguish alternative-goal responsiveness, goal grounding, and goal realization as separate bottlenecks.

Our contributions are threefold:

- **A failure decomposition of WAM instruction following.** We characterize a hierarchy from alternative-goal responsiveness to grounded, complete execution by identifying behavioral persistence, goal grounding errors, and incomplete goal realization through controlled same-scene interventions.

- **A failure-targeted post-training framework.** We introduce LIFT-WAM, combining counterfactual adaptation, grounded intent-to-action conditioning, and replay-verified full-goal corrections to address these three failure modes.

- **Cross-benchmark behavioral and mechanistic evidence.** We demonstrate improvements in alternative-goal execution, target-directed manipulation, and goal completion across LIBERO and RoboTwin, and trace a language-to-control pathway through controlled Video Expert interventions.

2 Related Work

World-Action Models. DreamZero and LingBot-VA jointly model video and actions to connect visual dynamics with control (Ye et al., 2026; Li et al., 2026). Fast-WAM retains video co-training while removing future video synthesis at test time (Yuan et al., 2026). To support object-specific control, OA-WAM separates persistent object addresses from evolving content (Liu et al., 2026). SG-WAM predicts text-grounded and spatial-aware semantic foresight and injects it into the video expert to guide coupled video-action generation (He et al., 2026). Building on a counterfactually adapted Fast-WAM, LIFT-WAM exposes instruction-specified entity roles and target relations as scene-grounded conditions directly to the existing action expert.

Grounded Task Representations. Language-guided control uses spatial value maps (Huang et al., 2023), relational keypoint constraints (Huang et al., 2024), and affordance representations (Nasiriany et al., 2024; Li et al., 2024) to connect task semantics with execution. Learned action-conditioning interfaces include predicted latent visual futures in DIAL (Chen et al., 2026) and goal embeddings that align symbolic goals with scene entities in ProGAL-VLA (Darabi & Trivedi, 2026). Afford-VLA conditions actions on interaction masks learned jointly with action prediction (Wang et al., 2026), while HINT communicates tracked targets through visual highlighting or attention priors without adding trainable parameters to the action backbone (Mei et al., 2026). Within LIFT-WAM, GIE jointly represents instruction-specified subject and reference roles, target relations, and the current geometric and relational states of the grounded entities. The Intent-to-Action Adapter converts this structured representation into compact tokens that augment the original language context of the WAM action expert.

Instruction Following and Counterfactual Adaptation. LIBERO-Plus identifies weak language dependence in evaluated policies (Fei et al., 2025), while LIBERO-CF and InstructMove test instruction following among feasible alternative goals (Fang et al., 2026; Zhao et al., 2026). Inference-time methods use action guidance (Fang et al., 2026) or contrastive instruction-action verification (Kwok et al., 2026). Training-based approaches include Bayesian decomposition with a visual prior in LangForce (Lian et al., 2026), synthetic counterfactual instruction-action labels in CAST (Glossop et al., 2025), and learned alignment rewards for offline reinforcement learning in Counter-Align (Kondoh et al., 2026). CAST supervises alternative action chunks at recorded states without explicit supervision at subsequent counterfactual states. LIFT-WAM builds on counterfactual adaptation with grounded goal conditions and replay-verified corrective continuations through goal completion. Our evaluations distinguish adaptation gains from those of explicit grounding and full-goal supervision.

3 Method

At replanning step t, the policy receives multiview RGB observations o_t, proprioception q_t, and an instruction ℓ, and generates an action chunk $A_t = (a_t, \ldots, a_{t+H-1})$. LIFT-WAM builds on counterfactual policy adaptation, adds a grounded intent-to-action interface, and trains that interface and the experts through full-goal corrections (Figure 2). This organization describes the learning foundation and its extensions; the actual initialization and update scopes are specified below.

3.1 Counterfactual Policy Adaptation

Starting from released Fast-WAM, we train Video and Action Expert LoRA on a mixture of native and counterfactual trajectories, using the original language and proprioceptive context. Each paired example associates one observed state and demonstrated action chunk with a correct instruction ℓ^+ and a competing instruction ℓ^-. Native examples treat the source instruction as correct; counterfactual examples treat the target instruction as correct. This supervision trains execution of alternative goals while retaining native behavior.

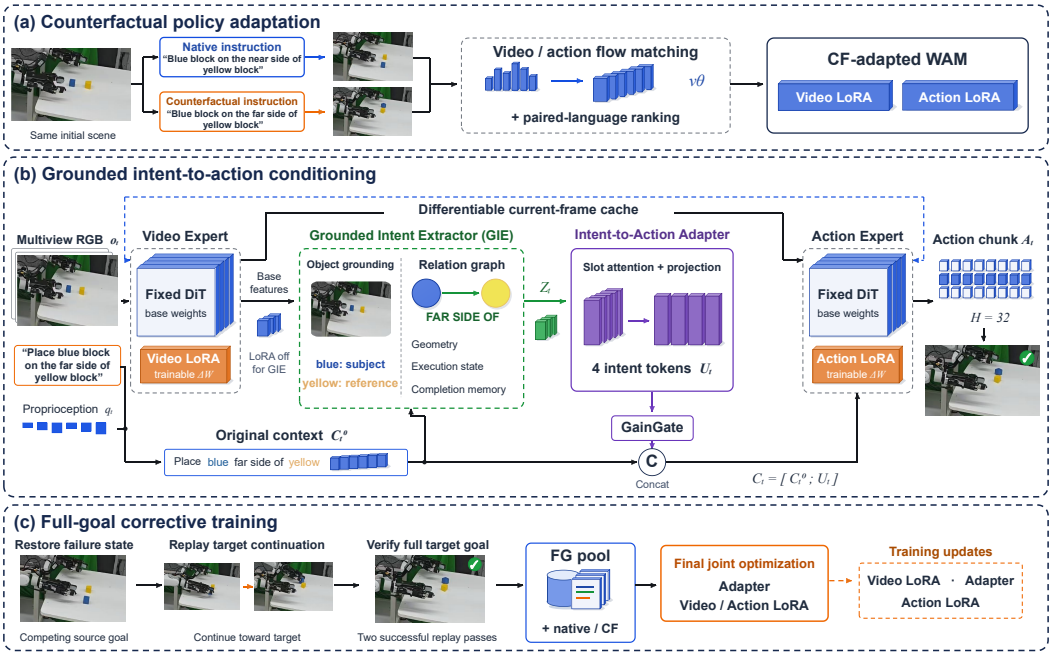

Figure 2: **From CF adaptation to grounded goal execution.** CF adaptation supplies the expert-policy foundation. GIE and the Adapter expose scene-grounded goals; verified full-goal continuations supervise final joint adaptation. Shading distinguishes fixed grounding modules from the updated Adapter and expert LoRA during that final stage. GIE is pretrained separately and the interface is warm-started before joint training. The frozen teacher fixes only the interface. Intent is admitted directly during training and by the frozen GainGate at deployment.

CF adaptation combines video and action flow matching, bidirectional paired-language ranking, and LoRA regularization. It updates only the experts' LoRA parameters, keeping the released backbone frozen. *CF-adapted WAM* uses this full recipe without structured intent tokens or the additional full-goal corrective pool. Appendix A.1 details the losses and benchmark-specific settings.

3.2 Grounded Intent-to-Action Conditioning

CF adaptation makes alternative-goal behavior learnable through language. The grounded interface further exposes what the requested goal requires in the current scene. GIE represents goal clauses as $c_i = (p_i, s_i, r_i)$: a predicate, subject role, and reference role. Separate role queries attend to multiview visual features to identify the corresponding entities and estimate their positions. A relation network combines these roles and relative geometry with predicate truth, execution phase, and interaction anchors. Goal routing produces the action-interface queries

$$I_t = G_\phi(o_t, q_t, \ell; m_t), \qquad Z_t = R_{\phi_R}(B_t, I_t), \tag{1}$$

where m_t retains completed-clause indicators and B_t contains queries derived from language, vision, and proprioception. Replay annotations train GIE and routing separately; deployment uses learned predictions. These modules read released-base video features with video LoRA temporarily disabled, preserving their initialization feature source.

The Adapter compresses Z_t into four tokens and maps them to the action expert's context dimension. For the augmented path,

$$\begin{aligned} T_t &= \mathrm{Comp}_\psi(Z_t), \quad & U_t &= P_\eta(T_t), \\ C_t &= [C_t^0; U_t], & A_t &\sim \pi_{\theta_0,\Delta\theta}(\cdot \mid o_t, q_t, C_t), \end{aligned} \tag{2}$$

where $C_t^0 = [H_\ell; H_{q_t}]$ is the original context, θ_0 denotes frozen backbone weights, and $\Delta\theta$ contains the two experts' LoRA parameters. The original context remains available throughout action denoising. Action-side visual conditioning uses a separate cache with video LoRA enabled.

With CF-adapted experts and pretrained grounding modules frozen, warm-start training updates the Adapter and GainGate. Augmentation is enabled during training; at deployment, the gate decides token insertion once per chunk before denoising. LoRA remains active regardless of gating. Appendices A.1 and A.2 detail initialization, attention, and gating.

3.3 Full-Goal Corrective Training

Correct target interaction does not ensure that the requested relation is established. Competing placement goals may share grasping and lifting but require different continuations. We restore captured states where execution followed a competing source goal, replay candidate expert continuations for the requested target, and retain those verified to satisfy the complete target-goal evaluator. The resulting full-goal (FG) corrective pool extends supervision through goal achievement and is mixed with native and counterfactual data. Appendix A.4 details construction and replay verification.

Final training uses the instruction pairing defined in Section 3.1. Correct and competing action predictions share noise, noise level, proprioception, and a differentiable action-side cache computed under ℓ^+; their language and intent conditions differ. Let e_i^+ and e_i^- denote their action-flow errors. We use positive-side ranking

$$r_i = [\delta_r + e_i^+ - \mathrm{sg}(e_i^-)]_+, \tag{3}$$

where sg stops gradients. The competing error provides a detached reference for correct-condition fitting. Semantic contrast also aligns injected intent tokens with the correct language representation.

The joint objective is

$$\min_{\psi,\eta,\Delta\theta} \quad \mathcal{L}_{\mathrm{action}} + \lambda_r \mathcal{L}_{\mathrm{rank}} + \lambda_s \mathcal{L}_{\mathrm{semantic}} + \mathcal{L}_{\mathrm{preserve}}. \tag{4}$$

The action term fits demonstrated continuations; preservation combines non-regression and selective distillation from a frozen copy of the warm-start Adapter evaluated through the current experts. Its teacher fixes the interface, not an entire action policy. Appendix A.3 specifies reductions, eligibility, and shared-prefix limitations.

This stage updates the Adapter and both experts' LoRA while freezing GIE, routing, GainGate, and the released backbone. Action losses reach video LoRA through the current-frame cache; unlike CF policy adaptation, this final joint stage adds no future-video flow loss. The optimization therefore learns how grounded conditions guide execution through verified goal completion.

4 Experiments

We evaluate the foundation provided by CF adaptation, trace language through the world–action interface, and examine how grounded intent and full-goal corrective training improve execution. CF target-goal success is the primary metric.

4.1 Experimental Setup

Instruction-following evaluation. We evaluate same-scene counterfactual instruction following on RoboTwin 2.0 (Chen et al., 2025) and LIBERO (Liu et al., 2023): the source environment and initial state are preserved, while the instruction specifies a competing goal. RoboTwin covers row-order reversal, stacking-role reversal, both spatial-placement directions, and tray-slot reassignment, with 40 episodes per setting under Clean and Random ($N = 200$ per environment and configuration). LIBERO uses compatible target-task predicates with strict-conflict filtering; LIBERO-10 and Object each use ten tasks and five trials per task ($N = 50$). Supplementary Goal results appear in the appendix. Our RoboTwin task definitions and evaluation procedure differ from RoboTwin-IF in Qwen-RobotManip (Qwen Team, 2026). Figure 6 and Appendices B.1–B.2 detail both protocols.

Implementation. CF adaptation updates Video/Action LoRA using the original context. Grounded intent adds four action-conditioning tokens, and FG supplies verified corrective continuations. LIBERO uses two camera views and seven-dimensional actions; RoboTwin uses three views and fourteen-dimensional actions. Benchmark-specific initialization, optimization, and training budgets are in Appendix B.3.

Table 1: Counterfactual goal success (%). LIBERO uses 50 episodes per suite and configuration; each RoboTwin column averages five equally weighted tasks, with 200 episodes in total. The three adapted configurations build on Fast-WAM. CF-adapted WAM uses neither grounded intent nor full-goal (FG) training; LIFT-WAM includes both.

	LIBERO		RoboTwin	
Configuration	10	Object	Clean	Random
LingBot-VA (Li et al., 2026)	20.0	10.0	6.5	3.5
Fast-WAM (base)	18.0	26.0	3.0	0.0
CF-adapted WAM	44.0	38.0	26.0	18.5
CF + grounded intent	46.0	40.0	33.5	27.5
LIFT-WAM (CF + intent + FG)	**54.0**	**42.0**	**36.0**	**28.0**

4.2 Counterfactual Adaptation Establishes a Strong Foundation

Adaptation improves execution of alternative instructions. Table 1 shows that CF adaptation raises Fast-WAM's LIBERO-10/Object success from 18.0%/26.0% to 44.0%/38.0%, and RoboTwin Clean/Random success from 3.0%/0.0% to 26.0%/18.5%. LingBot-VA also struggles with counterfactual goals, achieving 20.0% on LIBERO-10 and 6.5%/3.5% on RoboTwin Clean/Random, indicating that this difficulty extends beyond the Fast-WAM baseline. The gains from CF adaptation precede grounded intent and full-goal training, establishing adaptation as a strong foundation for alternative-goal execution. Detailed LingBot-VA evaluation settings and results, including the cross-suite LIBERO-Object reference, are provided in Appendix B.8.

Grounding and full-goal training build on this foundation. Grounded intent further raises LIBERO-10/Object success to 46.0%/40.0% and RoboTwin Clean/Random success to 33.5%/27.5%. Complete LIFT-WAM reaches 54.0%/42.0% and 36.0%/28.0%, respectively, achieving the highest success rates among the configurations in Table 1. Relative to CF adaptation alone, the complete framework adds 10.0/4.0 percentage points on LIBERO and 10.0/9.5 percentage points on RoboTwin. These improvements show that grounded intent and full-goal training strengthen the execution capability established by counterfactual adaptation.

4.3 Tracing Language through the World–Action Interface

We compare released Fast-WAM with a CF-adapted diagnostic checkpoint, both without structured intent, to examine the Video Expert's role in conditional prediction and control. Figure 3 summarizes fixed-input probes and closed-loop interventions, using protocols separate from the main benchmark.

Goal-related changes in world-model predictions. On ten shared states from five RoboTwin tasks, with three noise seeds per state, we fix observations, robot states, and noisy video inputs while changing language. At $\sigma = 1$, goal changes produce larger denoising-prediction RMS differences than either corresponding paraphrase in all five tasks for the adapted model. In stacking, the goal-change difference is 0.3422, versus 0.0370 and 0.0484 for source and CF paraphrases, indicating sensitivity beyond rewording.

We assess alignment with the requested future using the *CF-reference language-matching margin*

$$\Delta_{\mathrm{CF}} = \mathcal{E}_\Omega(\ell^S; V^{\mathrm{CF}}) - \mathcal{E}_\Omega(\ell^{\mathrm{CF}}; V^{\mathrm{CF}}), \tag{5}$$

where $\mathcal{E}_\Omega$ is velocity MSE against the same CF expert future over a fixed object region. Positive values favor CF language. The adapted model has positive margins for sorting, stacking, and both placement directions at $\sigma \in \{0.5, 1\}$; slot reassignment retains negative margins (Figure 3A). These margins characterize instruction matching against reference futures under the denoising objective. We next examine how language conditioning within the Video Expert affects action generation and closed-loop goal execution.

Cross-attention interventions steer WM predictions. We mask text conditioning or replace selected layers' cross-attention outputs with CF-conditioned donor residuals while retaining source

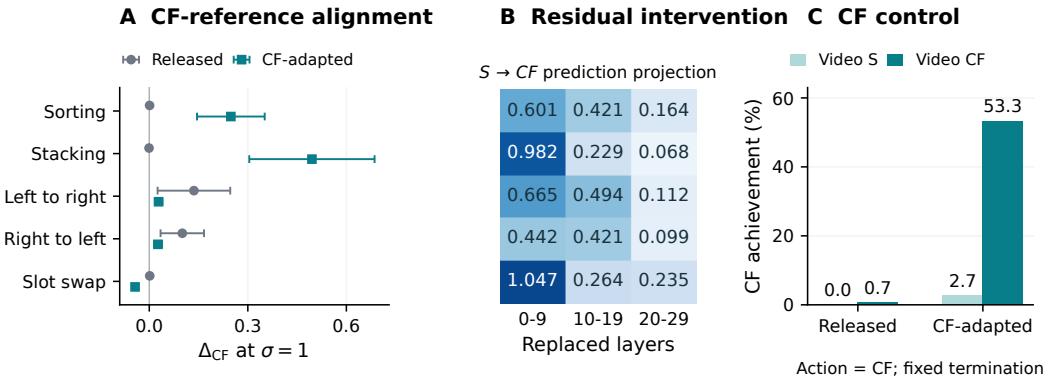

Figure 3: **Language effects on WM predictions and CF control.** A: CF-reference margins at $\sigma = 1$; positive values favor CF text. Segments span two scene means, not confidence intervals. B: Residual-replacement projections onto the model's S-to-CF prediction axis; task order follows A. C: CF success with action text and termination fixed to CF. A/B use ten shared states with three noise repetitions each; C uses 150 rollouts per bar. Generated futures are not policy inputs.

text. Donors are recomputed at the recipient's current noisy input. In stacking, masking layers 0–9 reduces the CF-reference margin from 0.49515 to 0.00502. Residual replacement in these layers yields a projection of 0.982 along the model's source-to-CF prediction direction, versus 0.068 for layers 20–29 (Figure 3B). Sorting and left-to-right placement also show stronger early-layer effects. Full controls are in Appendix B.9.

Language-conditioned video representations redirect control. With action-side CF text and termination fixed, switching video language from S to CF raises adapted-policy CF achievement from $4/150$ (2.7%) to $80/150$ (53.3%), versus $0/150$ to $1/150$ for released Fast-WAM (Figure 3C). Evaluation covers five tasks, ten matched scenes per task, and three noise seeds. Together with matched-state KV interventions (Appendix B.7), these results show that language-conditioned Video KV from current observations guides the Action Expert toward the requested goal.

4.4 Grounded Intent Improves Target-Directed Manipulation

Table 2 compares grounded intent and FG on top of CF adaptation, measuring both target interaction and complete task success. The CF + FG arm without intent completes the four configuration comparison, allowing us to examine each component's contribution individually and assess the additional gains from their combination.

Table 2: **Grounded intent and FG on top of CF adaptation (%).** Every row includes CF adaptation. Grasp and Lift are LIBERO-10 target-event rates; the success columns measure complete CF goals. All rates use 50 episodes. Bold marks column maxima, including ties.

Configuration	Intent	FG	Target Grasp	Target Lift	LIBERO-10 Success	Object Success
CF-adapted WAM	No	No	60.0	56.0	44.0	38.0
CF-adapted WAM + FG	No	Yes	62.0	56.0	50.0	**42.0**
CF + grounded intent	Yes	No	**72.0**	62.0	46.0	40.0
LIFT-WAM	Yes	Yes	**72.0**	**64.0**	**54.0**	**42.0**

Intent improves interaction with the requested target. On LIBERO-10, intent raises target grasp from 60% to 72% without FG and from 62% to 72% with FG. Target lift rises from 56% to 62% and 64%, respectively, while success improves by 2 and 4 pp. On Object, intent raises success from 38% to 40% without FG; both FG configurations reach 42%.

RoboTwin extends the evidence to relational manipulation. Without FG, intent raises Clean/Random success from 26%/18.5% to 33.5%/27.5%, gains of 7.5 and 9 pp. For left-to-right placement, success rises from 20%/12.5% to 42.5%/45%. Together with LIBERO's target-interaction measurements, these results support exposing requested roles and relations as scene-grounded action conditions.

4.5 Full-Goal Corrective Training Improves Goal Completion

Target interaction alone does not ensure that the robot establishes the relation specified by the instruction. Competing goals may share grasping and lifting motions but require different subsequent actions. We therefore examine whether FG improves complete task success beyond these shared interaction stages.

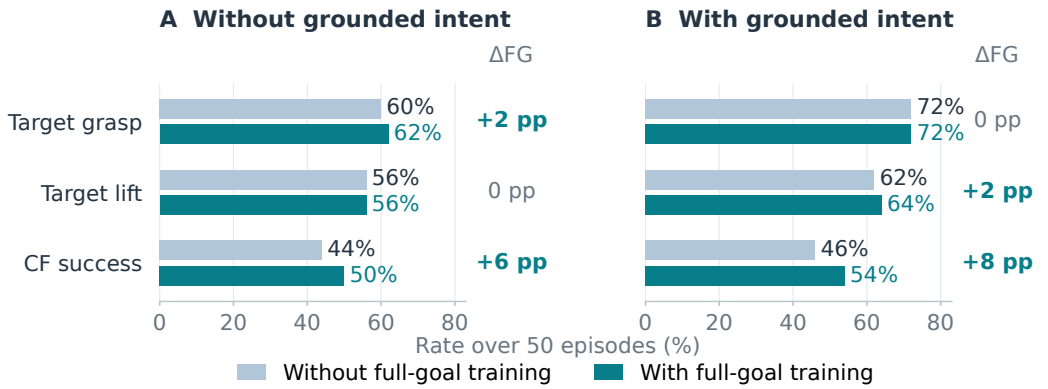

Figure 4: **Full-goal training improves completion beyond target interaction.** Bars compare FG within each intent configuration on LIBERO-10; ΔFG is the percentage-point change. All configurations use CF adaptation and 50 episodes. CF success records target-goal achievement during the rollout. The metrics are aggregate event rates, not stage-transition probabilities.

FG produces larger gains in complete success than in lifting. FG raises success from 46% to 54% with intent and from 44% to 50% without intent (Figure 4), gains of 8 and 6 pp. With intent, grasp remains 72% and lift rises from 62% to 64%; without intent, lift remains 56%. The corresponding lift gains are 2 and 0 pp. Both comparisons support corrective supervision through full goal completion, including for policies using the original conditioning path.

The complete framework improves on the adaptation foundation. FG raises intent-conditioned Object success from 40% to 42% and RoboTwin Clean/Random means from 33.5%/27.5% to 36%/28%. RoboTwin's smaller mean gains combine task-level improvements and regressions (Appendix B.5). LIFT-WAM leads on LIBERO-10 and RoboTwin, tying CF + FG on Object.

4.6 Behavioral Outcomes and Failure Analysis

Table 3 reports a separate 50-episode RoboTwin Random diagnostic, distinct from the main 200-episode benchmark. The base policy records a lift in every episode but achieves no CF goal, with 48 source-only outcomes. CF adaptation reduces source-only outcomes to 4 and raises CF successes to 10, demonstrating a substantial change in execution under alternative instructions. However, 36 episodes still satisfy neither goal, showing that reducing source-goal execution alone does not ensure completion of the requested target.

Table 3: **RoboTwin Random behavioral outcomes.** Counts out of 50 episodes per configuration. Lift events may overlap goal outcomes.

Configuration	CF	Source-only	Lift
Base policy	0	48	50
CF-adapted WAM	10	4	23
CF + intent	14	2	24
LIFT-WAM	15	2	24

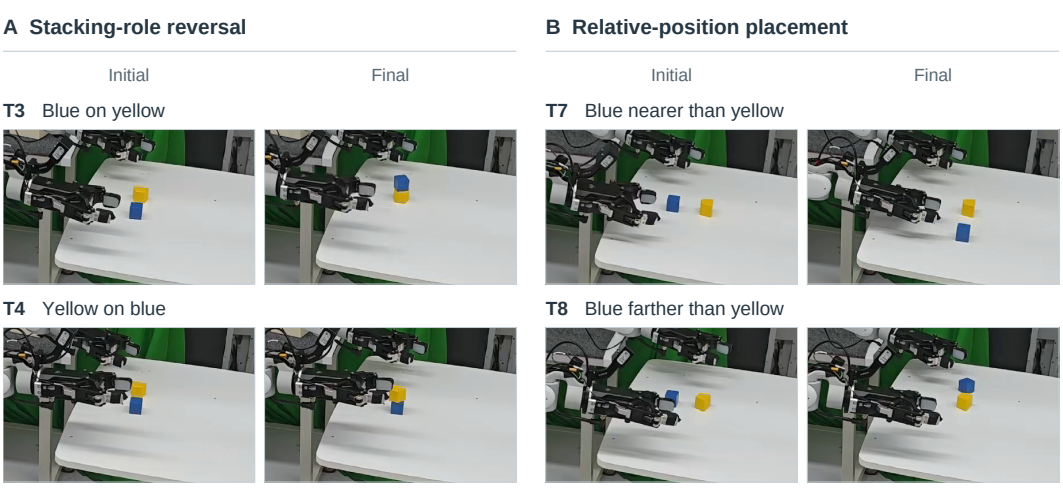

Figure 5: **Selected real-world executions on the UNT David A1C.** Initial and final frames show stacking-role reversal (T3–T4) and relative-position placement (T7–T8). Nearer/farther refer to the camera; labels summarize task goals.

Adding grounded intent increases CF successes from 10 to 14 while lift counts change from 23 to 24. FG adds one further success with the lift count unchanged, reducing lift-without-CF-success outcomes from 10 to 9. These descriptive comparisons complement the main benchmark by showing higher target-goal completion at similar frequencies of object interaction. Lift denotes lifting at least one tracked instruction-related object. Event definitions and full failure breakdowns appear in Appendix B.6; paired instruction-swap examples from matched initial states appear in Appendix B.11.

4.7 Real-World Evaluation

We evaluate LIFT-WAM on a UNT David A1C dual-arm robot following real-robot fine-tuning. The evaluation covers eight task conditions across four families: destination selection, stacking-role reversal, target-object selection, and relative-position placement. We evaluate each condition over 10 episodes, yielding 80 episodes in total.

LIFT-WAM achieves 28.75% success (23/80), with successful executions in all eight conditions. Figure 5 illustrates stacking-role reversal and relative-position placement through selected executions. These results provide preliminary evidence of goal-specific physical manipulation across object identity, destination, relational roles, and spatial arrangement. Training and deployment details, per-task results, and visualizations of all eight conditions appear in Appendix C.

5 Conclusion

Our study characterizes the progression from counterfactual adaptation to grounded goal execution in WAMs. LIFT-WAM builds on CF adaptation, which substantially improves execution of alternative instructions. Video-branch interventions show goal-related conditional predictions and a language pathway through current-observation representations that redirects CF control. On that foundation, GIE and the Intent-to-Action Adapter expose requested roles and relations to the action expert, while replay-verified full-goal corrections supervise their realization through complete execution. LIBERO and RoboTwin results distinguish the adaptation benefit from the further gains of these designs. LIBERO-10 diagnostics connect intent to improved target interaction and FG to larger gains in complete success than in lifting. Together, the findings support building faithful task execution through counterfactual learning, explicit grounded conditions, and supervision through goal completion.

AI USE STATEMENT

AI tools were used to assist with programming, improve the clarity and fluency of the manuscript, and support literature search and background research. They were not used to generate the core research ideas or to conduct experiments in a fully autonomous manner. The authors take full responsibility for the implementation, experimental results, references, and claims presented in this paper.

ETHICS STATEMENT

LIFT-WAM aims to improve the controllability of language-guided robots by reducing mismatches between instructed goals and executed behavior. Our evaluation covers simulated manipulation benchmarks and tabletop experiments on a physical robot following fine-tuning on teleoperated demonstrations. Improved instruction following does not ensure that an instruction is safe or appropriate, and errors in perception, grounding, or execution can still cause physical harm. The reported results therefore do not establish readiness for unsupervised deployment. Practical deployment should include human oversight, workspace and motion constraints, and independent emergency-stop mechanisms. Any reuse or release of pretrained models, benchmark assets, and robot recordings should respect applicable licenses and protect any identifying information incidentally captured during data collection.

REPRODUCIBILITY STATEMENT

Section 3 describes the framework and training objectives. Appendix A provides the model components, initialization procedures, parameter-update scopes, detailed losses, and full-goal data construction. Appendix B documents the benchmark-specific training configurations, evaluation protocols, success criteria, episode counts, and per-task results. Appendices B.7 and B.9 specify the closed-loop language interventions and fixed-input world-model diagnostics, including their sample definitions and experimental controls. Appendix C describes the real-robot demonstration dataset, fine-tuning configuration, deployment procedure, and task-level evaluation results.

REFERENCES

Tianxing Chen, Zanxin Chen, Baijun Chen, et al. RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088.

Yi Chen, Yuying Ge, Hui Zhou, Mingyu Ding, Yixiao Ge, and Xihui Liu. DIAL: Decoupling Intent and Action via Latent World Modeling for End-to-End VLA. *arXiv preprint arXiv:2603.29844*, 2026. URL https://arxiv.org/abs/2603.29844.

Nastaran Darabi and Amit Ranjan Trivedi. ProGAL-VLA: Grounded Alignment through Prospective Reasoning in Vision-Language-Action Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 9013–9022, 2026. URL https://openaccess.thecvf.com/content/CVPR2026F/html/Darabi_ProGAL-VLA_Grounded_Alignment_through_Prospective_Reasoning_in_Vision-Language-Action_Models_CVPRF_2026_paper.html.

Yu Fang, Yuchun Feng, Dong Jing, Jiaqi Liu, Yue Yang, Zhenyu Wei, Daniel Szafir, and Mingyu Ding. When Vision Overrides Language: Evaluating and Mitigating Counterfactual Failures in VLAs. *arXiv preprint arXiv:2602.17659*, 2026. URL https://arxiv.org/abs/2602.17659.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth Robustness Analysis of Vision-Language-Action Models. *arXiv preprint arXiv:2510.13626*, 2025. URL https://arxiv.org/abs/2510.13626.

Catherine Glossop, William Chen, Arjun Bhorkar, Dhruv Shah, and Sergey Levine. CAST: Counterfactual Labels Improve Instruction Following in Vision-Language-Action Models. *arXiv preprint arXiv:2508.13446*, 2025. URL https://arxiv.org/abs/2508.13446.

Junjie He, Junfeng Li, Zhide Zhong, Haodong Yan, Ruixin Li, Yangyang Zheng, Jiaguan Zhu, Tianran Zhang, Yuqiao Du, Wen Chen, Shunbo Zhou, and Haoang Li. SG-WAM: Text-Grounded and Spatial-aware Semantic Guidance for World-Action Models. *arXiv preprint arXiv:2608.08839*, 2026. URL https://arxiv.org/abs/2608.08839.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models. In *Proceedings of The 7th Conference on Robot Learning*, volume 229, pp. 540–562. PMLR, 2023. URL https://proceedings.mlr.press/v229/huang23b.html.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation. *arXiv preprint arXiv:2409.01652*, 2024. URL https://arxiv.org/abs/2409.01652.

Haru Kondoh, Kei Ota, Asako Kanezaki, and Yueh-Hua Wu. CounterAlign: Counterfactual Supervision for Vision-Language-Action Models. *arXiv preprint arXiv:2608.21740*, 2026. URL https://arxiv.org/abs/2608.21740.

Jacky Kwok, Xilun Zhang, Mengdi Xu, Yuejiang Liu, Azalia Mirhoseini, Chelsea Finn, and Marco Pavone. Scaling Verification Can Be More Effective than Scaling Policy Learning for Vision-Language-Action Alignment. *arXiv preprint arXiv:2602.12281*, 2026. URL https://arxiv.org/abs/2602.12281.

Jinming Li, Yichen Zhu, Zhibin Tang, Junjie Wen, Minjie Zhu, Xiaoyu Liu, Chengmeng Li, Ran Cheng, Yaxin Peng, Yan Peng, and Feifei Feng. CoA-VLA: Improving Vision-Language-Action Models via Visual-Textual Chain-of-Affordance. *arXiv preprint arXiv:2412.20451*, 2024. URL https://arxiv.org/abs/2412.20451.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal World Modeling for Robot Control. *arXiv preprint arXiv:2601.21998*, 2026. URL https://arxiv.org/abs/2601.21998.

Shijie Lian, Bin Yu, Xiaopeng Lin, Laurence T. Yang, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Cong Huang, and Kai Chen. LangForce: Bayesian Decomposition of Vision Language Action Models via Latent Action Queries. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2601.15197.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Yushan Liu, Peibo Sun, Shoujie Li, Yifan Xie, Lingfeng Zhang, Xintao Chao, Shiyuan Dong, Fang Chen, Xiao-Ping Zhang, and Wenbo Ding. OA-WAM: Object-Addressable World Action Model for Robust Robot Manipulation. *arXiv preprint arXiv:2605.06481*, 2026. URL https://arxiv.org/abs/2605.06481.

Mingyu Mei, Haojie Xu, Shihao Jin, Zibo Dai, Qihao Cheng, Zhengrui Lv, Hongjie Fang, Shirun Tang, Guang Chen, Xinyue Zhao, Huiliang Shen, and Zaixing He. HINT: Human-Intent Inception for Long-Horizon Robot Manipulation. *arXiv preprint arXiv:2609.02653*, 2026. URL https://arxiv.org/abs/2609.02653.

Soroush Nasiriany, Sean Kirmani, Tianli Ding, Laura Smith, Yuke Zhu, Danny Driess, Dorsa Sadigh, and Ted Xiao. RT-Affordance: Affordances are Versatile Intermediate Representations for Robot Manipulation. *arXiv preprint arXiv:2411.02704*, 2024. URL https://arxiv.org/abs/2411.02704.

Qwen Team. Qwen-RobotManip technical report: Alignment unlocks scale for robotic manipulation foundation models, 2026. URL https://arxiv.org/abs/2606.17846.

Runze Wang, Yuqian Fu, Yu Li, Tao Lin, Tianwen Qian, Mohamed Elhoseiny, Bo Zhao, Yanwei Fu, Yu-Gang Jiang, and Xiangyang Xue. Afford-VLA: Action-Aligned Visual Planning via Internalized Affordance. *arXiv preprint arXiv:2605.24203*, 2026. URL https://arxiv.org/abs/2605.24203.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Fan, and Joel Jang. World Action Models are Zero-shot Policies. *arXiv preprint arXiv:2602.15922*, 2026. URL https://arxiv.org/abs/2602.15922.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do World Action Models Need Test-time Future Imagination? *arXiv preprint arXiv:2603.16666*, 2026. URL https://arxiv.org/abs/2603.16666.

Mengao Zhao, Ziang Li, Chaodong Huang, Mengchen Ma, Haoyi Jiang, Yiwei Jin, Xinjie Wang, Yun Du, Xuewu Lin, Taojun Ding, Hongyu Xie, Jackson Jiang, Chunlei Yu, Kaihua Zhang, Lichao Huang, Liu Liu, Tianwei Lin, and Zhizhong Su. InstructMove: A Text-Indispensable Benchmark for Instruction-Following Manipulation. *arXiv preprint arXiv:2608.22990*, 2026. URL https://arxiv.org/abs/2608.22990.

A Additional Method Details

This appendix specifies the CF-adaptation foundation, grounded-representation learning, interface initialization, and final joint adaptation. The warm-start and final-stage objectives have different gradient routes and trainable parameter scopes.

A.1 CF Adaptation, Representation Learning, and Interface Initialization

Training stages and parameter scopes. The grounded representation and the language-conditioned expert policy provide two initialization components. Replay supervision trains GIE, GoalGraph, and their grounding and routing modules. The expert policy is the CF-adapted WAM described in Section 3.1, trained with native and counterfactual supervision without intent augmentation or the additional FG corrective pool. The warm-start stage combines these components and freezes them, training only the intent-token compressor, context projection, and GainGate. Final joint adaptation freezes the gate and grounded representation, and updates the compressor, context projection, and expert LoRA parameters. The interface teacher is copied from the warm-start compressor and context projection.

Counterfactual expert-policy adaptation. CF-adapted WAM retains the original language/proprioceptive conditioning path and trains only Video and Action Expert LoRA. Its learning objective includes correct-condition video flow matching, separately reduced native and counterfactual action flow matching, paired-language ranking in the video and action branches, and LoRA regularization:

$$\begin{aligned}\mathcal{L}_{\mathrm{CF}} = \lambda_v \mathcal{L}_{\mathrm{video}} + \lambda_a(\mathcal{L}^a_{\mathrm{native}} + \mathcal{L}^a_{\mathrm{CF}}) \\ + \beta_v \mathcal{L}^v_{\mathrm{pair}} + \beta_a \mathcal{L}^a_{\mathrm{pair}} + \beta_{\mathrm{reg}} \mathcal{R}_{\mathrm{LoRA}}.\end{aligned} \tag{6}$$

The backbone video/action loss weights are retained; the recipe sets $\beta_v = 0.10$, $\beta_a = 1$, and $\beta_{\mathrm{reg}} = 10^{-6}$. Both native and CF action weights are one. Correct/competing predictions share the demonstrated sample, sampled noise, and noise level. Native pairs supervise the source instruction and use the target as competing; CF pairs reverse these assignments, with validity masks selecting the eligible pairs.

For video-flow errors e_v^+, e_v^-, the per-pair hinge is $[\delta_v + \mathrm{sg}(e_v^+) - e_v^-]_+$ with $\delta_v = 0.01$. The LIBERO CF-adaptation recipe uses the analogous competing-side action hinge; the RoboTwin recipe enables correct-side action ranking and a language-matched action-side video cache. These recipe choices differ from the final LIFT-WAM objective in Section 3.3. LoRA regularization averages squared entries of the low-rank B factors. No GIE, intent-token compressor, context injector, or interface-preservation objective participates in CF adaptation.

The LIBERO recipe samples offline native, closed-loop native, historical CF, and strict CF pools with weights 1:1:1:1; it excludes the additional FG corrective pool. The corresponding RoboTwin recipe uses its benchmark-specific pools. Both configuration recipes specify rank-16 Video/Action LoRA, scale 16, dropout 0.05, and an AdamW learning rate of 5×10^{-6} with cosine decay for a 10,000-step training schedule. This schedule is distinct from interface warm-start and the final 25-step joint stage.

Role-specific visual grounding. The language decoder uses up to four clause slots, predicting clause activity and soft predicate distributions. Separate learned subject and reference embeddings produce role queries Q_i^k, $k \in \{s, r\}$. GIE reads current-frame Video Expert patch tokens before the transformer; their projections V_n include spatial positions and camera identities. The initial role attention and pooled features are

$$\alpha_{in}^{k,0} = \frac{\exp(\langle \widehat{Q}_i^k, \widehat{V}_n \rangle / \tau_v)}{\sum_m \exp(\langle \widehat{Q}_i^k, \widehat{V}_m \rangle / \tau_v)}, \qquad e_i^k = \sum_n \alpha_{in}^k V_n, \tag{7}$$

where hats denote normalized features and τ_v is the temperature. The final attention α_{in}^k applies the visibility-conditioned camera fusion described below. Entity positions are regressed from RGB features, predicted visibility, and attention-weighted image centers.

Visual features and camera fusion. The entity grounder reads current-frame pre-transformer video patch tokens, while GoalGraph reads final current-frame video hidden states. These grounding

and routing features are extracted with video LoRA temporarily disabled. A separate forward pass through the same Video Expert, with LoRA enabled, constructs the cache used for action prediction. During interface/gate warm-start, the expert parameters are frozen. During final joint adaptation, the student action cache is computed with autograd enabled, allowing gradients from action prediction to update video LoRA. The released Video Expert weights remain frozen throughout these two stages.

For the initial attention in Equation 7, suppress role superscripts and let

$$b_{iv} = \sum_{n \in v} \alpha^0_{in}, \qquad \widetilde{\alpha}_{ivn} = \frac{\alpha^0_{in} \mathbf{1}[n \in v]}{b_{iv}} \tag{8}$$

denote camera mass and within-camera attention. A bounded residual d_{iv} is predicted from the role query, within-camera pooled features, predicted visibility, and log camera mass. Camera fusion applies

$$\begin{aligned} \pi_{iv} &= \mathrm{softmax}_v(\log b_{iv} + d_{iv}), \qquad \pi^0_{iv} = \mathrm{softmax}_v(\log b_{iv}), \\ \alpha_{in} &= \alpha^0_{in} + \sum_v (\pi_{iv} - \pi^0_{iv}) \widetilde{\alpha}_{ivn}, \qquad e_i = \sum_n \alpha_{in} V_n. \end{aligned} \tag{9}$$

Camera masses are clamped before division and logarithms. A zero fusion residual produces zero attention correction. The grounder predicts overall and per-camera visibility and obtains image centers by attention-weighted pooling within each camera. Three-dimensional positions are bounded regression outputs.

Replay supervision. Trajectory replay supplies entity masks, entity positions, predicate truth, goal regions, and end-effector event tracks. Workspace coordinates are normalized componentwise as

$$\widetilde{x} = 2 \frac{x - x_{\min}}{x_{\max} - x_{\min}} - 1, \tag{10}$$

with numerical clipping. Goal anchors come from task goal regions or target geometry. Interaction anchors use end-effector positions at detected interaction events; for spatial manipulation, grasp anchors use the first detected object motion when available. These anchors provide supervision proxies for interaction locations.

The three phase labels denote pre-interaction, interaction/execution, and predicate completion. They are assigned from replay events such as object motion, gripper closure, and predicate truth. Grounding supervision includes predicate/activity and phase classification, patch-mask and attention supervision, visibility prediction, coordinate and anchor regression, and role-assignment constraints. Grounding refinements additionally train role binding, camera fusion, clause activation, and execution scheduling.

For counterfactual rollouts, execution-dependent source labels are masked when the behavior follows the target instruction. Available source-object geometry and predicate states can still be supervised, but the target execution is not treated as a source-goal grasp or interaction trajectory. These supervised losses train the initialization components and are inactive in Equation 4.

Routing and recurrent state. GoalGraph queries attend to language and current visual features. The relation bridge adds an attention residual over scheduled clause tokens. The grounding bridge constructs subject, reference, and relation tokens for each clause. Subject/reference tokens incorporate positions and grasp/goal anchors. Relation tokens incorporate the interaction anchor and goal-minus-grasp displacement. Coordinate features include the coordinates and sine/cosine terms at two frequencies. Role, clause, phase, and predicate-truth features complete the representation used by the bridge.

The execution-state head predicts pending, holding, retry, or completed. Across replans, only valid completed states are retained; holding and retry predictions serve as current-frame diagnostics. New completion combines predicted predicate truth with phase/state checks, and previously completed clauses remain marked completed. Offline paired training supplies the same replay-derived execution state to both instruction conditions. At deployment, recurrent state is updated from learned predictions; simulator labels and task identifiers are not policy inputs.

Warm-start action objective. Let $\mathcal{D}_{\mathrm{w}}$ contain warm-start rows with valid action supervision. Partition these rows into native and counterfactual sets $\mathcal{N}_{\mathrm{w}}$ and $\mathcal{F}_{\mathrm{w}}$. Let $\mathcal{S}_{\mathrm{w}}$ and $\mathcal{T}_{\mathrm{w}}$ denote eligible source-correct and target-correct pairs, using the validity rules in Appendix A.3. Warm-start draws

one noise realization per row. The correct, competing, and no-augmentation predictions share the demonstrated action, action noise, noise level, physical state, and correct-condition action-side video cache.

Using the errors from Equation 21, define

$$\begin{aligned} \mathcal{L}^{\mathrm{w}}_{\mathrm{act}} &= \langle \omega(\sigma) e^{+} \rangle_{\mathcal{N}_{\mathrm{w}}} + \langle \omega(\sigma) e^{+} \rangle_{\mathcal{F}_{\mathrm{w}}}, \\ r^{\mathrm{w}}_{i} &= [\delta_r + \mathrm{sg}(e^{+}_{i}) - e^{-}_{i}]_{+}, \\ \mathcal{L}^{\mathrm{w}}_{\mathrm{rank}} &= \langle r^{\mathrm{w}} \rangle_{\mathcal{S}_{\mathrm{w}}} + \langle r^{\mathrm{w}} \rangle_{\mathcal{T}_{\mathrm{w}}}, \\ \mathcal{L}^{\mathrm{w}}_{\mathrm{nr}} &= \left\langle [e^{+} - \mathrm{sg}(e^{0})]_{+} \right\rangle_{\mathcal{D}_{\mathrm{w}}}, \end{aligned} \tag{11}$$

where e^0 is the error of the same frozen experts without intent augmentation, and ω is the scheduler weight defined in Appendix A.3. The ranking margin is $\delta_r = 0.01$.

Warm-start ranking propagates gradients through the competing prediction, with the correct error detached. Because expert parameters are frozen, this gradient updates the Adapter through the competing context. Final joint adaptation instead uses the positive-side ranking in Equation 3, which detaches the competing error. The two stages therefore do not share the same ranking gradient.

GainGate supervision. Let g^{+}_{i} and g^{-}_{i} be gate probabilities under the correct and competing conditions. The correct-condition gate target is

$$y_i = \mathbf{1}\left[\mathrm{sg}(e^{+}_{i}) + \delta_g < \mathrm{sg}(e^{0}_{i})\right], \qquad \delta_g = 0.002. \tag{12}$$

This is an online-computed label from offline action-flow errors. It measures an improvement in predicting the demonstrated chunk; it is not a simulator-success label.

Set $\mathcal{P}_{\mathrm{w}} = \mathcal{S}_{\mathrm{w}} \cup \mathcal{T}_{\mathrm{w}}$ and $\mathcal{P}^{+}_{\mathrm{w}} = \{i \in \mathcal{P}_{\mathrm{w}} : y_i = 1\}$. Gate supervision is

$$\begin{aligned} \mathcal{L}_{\mathrm{gate}} = \left\langle \mathrm{BCE}(g^{+}, y) \right\rangle_{\mathcal{D}_{\mathrm{w}}} + \frac{1}{2} \left\langle \mathrm{BCE}(g^{-}, 0) \right\rangle_{\mathcal{P}_{\mathrm{w}}} \\ + \left\langle [m_g + \mathrm{logit}(g^{-}) - \mathrm{logit}(g^{+})]_{+} \right\rangle_{\mathcal{P}^{+}_{\mathrm{w}}}, \qquad m_g = 1. \end{aligned} \tag{13}$$

Probabilities are clamped to $[10^{-6}, 1 - 10^{-6}]$ for numerical stability. The competing condition supplies an explicit rejection target. The logit-ranking term applies only when correct-condition augmentation improves prediction according to Equation 12.

Writing gate parameters as ξ, the complete warm-start objective is

$$\min_{\psi,\eta,\xi} \quad \mathcal{L}^{\mathrm{w}}_{\mathrm{act}} + \mathcal{L}^{\mathrm{w}}_{\mathrm{rank}} + 2\mathcal{L}^{\mathrm{w}}_{\mathrm{nr}} + \mathcal{L}_{\mathrm{gate}}. \tag{14}$$

The Adapter and gate are optimized together. Gate targets and grounded inputs are detached, while compressed tokens remain differentiable into the gate. Consequently, gate losses also update the compressor. Action predictions use augmented context directly, so a hard gate decision does not block Adapter gradients during warm-start.

The LIBERO warm-start mixture includes native trajectories and historical and strictly verified counterfactual data, together with their replay annotations. It excludes the full-goal corrective pool introduced in final adaptation. The warm-start recipe uses 10,000 optimizer steps, learning rate 2×10^{-5} with cosine decay, weight decay 0.01, and gradient clipping at 1.0. It does not include paired semantic contrast or frozen-interface distillation. After warm-start, the compressor and context projection are copied to form the frozen interface teacher.

A.2 Adapter and Deployment Configuration

Adapter architecture. Four learned slots S read the routed queries through eight-head attention:

$$\begin{aligned} T_t &= \mathrm{LN}\big(S + \mathrm{MHA}\big(\mathrm{LN}(S), \mathrm{LN}(Z_t), \mathrm{LN}(Z_t)\big)\big), \\ U_t &= \mathrm{sigmoid}(\gamma)\ \tanh(\mathrm{MLP}_{\eta}(\mathrm{LN}(T_t))). \end{aligned} \tag{15}$$

The projection uses a two-layer MLP with hidden width 256 and GELU, mapping to the 4096-dimensional expert context. Its parameters η include the learned scale γ. Each output component is bounded in magnitude by $\mathrm{sigmoid}(\gamma)$.

Gate inputs. GainGate concatenates the mean of layer-normalized compressed tokens with eight layer-normalized diagnostics: subject/reference maximum patch-attention mass, averaged over clauses; subject/reference attention concentration, measured as one minus normalized entropy; mean clause activity; maximum clause execution probability; mean maximum execution-state probability; and mean execution-state validity. A two-layer MLP with hidden width 128 and a sigmoid produces g_t. The output layer is initialized to give admission probability 0.10. The configured deployment threshold is $\tau_g = 0.80$, with an optional runtime override. Freezing gate parameters during final adaptation does not fix its scores, because the trainable compressor can change its inputs.

Deployment context selection. Given the predicted admission score g_t and threshold τ_g,

$$C_t = \begin{cases} [C_t^0; U_t], & g_t \geq \tau_g, \\ C_t^0, & g_t < \tau_g. \end{cases} \tag{16}$$

The decision precedes action denoising and stays fixed throughout the chunk. Admitted tokens receive valid attention-mask entries at every denoising step; otherwise their positions are omitted. LoRA remains active in both cases. Warm-start and final joint adaptation enable augmentation directly, so the hard admission decision does not block Adapter gradients.

Expert adaptation. Both experts use LoRA with rank 16, scale 16, and dropout 0.05. Selected matrices include self-attention and cross-attention query/key/value/output projections, feed-forward projections, and text-embedding linear layers. For each selected matrix,

$$W = W_0 + \frac{\alpha_{\mathrm{L}}}{r_{\mathrm{L}}} BA, \qquad r_{\mathrm{L}} = \alpha_{\mathrm{L}} = 16. \tag{17}$$

The same expert parameters serve all instruction and augmentation conditions. Video and Action Experts have separate low-rank factors. Final adaptation adds no trainable action-output head or proprioceptive encoder.

Action generation. The action horizon is $H = 32$. At deployment, the gate selects the context once before denoising. The policy uses ten denoising steps with classifier-free guidance scale 1.0. Action inference uses current-frame visual conditioning and does not require generating a future video.

A.3 Final-Stage Losses and Gradient Routes

Validity masks and reductions. For an eligible set $\mathcal{E}$, define

$$\langle x \rangle_{\mathcal{E}} = \frac{\sum_i \mathbf{1}[i \in \mathcal{E}] x_i}{\max(1, \sum_i \mathbf{1}[i \in \mathcal{E}])}. \tag{18}$$

An empty eligible set contributes zero. Let $\mathcal{N}$, $\mathcal{F}$, and $\mathcal{K}$ contain rows with valid direct action supervision from native, noncorrective counterfactual, and corrective data, respectively. Native paired rows $\mathcal{S} \subseteq \mathcal{N}$ require bidirectional-language validity. Target-correct paired rows $\mathcal{T} \subseteq \mathcal{F} \cup \mathcal{K}$ require both paired-language and bidirectional-language validity. The same rules define the corresponding warm-start sets.

Padding is excluded within each per-row action error. Ranking and semantic multipliers modify numerators without renormalizing over only the nonzero-weight rows. In the full model, corrective action weights remain active, while $\kappa_i = 0$ suppresses ranking for lift-only corrective rows. Full-goal rows use $\kappa_i = 1$, subject to paired validity.

Ranking reduction. For eligible source-correct and target-correct sets $\mathcal{S}$ and $\mathcal{T}$, the penalties in Equation 3 are aggregated as

$$\mathcal{L}_{\mathrm{rank}} = \langle r \rangle_{\mathcal{S}} + \langle \kappa r \rangle_{\mathcal{T}}. \tag{19}$$

The multiplier κ_i is zero for corrective examples verified only by target lift and one otherwise. It changes the numerator without changing the masked mean's denominator. Full-goal corrections contribute when they satisfy paired-validity requirements. The competing error is detached in final adaptation; the distinct warm-start gradient rule is specified in Appendix A.1.

Paired semantic contrast. Let z_i be the normalized mean of injected intent tokens and h_i^+, h_i^- the normalized masked means of correct and competing language embeddings. We use

$$s_i = \left[\delta_s + \langle z_i, \mathrm{sg}(h_i^-)\rangle - \langle z_i, \mathrm{sg}(h_i^+)\rangle\right]_+,$$
$$w_i = \begin{cases} 1, & \kappa_i = 1 \text{ and } \mathrm{sg}(r_i) > 0, \\ \rho, & \text{otherwise}, \end{cases} \qquad \mathcal{L}_{\text{semantic}} = \langle w s \rangle_{\mathcal{S}\cup\mathcal{T}}. \tag{20}$$

We set $\rho = 0.5$. Language anchors and weighting decisions are detached. The loss updates the Adapter's intent context without training the text encoder or adding another action head.

Action-flow prediction errors. For a demonstrated action chunk A, noise $\epsilon \sim \mathcal{N}(0, I)$, and noise level σ, define

$$A_\sigma = (1-\sigma)A + \sigma\epsilon, \qquad v^* = \epsilon - A, \qquad e^{\pm} = \|v^{\pm} - v^*\|^2_{\text{valid}}. \tag{21}$$

The norm averages over action coordinates and unpadded timesteps. Correct and competing predictions share the demonstrated chunk, action noise, noise level, proprioception, and correct-condition action-side video cache. Their explicit language and intent conditions differ. The student cache remains differentiable during final joint adaptation.

Noise sampling and action loss. The scheduler samples $u \sim \mathcal{U}(0,1)$ and sets

$$\sigma = \frac{5u}{1+4u}. \tag{22}$$

Define $y(\sigma) = \exp[-2(\sigma - \frac{1}{2})^2]$. The action weight is

$$\omega(\sigma) = \frac{y(\sigma) - y_{\min}}{\overline{y} - y_{\min} + 10^{-10}}, \tag{23}$$

where $y_{\min}$ and $\overline{y}$ are the minimum and mean over the 1,000-point grid $\sigma_k = 5u_k/(1+4u_k)$, $u_k = 1 - k/1000$, $k = 0, \ldots, 999$.

Let $j = 1, \ldots, J$ index independent action-noise and noise-level draws. The final-stage action loss is

$$\widetilde{e}_i = \frac{1}{J}\sum_{j=1}^{J} \omega(\sigma_{ij}) e_{ij}^+,$$
$$\mathcal{L}_{\text{action}} = a_N \langle \widetilde{e} \rangle_{\mathcal{N}} + a_F \langle \widetilde{e} \rangle_{\mathcal{F}} + a_K \langle \widetilde{e} \rangle_{\mathcal{K}}, \tag{24}$$

with $J = 2$ and $(a_N, a_F, a_K) = (1, 1, 2)$. Ranking uses the first draw only, with $\delta_r = 0.01$. Semantic contrast is evaluated once per row from the injected context, with $\delta_s = 0.1$, $\rho = 0.5$, $\lambda_r = 1$, and $\lambda_s = 0.1$.

Shared-prefix limitation. Full-goal verification confirms the goal achieved by a corrective trajectory. Bidirectional pair validation checks the goal assignments and outcomes of the paired demonstrations. Neither check determines whether each local action chunk is instruction-discriminative. The implementation does not apply an additional action-chunk divergence filter.

For a shared chunk fitted perfectly under both conditions, $e_i^+ = e_i^- = 0$, Equation 3 still gives $r_i = \delta_r > 0$. At this exact fit, its gradient with respect to the correct flow prediction is nevertheless zero because $\nabla_{v_i^+} e_i^+ = 0$. Thus, the detached competing error prevents a direct gradient that increases competing-condition error in final adaptation, but does not remove the positive-margin incompatibility on shared prefixes.

Frozen-interface teacher. For draw j, let v_{ij}^0 be the current experts' prediction without intent augmentation and $\bar{v}_{ij}$ their prediction using the frozen warm-start interface. Both are evaluated without gradients using a detached correct-condition video cache. Define

$$e_{ij}^0 = \|v_{ij}^0 - v_{ij}^*\|^2_{\text{valid}}, \qquad e_{ij}^{\text{teacher}} = \|\bar{v}_{ij} - v_{ij}^*\|^2_{\text{valid}}. \tag{25}$$

Only the teacher's compressor and context projection are fixed. Its action predictions can change as the shared expert LoRA parameters change.

Let $\mathcal{K}_G \subseteq \mathcal{K} \cap \mathcal{T}$ contain full-goal-verified corrective rows. For each draw, preservation eligibility is

$$\begin{aligned} \mathcal{E}_{Nj} &= \left\{ i \in \mathcal{N} \cup \mathcal{F} : e_{ij}^{\text{teacher}} \le e_{ij}^{0} + \delta_N \right\}, \\ \mathcal{E}_{Gj} &= \left\{ i \in \mathcal{K}_G : e_{ij}^{\text{teacher}} \le e_{ij}^{0} + \delta_G \right\}, \end{aligned} \tag{26}$$

where $\delta_N = \delta_G = 0$. Rows without a valid action timestep are excluded. Eligibility is detached and measures an offline prediction-error comparison.

Define the action, token, and context discrepancies as

$$\begin{aligned} d_{ij}^{v} &= \|v_{ij}^{+} - \text{sg}(\bar{v}_{ij})\|_{\text{valid}}^{2}, \\ d_i^{T} &= \text{MSE}(T_i, \text{sg}(\bar{T}_i)), \\ d_i^{U} &= \text{MSE}(U_i, \text{sg}(\bar{U}_i)), \end{aligned} \tag{27}$$

where token/context MSE averages all token coordinates. The preservation term in Equation 4 groups non-regression and teacher penalties:

$$\begin{aligned} n_i &= \frac{1}{J} \sum_{j=1}^{J} [e_{ij}^{+} - \text{sg}(e_{ij}^{0})]_{+}, \\ \mathcal{L}_{\text{preserve}} &= b_N \langle n \rangle_{\mathcal{N} \cup \mathcal{F}} + b_K \langle n \rangle_{\mathcal{K}} \\ &\quad + \frac{1}{J} \sum_{j=1}^{J} \Big(c_N \langle d_j^{v} \rangle_{\mathcal{E}_{Nj}} + c_G \langle d_j^{v} \rangle_{\mathcal{E}_{Gj}} \\ &\qquad + c_T \langle d^{T} \rangle_{\mathcal{E}_{Gj}} + c_U \langle d^{U} \rangle_{\mathcal{E}_{Gj}} \Big), \end{aligned} \tag{28}$$

with $(b_N, b_K) = (2, 4)$ and $(c_N, c_G, c_T, c_U) = (1, 1, 0.1, 1)$. Token and context discrepancies are shared across noise draws, but their eligibility can vary with the draw. Lift-only corrective rows do not enter $\mathcal{E}_{Gj}$.

Final-stage gradients. The student action predictions use a differentiable current-frame video cache computed with video LoRA enabled. The action loss, positive-side ranking loss, and student-side action preservation and non-regression terms can therefore propagate gradients through the Action Expert and its visual conditioning into the Video Expert's LoRA parameters.

The no-augmentation reference and frozen-interface teacher are evaluated separately without gradients, using the current expert parameters. Their predictions provide detached comparison values and preservation targets; these reference evaluations do not detach the student action cache. Final ranking stops gradients through the competing error, while semantic contrast stops gradients through the language anchors and weighting decision.

Optimization updates the Intent-to-Action Adapter and the LoRA parameters of both experts. The released backbone, GIE, GoalGraph, grounding bridges, and GainGate remain frozen. No future-video flow loss is added; video LoRA learns through the action-prediction path.

A.4 Full-Goal Data and Optimization

Corrective trajectory construction. The data builder restores captured simulator states, checks restoration accuracy, searches expert-action suffixes for the target instruction, and evaluates the complete target goal during replay. A successful candidate is replayed again to record the corrective trajectory and confirm target-goal achievement. The recorded continuation reaches the verified success event and may include a short tail. Lift-only verification is stored separately and is not treated as full-goal success.

Full-goal admission criterion. Candidate target-action continuations are replayed from restored failure states. A continuation is admitted only if two replay passes both satisfy the complete target-goal evaluator:

$$\text{Success}(s_{T_{\text{goal}}}, \ell^{\text{tgt}}) = 1. \tag{29}$$

The recorded trajectory extends through verified goal achievement. This is a data-admission criterion; the continuation is learned through the action and paired objectives, without a separate full-goal loss.

Data mixture. The LIBERO final-stage corrective pool contains 35 episodes, covering seven designated source–target pairs with five episodes per pair. It is mixed with offline native, historical counterfactual, and strict counterfactual data using a $1{:}1{:}1{:}1$ pool-sampling ratio. This ratio specifies sampling weights, not equal dataset sizes.

Strict pairs have no identical goal clause and pass bidirectional replay checks: demonstrations achieve their own goal, fail the competing goal, and do not begin with the relevant goal already satisfied. Strict examples are balanced by conflict type and corrective examples by source–target pair. These checks establish trajectory-level goal assignments; they do not add a local action-divergence test. The 35 corrective episodes exclude all other pools and initialization data. RoboTwin uses its own benchmark-specific initialization and four data pools; the LIBERO episode count does not apply to RoboTwin.

Optimization budget. Training uses six NVIDIA L40S GPUs with bfloat16 mixed precision. Final joint adaptation uses AdamW with constant learning rate 2×10^{-6}, weight decay 0.01, gradient clipping at 1.0, and effective batch size 12. It runs for 25 optimizer steps, saving checkpoints every five steps. This budget counts only the final joint stage and excludes grounded-representation learning, counterfactual expert-policy adaptation, and the 10,000-step interface/gate warm-start.

B Evaluation Protocols and Additional Results

This appendix specifies the counterfactual interventions, benchmark-specific implementation settings, complete success counts, and statistical reporting. Configuration names follow Table 1.

Figure 6 provides a visual overview of the two instruction-following protocols. Task coverage is detailed in Appendices B.1 and B.2.

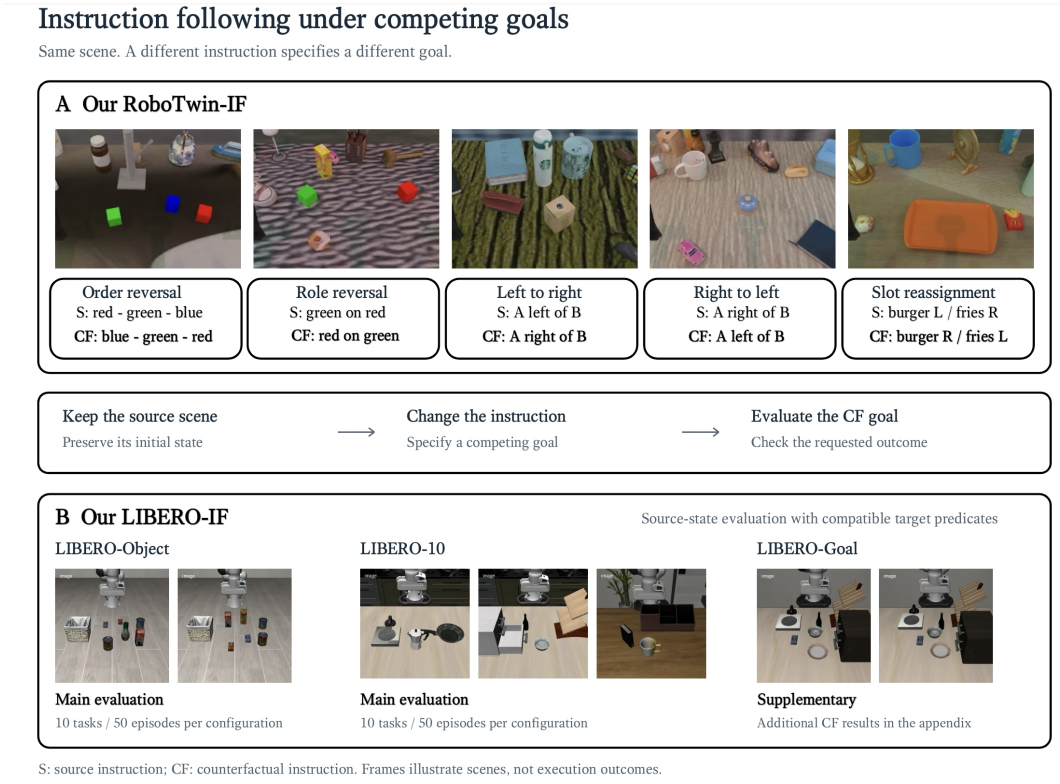

Figure 6: **Instruction-following evaluation under competing goals.** RoboTwin-IF covers five same-scene goal transformations under Clean and Random environments. The instruction summaries show source (S) and counterfactual (CF) goals; A is the manipulated object, B the reference object, and L/R the left/right tray slots. LIBERO-IF uses compatible target predicates in the source environment, with main comparisons on LIBERO-10 and LIBERO-Object and supplementary results on LIBERO-Goal. These representative scene frames come from recorded simulator videos; they are not evidence of policy success or tied to specific reported evaluation episodes. LIBERO panels show the external camera. The IF names are shorthand for the protocols used in this paper.

Metrics and comparisons. The primary metric is counterfactual target-goal success:

$$\text{CF success} = \frac{100}{N}\sum_{i=1}^{N}\mathbf{1}\left[\text{Success}(\tau_i, \ell_i^{\text{tgt}})\right], \qquad \tau_i \sim \pi(\cdot \mid \ell_i^{\text{tgt}}). \tag{30}$$

This corresponds to CIS in LIBERO and records target-goal achievement during the rollout. RoboTwin reports the macro-average over its five equally weighted settings, equivalent to pooled success over 200 episodes per environment and configuration. On LIBERO-10, we additionally measure Target Grasp and Target Lift, each over all 50 episodes.

The main table reports Base policy, CF-adapted WAM, CF + grounded intent, and complete LIFT-WAM. The LIBERO component comparison additionally includes CF-adapted WAM + FG, giving the four intent/FG combinations on top of CF adaptation. All configurations retain the original language context. Complete LIFT-WAM uses both grounded intent and FG.

The label CF-adapted WAM denotes the full adaptation recipe in Appendix A.1; it is not a data-only intervention. CF + grounded intent is the intent-conditioned configuration without FG. All RoboTwin CF-adapted WAM results use neither intent nor FG.

B.1 LIBERO Protocol and Execution-Stage Diagnostics

LIBERO-IF task coverage. LIBERO-IF evaluates whether a policy can redirect its behavior toward a language-specified alternative goal within the source environment. The evaluation covers three suites:

- **LIBERO-Object.** These tasks emphasize object identity: the policy must select the object specified by the current instruction and complete the corresponding manipulation. Other manipulable objects provide competing targets, testing whether the instructed object is correctly grounded in the scene.
- **LIBERO-10.** These tasks involve varied manipulation requirements and composite goals. Under an alternative instruction, the policy must adjust its execution and satisfy the corresponding target conditions, testing whether the requested goal guides behavior throughout task execution.
- **LIBERO-Goal.** This suite supplements the evaluation of behavior under different goal conditions in a shared environment. Its CF results are reported separately in Appendix B.4.

Our main evaluation uses ten tasks and five trials per task in each of LIBERO-10 and LIBERO-Object. The primary metric is CF target-goal success. On LIBERO-10, we additionally record Target Grasp and Target Lift to distinguish interaction with the instructed targets from achievement of the complete goal.

Instruction intervention. The evaluator preserves the source environment and imports the compatible target task's goal predicates. It checks instruction identity, environment compatibility, and the presence of the entities required by the target goal. Strict-conflict construction rejects identical or shared goal clauses. The associated bidirectional replay validation requires accepted demonstrations to achieve their own goal while failing the competing goal, with the alternative goal initially false.

Evaluation begins from benchmark initial states, with five trials for each of ten tasks. This differs from corrective-data construction, which restores intermediate states captured during unsuccessful closed-loop execution.

Rollout termination and success. The implementation provides suite-dependent policy-step limits of 700 for LIBERO-10 and 400 for LIBERO-Object and LIBERO-Goal. These limits, the initial settling period, and the replanning interval are configurable.

The diagnostic tracker records whether all target predicates hold simultaneously at any point during the rollout and records final-state truth separately. The reported Final Success metric uses target-goal achievement during the rollout. It does not add a separate requirement that the goal remain satisfied at the last recorded frame or indefinitely afterward.

Grasp and lift events. Target Grasp uses the simulator's grasp check. Target Lift measures vertical displacement from an object's initialized height, with a configured threshold of $0.04\,\mathrm{m}$ in the diagnostic implementation. Interaction with any subject appearing in the target goal can trigger a diagnostic event, including for goals with several subjects.

Each event rate uses all 50 rollouts as its denominator. The available summaries contain marginal counts of grasp, lift, and success.

B.2 RoboTwin Same-Scene Goal Interventions

Relation to prior instruction-following evaluation. Qwen-RobotManip (Qwen Team, 2026) evaluates language grounding through object selection, relative placement, and verb- or sequence-dependent manipulation, using instruction templates held out from training. Our evaluation focuses on changing the requested goal within the source scene through explicit counterfactual interventions. The five transformations below define our protocol.

Table 4: **RoboTwin counterfactual interventions.** Instruction descriptions summarize goal semantics rather than reproduce every sampled prompt verbatim. All distances are in meters. Each target condition is conjoined with both grippers being open.

Setting	Source → target	Target success condition
Row order	R–G–B → B–G–R, left to right	For adjacent blocks in B–G–R order, $0 < \Delta x < 0.13$ and $\lvert\Delta y\rvert < 0.03$.
Stack order	Green on red → red on green	Red relative to green: $\lvert\Delta x\rvert, \lvert\Delta y\rvert < 0.025$ and $\lvert\Delta z - 0.05\rvert < 0.012$.
Left to right	Object left of reference → right of reference	$\Delta x > 0$, $0.08 < \sqrt{\Delta x^2 + \Delta y^2} < 0.20$, and $\lvert\Delta y\rvert < 0.05$.
Right to left	Object right of reference → left of reference	$\Delta x < 0$, with the same planar-distance and orthogonal-distance bounds.
Slot reassignment	Burger left, fries right → burger right, fries left	Each object's designated functional point lies within planar distance 0.08 of its assigned tray slot.

RoboTwin-IF task coverage. RoboTwin-IF evaluates instruction following through five executable goal transformations in RoboTwin 2.0. These settings cover object ordering, entity roles, spatial relations, and assignments between multiple objects and destinations. The descriptions below summarize goal semantics rather than reproduce every sampled instruction verbatim.

- **Row-order reversal.** The scene contains red, green, and blue blocks. The requested left-to-right order changes from red–green–blue to blue–green–red. The policy must reorganize the relative positions of multiple objects according to the instruction, testing language-specified global ordering.
- **Stacking-role reversal.** The goal changes from placing the green block on the red block to placing the red block on the green block. The objects and stacking relation remain the same, while their manipulated and supporting roles are exchanged. This tests whether the policy distinguishes entity roles within a relation.
- **Left-to-right placement.** The goal changes from placing object A to the left of reference object B to placing it to the right. The manipulated and reference objects remain unchanged, while the required spatial relation changes. This tests whether language redirects transport and placement.
- **Right-to-left placement.** The reverse intervention changes the requested placement from the right of the reference object to its left. Evaluating both directions separately examines spatial redirection under opposite relational requirements.
- **Tray-slot reassignment.** The goal changes from placing the hamburger in the left tray slot and the fries in the right slot to the opposite assignment. The policy must update both object–destination correspondences and complete the combined instruction.

The two spatial-placement settings form a bidirectional native task pair; the remaining settings specify alternative goals within their source scenes. The evaluator checks that both source and target goals are initially false and selects the target-goal predicate for CF evaluation.

Each transformation is evaluated with 40 episodes per policy configuration under both Clean and Random settings, totaling 200 episodes per environment and configuration. Success requires the requested ordering, stacking, relative placement, or slot assignment to hold while both grippers are open. This evaluates whether the required relation has been established after release. The gripper requirement applies to these RoboTwin interventions and is not assumed for every LIBERO task. Table 4 specifies the corresponding geometric predicates and tolerances.

Scene reuse and outcome recording. The evaluation procedure supports recording accepted source scenes and source/target instructions, then reusing those records for the CF condition. The evaluator records selected-goal ever-success and final-state success separately. The CF results use target-goal achievement during the rollout.

Table 5: **Benchmark-specific final-stage recipes for LIFT-WAM.** Entries report initialization, trainable parameters, data mixtures, and optimization settings for each benchmark.

Setting	LIBERO	RoboTwin
Initialization	LIBERO warm interface checkpoint	RoboTwin intent-conditioned checkpoint after 200 optimization steps
Camera input	Two views; 224×448	Three views; 384×320
Action / proprioception dimension	7 / 8	14 / 14
Final trainable parameters	Adapter; Video LoRA; Action LoRA	Adapter; Video LoRA; Action LoRA
LoRA rank / scale / dropout	16 / 16 / 0.05	16 / 16 / 0.05
Optimizer and schedule	AdamW; constant learning rate	AdamW; constant learning rate
Learning rate	2×10^{-6}	3×10^{-6}
Effective batch size	12	12
Final optimizer steps	25	200 additional FG-stage optimizer steps
FG pool	35 trajectories; seven pairs; five per pair	96 training trajectories across two goal pairs, 48 per pair
Pool sampling	Native : historical CF : strict CF : FG = $1:1:1:1$	Benchmark-specific four-pool mixture, $1:1:1:1$
Checkpoint saving	Every five steps	Every 50 steps
Checkpoint selection	Final checkpoint at 25 steps	Final checkpoint at 200 steps

B.3 Benchmark-Specific Training Recipes

CF adaptation and the no-intent FG configuration. The CF-adapted WAM learning recipe is defined in Appendix A.1. It includes native/CF supervision and paired-language losses, with only Video/Action LoRA trainable. The LIBERO CF-adapted WAM + FG implementation starts from the CF-adapted expert policy and trains on a mixture including verified FG corrections while keeping the complete structured-intent path disabled. Its trainable parameters remain the experts' LoRA. This configuration tests full-goal supervision without the interface; it does not use the interface-specific semantic contrast or teacher-preservation terms.

Final joint adaptation. Both benchmark implementations train the Intent-to-Action Adapter and LoRA parameters of the Video and Action Experts. The student action branch uses a differentiable current-frame video cache, allowing action-prediction losses to update video LoRA. GIE and routing extract their features with video LoRA temporarily disabled. This branch-specific forward pass does not freeze video LoRA in the action-conditioning branch.

The released backbone, GIE, GoalGraph, grounding bridges, and GainGate remain frozen during final adaptation. The interface teacher fixes only the warm-start compressor and context projection; its predictions are evaluated through the current experts without gradients. No future-video flow loss is used in this final joint LIFT-WAM stage; the preceding CF-adaptation recipe retains video flow matching.

Training cost and data scope. The LIBERO final-stage recipe uses six NVIDIA L40S GPUs, BF16 mixed precision, gradient checkpointing, weight decay 0.01, and gradient clipping at 1.0. The 25 optimizer steps count only final joint adaptation. They exclude grounded-representation learning, counterfactual expert-policy adaptation, and interface/gate warm-start.

The 35 LIBERO FG trajectories form one pool rather than the complete training dataset. The four-pool ratio specifies sampling weights, not equal numbers of stored trajectories. RoboTwin uses its own initialization and data pools, so the LIBERO corrective-trajectory count does not specify RoboTwin's training-data size.

Table 6: **LIBERO configuration-level CF counts.** Each available cell uses 50 trials. Bold includes ties for the best LIBERO-10 and Object results.

Configuration	Intent	FG	LIBERO-10	Object	Goal
Released Fast-WAM	—	—	9/50 (18%)	13/50 (26%)	49/50 (98%)
CF-adapted WAM	No	No	22/50 (44%)	19/50 (38%)	49/50 (98%)
CF-adapted WAM + FG	No	Yes	25/50 (50%)	**21/50 (42%)**	49/50 (98%)
CF + grounded intent	Yes	No	23/50 (46%)	20/50 (40%)	**50/50 (100%)**
LIFT-WAM	Yes	Yes	**27/50 (54%)**	**21/50 (42%)**	**50/50 (100%)**

Table 7: **Approximate 95% Wilson intervals for LIBERO CF success.** Entries show success rate and interval endpoints in percent, using $n = 50$.

Configuration	LIBERO-10	Object
Released Fast-WAM	18.0 [9.8, 30.8]	26.0 [15.9, 39.6]
CF-adapted WAM	44.0 [31.2, 57.7]	38.0 [25.9, 51.8]
CF-adapted WAM + FG	50.0 [36.6, 63.4]	42.0 [29.4, 55.8]
CF + grounded intent	46.0 [33.0, 59.6]	40.0 [27.6, 53.8]
LIFT-WAM	54.0 [40.4, 67.0]	42.0 [29.4, 55.8]

B.4 LIBERO Counts and Statistical Reporting

Configuration-level CF counts. Table 6 provides the integer counts underlying the LIBERO results. All four adapted configurations are reported on LIBERO-10 and Object.

Approximate uncertainty. For k successes in n rollouts, we compute a 95% Wilson interval (Wilson, 1927), with $\hat{p} = k/n$ and $z = 1.959964$:

$$\frac{\hat{p} + z^2/(2n) \;\pm\; z\sqrt{\hat{p}(1-\hat{p})/n + z^2/(4n^2)}}{1 + z^2/n}. \tag{31}$$

These intervals use an aggregate binomial approximation. They do not account for dependence among trials of the same task, variation across training seeds, or checkpoint-selection effects. They are not confidence intervals over independently trained models. We do not derive paired significance tests or task-clustered uncertainty from the aggregate counts.

Execution-stage counts. Table 8 reports the marginal event counts used in Figure 4. The final column retains the aggregate success-to-lift count ratio as a descriptive statistic.

The reported ratio is $100\, n_{\text{success}}/n_{\text{lift}}$. Estimating success conditional on a lift event requires the joint count of episodes that both lift and succeed. Interpreting completion after lifting additionally requires event ordering. Neither is available in the marginal summaries. The ratios therefore do not establish that the same episodes became more likely to complete after lifting.

Contrasts supported by the stage counts. With FG absent, intent increases the lift count from 28 to 31 out of 50; with FG present, it increases the count from 28 to 32. These are gains of 6 and 8 pp. With intent present, FG changes grasp/lift/success counts from $36/31/23$ to $36/32/27$; without intent, lift remains 28 while success rises from 22 to 25.

B.5 RoboTwin Full Results and Task-Wise FG Differences

Table 9 reports complete task-level counts for the base policy, CF-adapted WAM, CF + grounded intent, and complete LIFT-WAM. Each configuration has 40 episodes per task and environment. CF-adapted WAM uses neither grounded intent conditioning nor FG post-training; CF + grounded intent adds intent conditioning, and complete LIFT-WAM additionally includes FG. This full table supports the base-to-complete, intent-conditioning, and task-wise FG comparisons summarized in Section 4.

Table 8: **LIBERO-10 execution-stage counts.** Every event count uses all 50 rollouts.

Configuration	Grasp	Lift	Success	Success/Lift
CF-adapted WAM	30/50	28/50	22/50	78.6%
CF-adapted WAM + FG	31/50	28/50	25/50	89.3%
CF + grounded intent	36/50	31/50	23/50	74.2%
LIFT-WAM	36/50	32/50	27/50	84.4%

Table 9: **Complete RoboTwin CF results.** Entries show successes/trials and percentages. Bold includes all within-row ties. CF-adapted WAM uses neither grounded intent nor FG. CF + grounded intent adds the interface without FG.

Setting	Base policy	CF-adapted WAM	CF + grounded intent	LIFT-WAM
Clean environment				
Row order	0/40 (0%)	2/40 (5%)	2/40 (5%)	**5/40 (12.5%)**
Stack order	0/40 (0%)	**20/40 (50%)**	15/40 (37.5%)	17/40 (42.5%)
Left to right	3/40 (7.5%)	8/40 (20%)	17/40 (42.5%)	**20/40 (50%)**
Right to left	3/40 (7.5%)	17/40 (42.5%)	18/40 (45%)	**20/40 (50%)**
Slot reassignment	0/40 (0%)	5/40 (12.5%)	**15/40 (37.5%)**	10/40 (25%)
Total / macro mean	6/200 (3%)	52/200 (26%)	67/200 (33.5%)	**72/200 (36%)**
Random environment				
Row order	0/40 (0%)	0/40 (0%)	0/40 (0%)	**2/40 (5%)**
Stack order	0/40 (0%)	**17/40 (42.5%)**	12/40 (30%)	13/40 (32.5%)
Left to right	0/40 (0%)	5/40 (12.5%)	**18/40 (45%)**	**18/40 (45%)**
Right to left	0/40 (0%)	**12/40 (30%)**	**12/40 (30%)**	8/40 (20%)
Slot reassignment	0/40 (0%)	3/40 (7.5%)	13/40 (32.5%)	**15/40 (37.5%)**
Total / macro mean	0/200 (0%)	37/200 (18.5%)	55/200 (27.5%)	**56/200 (28%)**

Aggregation and uncertainty. For configuration c and environment e, the macro-average is

$$\mathrm{MacroSuccess}_{c,e} = \frac{1}{5}\sum_{s=1}^{5}\frac{100k_{c,e,s}}{40} = \frac{100\sum_{s=1}^{5}k_{c,e,s}}{200}. \tag{32}$$

Thus, one successful episode changes a task rate by 2.5 pp and the five-setting mean by 0.5 pp. Table 10 uses the aggregate Wilson approximation in Equation 31. The 200 episodes do not represent 200 independent task settings or independently trained models.

Task-wise FG differences. Figure 7 compares complete LIFT-WAM with CF + grounded intent within each environment. In the order row order, stacking, left-to-right placement, right-to-left placement, and slot reassignment, the Clean differences are $(+7.5, +5.0, +7.5, +5.0, -12.5)$ pp, and the Random differences are $(+5.0, +2.5, 0.0, -10.0, +5.0)$ pp.

The mean differences are $+2.5$ pp in Clean and $+0.5$ pp in Random, corresponding to five and one additional successful episodes, respectively. In Random, gains of two successes in row order, one in stacking, and two in slot reassignment are partly offset by four fewer successes in right-to-left placement. Left-to-right placement is unchanged.

The main 200-episode RoboTwin benchmark and the 50-episode behavioral diagnostic in Section 4.6 are reported separately. Table 3 provides lift and goal-outcome counts for the diagnostic evaluation.

B.6 RoboTwin Random Behavioral Metrics and Failure Summaries

Evaluation scope and event definitions. The behavioral diagnostic in Section 4.6 contains 50 episodes per configuration under counterfactual instructions. Its success rates of 0%, 20%, 28%, and 30% describe this evaluation separately from the main 200-episode RoboTwin benchmark. For episode i, let C_i indicate achievement of the benchmark CF goal, S_i indicate at least one source-goal

Table 10: **RoboTwin macro success and approximate 95% Wilson intervals.** Entries show success rate and interval endpoints in percent. Each environment uses 200 episodes per configuration. Intervals use an aggregate binomial approximation and do not account for task clustering or training-seed variability.

Configuration	Clean	Random
Base policy	3.0 [1.4, 6.4]	0.0 [0.0, 1.9]
CF-adapted WAM	26.0 [20.4, 32.5]	18.5 [13.7, 24.5]
CF + grounded intent	33.5 [27.3, 40.3]	27.5 [21.8, 34.1]
LIFT-WAM	36.0 [29.7, 42.9]	28.0 [22.2, 34.6]

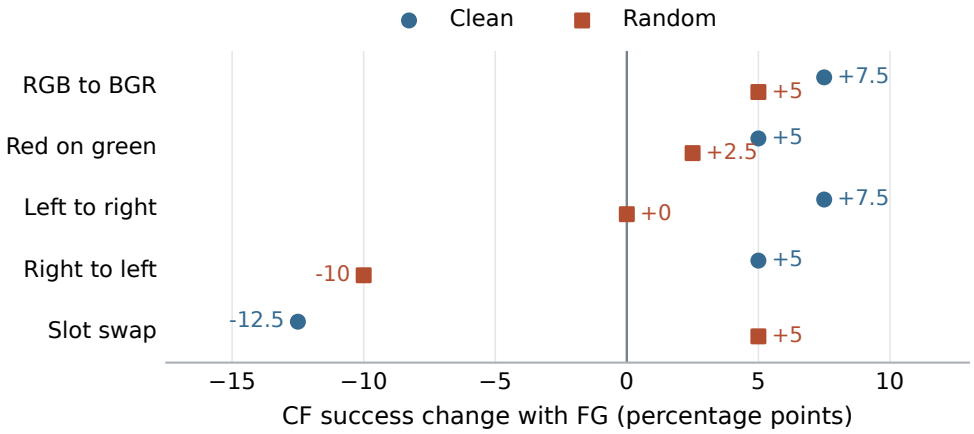

Figure 7: **Task-wise FG differences on RoboTwin.** Each point subtracts the success rate of CF + grounded intent from that of complete LIFT-WAM within the same environment setting. Each configuration has 40 episodes per setting. Positive values favor the FG configuration. The differences compare aggregate counts and are not paired episode transitions.

success event, and L_i indicate a recorded lift of at least one tracked instruction-related object. The mutually exclusive goal-outcome counts are

$$n_{\text{CF}} = \sum_{i=1}^{N} C_i, \tag{33}$$

$$n_{\text{source-only}} = \sum_{i=1}^{N} (1 - C_i) S_i, \tag{34}$$

$$n_{\text{neither}} = \sum_{i=1}^{N} (1 - C_i)(1 - S_i). \tag{35}$$

These counts sum to $N = 50$. Episodes that achieve both goals during execution are included in CF success. Source-only outcomes do not measure performance under the original instruction.

Table 11: **Two complementary partitions of failed Random episodes.** Entries show counts and percentages among CF failures; the denominator is listed in the second column. Source-only and neither partition failures by goal outcome. Lift and no lift partition the same failures by lift status.

		Goal outcome		Lift status	
Configuration	Failures	Source-only	Neither	Lift	No lift
Base policy	50	48 (96.0%)	2 (4.0%)	50 (100.0%)	0 (0.0%)
CF-adapted WAM	40	4 (10.0%)	36 (90.0%)	13 (32.5%)	27 (67.5%)
CF + grounded intent	36	2 (5.6%)	34 (94.4%)	10 (27.8%)	26 (72.2%)
LIFT-WAM (+FG)	35	2 (5.7%)	33 (94.3%)	9 (25.7%)	26 (74.3%)

Lift events and failure accounting. The lift-related joint counts are

$$n_{\text{lift}} = \sum_{i=1}^{N} L_i, \tag{36}$$

$$n_{\text{lift,fail}} = \sum_{i=1}^{N} L_i(1 - C_i), \tag{37}$$

$$n_{\text{lift,CF}} = n_{\text{lift}} - n_{\text{lift,fail}}. \tag{38}$$

Across all configurations, every CF-success episode also contains a recorded lift, giving $n_{\text{lift,CF}} = n_{\text{CF}}$. The total number of failures is $n_{\text{fail}} = N - n_{\text{CF}}$, and failures without a recorded lift are $n_{\text{no-lift,fail}} = n_{\text{fail}} - n_{\text{lift,fail}}$. Lift records object interaction, while CF success records achievement of the requested goal. Both are episode-level events, and configuration differences are reported descriptively from aggregate counts.

From source-goal persistence to goal completion. The base policy records a lift in all 50 episodes, yet achieves no CF goal and completes only the source goal in 48 episodes. Object manipulation alone therefore does not establish adherence to the changed instruction. CF adaptation reduces source-only outcomes from 48 to 4 and produces 10 successful CF executions, showing a substantial reduction in source-goal persistence alongside successful execution of alternative goals.

Grounded intent further increases CF success from 10 to 14 episodes and reduces source-only outcomes from 4 to 2. Full-goal corrective training raises CF success to 15 episodes. Across the CF-adapted, intent-conditioned, and complete configurations, recorded lift counts are 23, 24, and 24, while lift-without-CF-success counts decrease from 13 to 10 to 9. The added conditioning and supervision thus yield more completed target goals with similar overall frequencies of recorded object interaction.

These behavioral outcomes complement the target-specific measurements on LIBERO-10. Without FG, grounded intent raises target grasp from 60% to 72%. Adding FG to the intent-conditioned policy raises complete success from 46% to 54%, while target lift changes from 62% to 64%. Together, the results support the progression described in the Introduction: counterfactual adaptation reduces persistence with scene-associated goals, grounded intent improves target-directed manipulation, and corrective supervision through goal completion improves realization of the requested outcome.

B.7 Language-Pathway Diagnostics on RoboTwin

Matched-scene protocol. The diagnostic includes five tasks under `demo_randomized`, ten matched scenes per task, and noise seeds 42, 43, and 44. Initial physical states, RGB observations, and joint observations are matched across instruction combinations. Both goals are initially false and admit feasible expert continuations. Each model and instruction combination has 150 executions from 50 scenes; the 30 executions per task are three repetitions of ten scenes. Instructions use predetermined templates. This diagnostic is reported separately from the 200-execution-per-environment RoboTwin comparison in Table 1.

Diagnostic checkpoint. Both released Fast-WAM and the CF-adapted diagnostic policy disable structured-intent conditioning. Relative to its directly loaded parent, the diagnostic checkpoint's fi-

Table 12: **Branch-wise language intervention on RoboTwin.** Every result measures the CF goal, including rows containing S. Entries give successful executions out of 150, with percentages in parentheses. Ever denotes achievement at any point during a rollout; Final denotes achievement at termination. Each condition contains 50 matched scenes with three noise repetitions per scene. The adapted policy uses no structured-intent conditioning.

Instruction		Released	CF-adapted WAM	
Video	Action	CF Ever	CF Ever	CF Final
S	S	0/150 (0.0%)	1/150 (0.7%)	0/150 (0.0%)
CF	S	0/150 (0.0%)	75/150 (50.0%)	50/150 (33.3%)
S	CF	0/150 (0.0%)	4/150 (2.7%)	4/150 (2.7%)
CF	CF	1/150 (0.7%)	**80/150 (53.3%)**	**80/150 (53.3%)**

Table 13: **CF effects of changing video language from S to CF.** Action language and its termination rule remain fixed at S. Values are percentage-point changes with 95% scene-bootstrap intervals. Each task contains ten scenes and three noise repetitions per scene.

Task	Δ CF Ever	Δ CF Final
Row-order reversal	0.0 [0.0, 0.0]	0.0 [0.0, 0.0]
Stacking-role reversal	73.3 [56.7, 86.7]	30.0 [10.0, 50.0]
Left-to-right placement	66.7 [50.0, 80.0]	50.0 [36.7, 63.3]
Right-to-left placement	83.3 [70.0, 96.7]	70.0 [50.0, 90.0]
Tray-slot reassignment	23.3 [10.0, 40.0]	16.7 [3.3, 33.3]

nal 200 training steps leave all 604 video LoRA tensors unchanged and change all 604 action LoRA tensors. The inherited video LoRA remains active. The interventions characterize the resulting policy's language pathway, rather than isolating the contribution of those final steps to video representations. This history differs from the final LIFT-WAM training configuration, which jointly updates the Adapter and video/action LoRA (Appendix B.3).

CF outcome definitions and controlled comparisons. CF Ever records whether the benchmark CF predicate is satisfied at any point during execution; CF Final records whether it is satisfied at termination. Both are reported as integer counts k/N and empirical percentages $100k/N$. The symbol S denotes an input instruction, never the target of an outcome column in these diagnostics. The primary contrasts compare (S, CF) with (CF, CF) and (S, S) with (CF, S), where the first entry is video language. Each contrast fixes action language and its associated termination rule. Changing action language also changes termination, so comparisons across action-language conditions do not isolate the action-text pathway. In particular, the Ever–Final gap under (CF, S) is not used to estimate the benefit of full-goal training.

Full branch-wise control matrix. Table 12 reports all four video/action instruction combinations and only CF outcomes. With action language fixed at CF, the adapted policy improves from 4/150 to 80/150 CF achievements when video language changes from S to CF. With action language fixed at S, CF Ever rises from 1/150 to 75/150, and CF Final from 0/150 to 50/150. The four combinations across two models yield 1,200 closed-loop executions. These executions are separate from the ten-state WM prediction probes in Appendix B.9.

Scene-paired effects. For each scene, the three noise-seed differences between (CF, S) and (S, S) are averaged before resampling the ten scenes within each task 5,000 times. Table 13 gives the resulting mean changes and 95% bootstrap intervals in percentage points, without multiple-comparison adjustment. Positive effects occur in four tasks. The degenerate sorting interval reflects zero observed differences in every sampled scene.

Representation controls and KV recombination. Hidden-state and KV probes cover 280 states per model and compare source and CF instructions, paraphrases of both, empty task text, and repeated source instructions. Repeating the same instruction produces identical results. Empty task

Table 14: **CF execution diagnostics under aligned (CF, CF) instructions.** All entries are raw counts for the adapted diagnostic policy. All lifted concerns the objects specified by the CF instruction; Strict denotes the lift-then-place criterion. Each task has 30 executions. CF Ever and CF Final coincide in this condition.

Task	All lifted	CF goal	Strict
Row-order reversal	12/30	1/30	0/30
Stacking-role reversal	29/30	27/30	6/30
Left-to-right placement	26/30	19/30	17/30
Right-to-left placement	28/30	26/30	14/30
Tray-slot reassignment	26/30	7/30	3/30
Total	121/150	80/150	40/150

Table 15: **CF-future object-region velocity MSE at $\sigma = 1$.** Entries are adapted-minus-released differences under CF language, with reported 95% intervals. Negative values indicate lower error. Sorting and stacking use initial states; the remaining tasks use shared decision states.

Task	Δ CF-future MSE [95% CI]
Row-order reversal	-0.501 [-0.704, -0.290]
Stacking-role reversal	-0.842 [-0.968, -0.690]
Left-to-right placement	0.004 [-0.051, 0.061]
Right-to-left placement	-0.017 [-0.073, 0.039]
Tray-slot reassignment	-0.025 [-0.039, -0.014]

text retains the default prefix and proprioception. In the adapted policy, final-layer relative L_2 changes at initial sorting and stacking states are 0.218 and 0.161, exceeding the respective paraphrase changes. Placement effects vary with execution phase. Spatial feature differences describe representation changes, not attention weights.

Independent source/CF video-KV and action-text combinations yield 6,720 offline action outputs. With source action text fixed, the adapted policy's projection changes along the source-expert-to-CF-expert action direction are 0.791 and 0.731 at initial sorting and stacking states, and 0.755, 0.745, and 0.303 at shared decision states for left-to-right placement, right-to-left placement, and tray-slot reassignment. Initial placement windows have identical expert futures, so directional comparisons use the shared decision states. These projections measure goal-directed changes in actions.

Goal achievement and physical execution. Under aligned (CF, CF) instructions, the adapted policy lifts all CF-specified objects in 121/150 executions, achieves the benchmark CF goal in 80/150, and meets the stricter lift-then-place criterion in 40/150 (Table 14). The stricter criterion checks lifting before placement together with target geometry and release/contact conditions. These are distinct predicates; their aggregate counts do not define a stage-transition funnel. The task differences show that language-directed actions and complete physical execution remain distinct challenges. FG's incremental effect is assessed by the dedicated training comparisons in the main paper.

Future-video diagnostics. Denoising probes fix observations, noisy video inputs, and noise seeds at $\sigma \in \{0.2, 0.5, 0.8, 1.0\}$. Velocity MSE excludes the initial frame and padding. Table 15 reports object-region errors for CF-expert futures under CF language at $\sigma = 1$. CF-future fitting improves for sorting and stacking, whereas the intervals for both placement directions cross zero. These measurements evaluate denoising accuracy.

The generation study contains 3,360 nine-frame clips. A fixed subset of 432 clips, 216 per model, was individually reviewed. All 216 released-model clips and 96 adapted-model clips were judged usable; 120 adapted-model clips exhibited substantial ambiguity or artifacts. Across these diagnostics, improved instruction-following behavior coexists with variable visual quality in decoded future predictions. Generated clips characterize the Video Expert's predictive outputs, while closed-loop interventions evaluate the control value of its language-conditioned representations. Together, these measurements provide complementary views of the Video Expert's role in WAMs.

Table 16: LingBot-VA instruction-following results. All rates are percentages; N denotes the number of episodes per instruction condition. Correct and Shuffled measure source-goal completion; CF measures the newly instructed goal. Source@CF records source-goal achievement during CF rollouts.

Benchmark	N	Correct	Shuffled	CF	Source@CF
RoboTwin Clean	200	98.5	94.5	6.5	94.0
RoboTwin Random	200	93.0	90.5	3.5	91.5
LIBERO-10	50	98.0	74.0	20.0	68.0
LIBERO-Object	50	4.0	0.0	10.0	4.0

B.8 Instruction-Following Evaluation of LingBot-VA

Evaluation protocol. We evaluate LingBot-VA (Li et al., 2026) as an additional WAM reference. Each initial state is evaluated under three instruction conditions: *Correct*, *Shuffled*, and *Counterfactual* (CF). RoboTwin includes five source tasks, each evaluated on 40 paired scenes in both Clean and Random settings, yielding 400 episodes per instruction condition. Each LIBERO suite includes ten tasks with five fixed initial states per task, using seed 42, yielding 50 episodes per condition. The evaluation therefore comprises 1,500 rollouts across the three instruction conditions.

Metrics. Under Correct instructions, we measure completion of the source goal. Under Shuffled instructions, we also report source-goal completion, which measures persistence of the original behavior under a changed instruction. Under CF instructions, the primary metric is completion of the newly instructed goal. We additionally record whether the source goal is achieved at any point during the CF rollout, denoted by *Source@CF*. Source@CF and CF-goal success are not mutually exclusive. Shuffled rollouts terminate upon source-goal completion, so their alternative-goal counts are not treated as directly comparable to CF-goal success.

Aggregate results. Table 16 reports the results under all three instruction conditions. Across RoboTwin Clean and Random, LingBot-VA completes the source goal in 383/400 Correct rollouts (95.75%) and 370/400 Shuffled rollouts (92.50%), but completes the CF goal in only 20/400 rollouts (5.00%). The source goal is still achieved in 371/400 CF rollouts (92.75%). On LIBERO-10, Correct success is 98.0%, while CF success is 20.0% and Source@CF is 68.0%. Together, these results reveal a gap between executing the source task and redirecting execution toward an alternative instructed goal.

Task-level results. Tables 17 and 18 provide the per-task counts. All 20 RoboTwin CF successes occur in the two placement tasks; stacking, ordering, and slot swapping yield no CF successes across their 240 rollouts. Likewise, the ten LIBERO-10 CF successes are concentrated in task IDs 0 and 1, with no CF successes on the remaining eight tasks. Thus, aggregate CF performance reflects success on a subset of goal changes rather than consistent instruction following across tasks.

LIBERO checkpoint and interpretation. Both LIBERO suites are evaluated using the LingBot-VA Long checkpoint in a cross-suite setting. On LIBERO-Object, Correct success is only 2/50 (4.0%), and the task manifest does not provide strict-conflict annotations. Its CF result is therefore reported as an additional reference rather than used independently to attribute failure to instruction following. The RoboTwin and LIBERO-10 diagnostics provide clearer evidence of the gap between source-task execution and alternative-goal execution.

B.9 Instruction Interventions in World-Model Prediction

Protocol and sample units. This experiment compares the same released and CF-adapted diagnostic checkpoints used in Appendix B.7, without structured intent or an FG intervention. It performs no training, action sampling, or robot rollout. Five RoboTwin tasks under `demo_randomized` contribute two preselected valid scenes each, giving ten shared states. Sorting and stacking use initial states; both placement directions and slot reassignment use common post-

Table 17: Per-task LingBot-VA results on RoboTwin. Every row contains 40 episodes per condition. C and S denote source-goal success under Correct and Shuffled instructions. CF denotes counterfactual-goal success, and Src. denotes Source@CF. All entries are successful-episode counts. Source-task identifiers are retained for traceability.

Source task	Setting	C	S	CF	Src.
`place_a2b_left`	Clean	39	35	6	33
	Random	36	31	5	31
`place_a2b_right`	Clean	40	35	7	37
	Random	33	32	2	33
`stack_blocks_two`	Clean	40	40	0	40
	Random	40	40	0	40
`blocks_ranking_rgb`	Clean	40	39	0	40
	Random	37	38	0	39
`place_burger_fries`	Clean	38	40	0	38
	Random	40	40	0	40
Total	Clean	197	189	13	188
	Random	186	181	7	183

Table 18: Per-task LingBot-VA results on LIBERO. Each task contains five episodes per condition. C, S, CF, and Src. follow the definitions in Table 17. All entries are successful-episode counts; task IDs are local to each suite.

	LIBERO-10				LIBERO-Object			
Task ID	C	S	CF	Src.	C	S	CF	Src.
0	5	0	5	0	0	0	0	0
1	5	0	5	0	0	0	0	0
2	5	5	0	4	2	0	1	1
3	5	5	0	5	0	0	0	1
4	4	4	0	4	0	0	0	0
5	5	5	0	5	0	0	0	0
6	5	5	0	3	0	0	4	0
7	5	3	0	4	0	0	0	0
8	5	5	0	4	0	0	0	0
9	5	5	0	5	0	0	0	0
Total	49	37	10	34	2	0	5	2

grasp decision states before the expert trajectories diverge. RGB observations and joint states are identical across instructions, while the two expert continuations differ.

Noise seeds 42, 43, and 44 yield 60 model/state/noise groups. Each group includes two expert references, two noise levels ($\sigma = 0.5, 1$), and 21 conditions: 5,040 fixed-input measurements in total, plus 7,200 layer-residual records. These are condition measurements, not independent scenes. Statistics first average three noise repeats within each scene, then average the two scenes per task. Reported ranges are the minimum and maximum of the two scene means, not confidence intervals. At $\sigma = 1$, the two expert references share the same noise input and differ in their reference errors; they do not supply additional independent samples. The checkpoint history described in Appendix B.7 applies: the final 200 steps did not update the inherited video LoRA.

Text-to-video conditioning and matched inputs. The observed context contains 128 text positions and one robot-state token, each with dimension 4,096. After context projection, the video cross-attention layers obtain keys and values from this context and queries from noisy video tokens. Their projected attention outputs are added residually to the video representation. All interventions hold the initial observation, robot state, and noisy video input fixed. Text masking blocks all text positions, including the fixed prefix and padding, while retaining the robot-state token. Empty task text retains the fixed prefix and is a different intervention.

Table 19: **Goal changes and paraphrases at $\sigma = 1$.** Entries are future-velocity RMS differences under matched inputs. Each paraphrase is compared with its own original instruction. Each task has two scenes and three noise repetitions per scene.

Task	Goal change	S paraphrase	CF paraphrase
Released Fast-WAM			
Row-order reversal	0.0165	0.0184	0.0173
Stacking-role reversal	0.0187	0.0184	0.0178
Left-to-right placement	0.3977	0.0231	0.0364
Right-to-left placement	0.2707	0.0204	0.0468
Tray-slot reassignment	0.0187	0.0202	0.0253
CF-adapted WAM			
Row-order reversal	0.2072	0.1285	0.0735
Stacking-role reversal	0.3422	0.0370	0.0484
Left-to-right placement	0.1667	0.0301	0.0296
Right-to-left placement	0.0961	0.0281	0.0273
Tray-slot reassignment	0.1165	0.0421	0.0281

Table 20: **CF-reference language-matching margins.** Positive values favor CF language against the same CF expert continuation. Entries are scene-averaged paired velocity-MSE differences, not success rates.

	$\sigma = 1$		$\sigma = 0.5$	
Task	Released	CF-adapted	Released	CF-adapted
Row-order reversal	+0.00094	+0.24833	+0.00105	+0.06050
Stacking-role reversal	-0.00073	+0.49515	+0.00007	+0.01310
Left-to-right placement	+0.13586	+0.02836	+0.00789	+0.00904
Right-to-left placement	+0.10053	+0.02650	-0.00383	+0.00472
Tray-slot reassignment	+0.00136	-0.04326	-0.00025	-0.00610

Goal changes versus paraphrases. Table 19 compares future-velocity RMS changes under a goal change and under paraphrases of each instruction at $\sigma = 1$. Each paraphrase is compared with its own original wording. For the adapted model, goal changes exceed both paraphrase changes in all five tasks. The released model already exhibits strong goal-related responses for both placement directions. At $\sigma = 0.5$, the adapted model's right-to-left goal-change RMS is approximately 0.0460, closer to its paraphrase values of 0.0401 and 0.0419. The stacking instructions differ in both the initial transport description and the final stacking relation.

CF-reference alignment. Equation 5 compares two language conditions of the same model against one CF expert continuation. The object region Ω is fixed by the union of segmentation regions for manipulated entities in the two expert trajectories; generated outputs do not select the region. Velocity errors exclude the initial frame and padding. Background regions are measured as controls. Positive Δ_{CF} means that CF language gives lower error against the CF continuation. Tables 20 and 21 provide means and scene ranges.

Sorting, stacking, and both placement directions have positive adapted means at both noise levels. Slot reassignment has negative CF-reference margins in the denoising probes. In the complementary closed-loop diagnostic, changing video language from source to CF increases this task's CF achievement by 23.3 percentage points with source action language and its termination rule fixed (Appendix B.7). These results distinguish reference-future matching from the control effect of language conditioning within the Video Expert.

Layer-group masking. We intervene on early (0–9), middle (10–19), late (20–29), or all video cross-attention layers. Table 22 reports the adapted model's CF-reference margins at $\sigma = 1$. Early and middle masking sharply reduce the observed differences, while late masking retains more of the original margin. Full masking makes both instructions' predictions identical. This last result verifies that the text pathway is blocked.

Table 21: **Descriptive scene ranges for CF-reference margins at $\sigma = 1$.** Brackets give the minimum and maximum of two scene means after averaging three noise repetitions within each scene. They are not confidence intervals.

Task	Released	CF-adapted
Row-order reversal	[+0.00028, +0.00161]	[+0.14534, +0.35133]
Stacking-role reversal	[-0.00134, -0.00012]	[+0.30442, +0.68587]
Left-to-right placement	[+0.02530, +0.24643]	[+0.01703, +0.03970]
Right-to-left placement	[+0.03436, +0.16669]	[+0.01427, +0.03872]
Tray-slot reassignment	[-0.00038, +0.00310]	[-0.05306, -0.03345]

Table 22: **Layer-group text masking in the adapted model at $\sigma = 1$.** Entries are CF-reference margins. Masking preserves the robot-state token but blocks all text positions in the selected layers.

Task	Unmasked	0–9	10–19	20–29	All
Row-order reversal	+0.24833	+0.00156	+0.02197	+0.10607	+0.00000
Stacking-role reversal	+0.49515	+0.00502	+0.01838	+0.14780	+0.00000
Left-to-right placement	+0.02836	+0.00168	+0.00100	+0.00850	+0.00000
Right-to-left placement	+0.02650	+0.00061	+0.00040	+0.00572	+0.00000
Tray-slot reassignment	-0.04326	-0.00022	+0.00028	-0.00576	+0.00000

Bidirectional residual replacement. While keeping recipient text unchanged, we replace selected layers' cross-attention outputs with donor outputs recomputed under the other instruction at the recipient's current noisy input. Let u_r and u_d be the unmodified recipient and donor velocity predictions, and $u^B_{r\leftarrow d}$ the prediction after replacement in layer group B. We use

$$p_B = \frac{\langle u^B_{r\leftarrow d} - u_r,\, u_d - u_r\rangle}{\|u_d - u_r\|_2^2}. \tag{39}$$

The axis connects the model's own instruction-conditioned predictions, not an expert future. Zero indicates no movement along that axis; one reaches the donor's coordinate along it, without requiring full vector equality. Values above one indicate overshoot. Group projections are neither additive nor fractions of language information. An axis is not reported when baseline source–CF velocity MSE is below 10^{-10}.

Table 23 includes both replacement directions at $\sigma = 1$. Early replacement has a larger effect than late replacement in all five adapted-model tasks; middle layers are also influential in right-to-left placement. Full-layer replacement recovers donor predictions and final generated latents exactly. Natural residual differences and intervention effects need not peak in the same layers: for adapted sorting, object-region residual RMS averages approximately 0.0079/0.0792/0.0994 across early/middle/late groups, despite the stronger early intervention effect. Donor residuals carry context propagated through preceding layers, and changes affect both objects and background.

Generated clips and denoising-time interventions. For the first preselected scene of each task with seed 42, we generate 27 conditions per model: 270 nine-frame clips in total. Each spans 32 action steps with frame stride four. Review covers all 50 comparison panels using head-camera frames 0, 2, 4, 6, and 8, corresponding to action offsets 0, 8, 16, 24, and 32. It is a qualitative review of selected views and times, without blind assessment or physical success evaluation. Selected examples are shown in Appendix B.10. These clips are separate from the 3,360-clip study in Appendix B.7.

The inspected adapted clips frequently remain nearly static or develop blur, ghosting, and object deformation. Residual replacement sometimes changes local motion or appearance. Sustained masking of whole text streams often disrupts global structure; this intervention jointly masks task text, prefix, and padding positions.

Temporal masking uses solver steps 0–5, 6–12, and 13–19, corresponding to stored times 1000–936, 920–768, and 728–208, followed by the final update to the endpoint. These groups span unequal solver-time intervals. The deployed action policy uses current-observation Video KV, so these generated futures are auxiliary diagnostics rather than inputs to control. FG's effects on complete execution remain assessed by the dedicated action-policy comparisons.

Table 23: **Bidirectional residual replacement in the adapted model at $\sigma = 1$.** Entries are projections along the model-prediction axis in Equation 39. $S \to CF$ replaces residuals in an S-text recipient using a CF donor; the reverse direction interchanges these roles. Projections are not expert-action alignment or success rates.

	$S \to CF$			$CF \to S$		
Task	0–9	10–19	20–29	0–9	10–19	20–29
Row-order reversal	0.601	0.421	0.164	0.611	0.480	0.197
Stacking-role reversal	0.982	0.229	0.068	0.961	0.297	0.066
Left-to-right placement	0.665	0.494	0.112	0.605	0.391	0.141
Right-to-left placement	0.442	0.421	0.099	0.610	0.608	0.085
Tray-slot reassignment	1.047	0.264	0.235	1.056	0.187	0.183

Implementation controls. Repeated conditions and self-replacement have zero maximum difference. Auxiliary comparisons with the joint predictor give identical video outputs, and full-layer replacement and masking satisfy the expected output-equality controls. The joint-predictor check evaluates action-head outputs for implementation equivalence. These controls support interpreting the measured changes as effects of the specified video-language interventions.

B.10 Selected Visual Diagnostics of World-Model Predictions

Figures 8–11 complement the main-text quantitative summary in Figure 3 with selected original frames from the generated-video study in Appendix B.9. Stacking illustrates a positive CF-reference margin alongside limited visible motion; spatial placement provides a contrasting text-conditioning example; slot reassignment illustrates the task with a negative adapted-model margin.

All examples use the preselected scene for each task and noise seed 42. Columns show head-camera frames at action offsets 0, 8, 16, 24, and 32. Expert rows are simulator trajectories; model rows are decoded WM predictions from the same observation. S and CF denote input instructions, not outcome measurements. The original sampled frames are preserved, with labels and layout reformatted for reading. These cases illustrate the inspected behaviors, not their population frequencies or physical task-success rates.

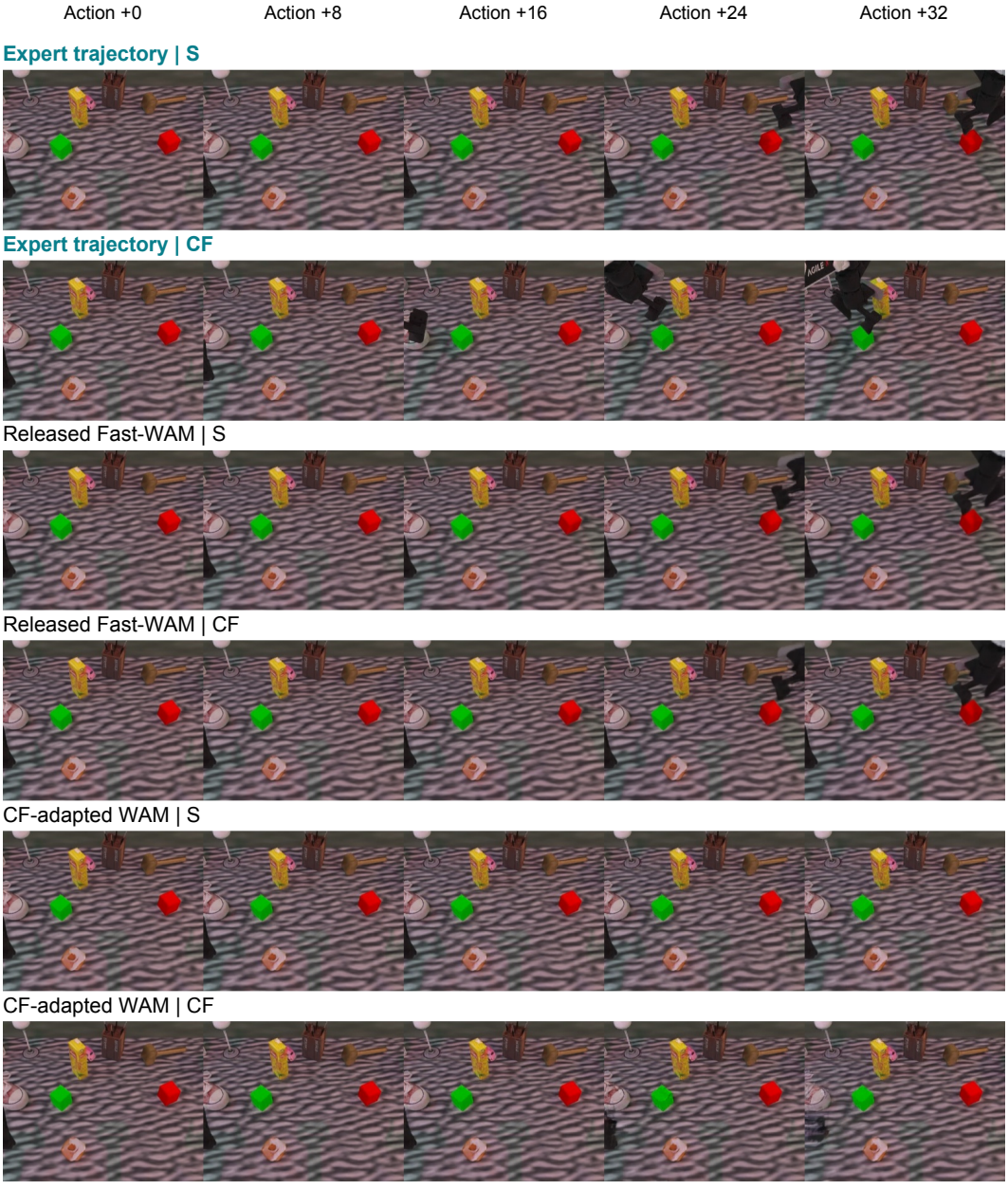

Figure 8: **Stacking: expert continuations and instruction-conditioned predictions.** The experts approach different blocks from the same initial state. Released Fast-WAM approaches the red block under both instructions, while the adapted predictions remain largely static in this window. The example distinguishes the goal-related denoising signal measured in Table 20 from a coherent visible CF continuation. The top two rows are expert trajectories; the remaining rows are WM predictions. The instruction pair also differs in its initial transport description, as discussed in Appendix B.9.



Figure 9: **Placement predictions under source, counterfactual, and empty task text.** The CF expert moves the bell leftward, while the released model's S and CF predictions follow similar downward/rightward paths. Empty task text changes the predicted motion while retaining the fixed prefix. The adapted predictions largely retain the initial pose before developing ghosting. All rows share the post-grasp observation. Expert frames are reference trajectories, and predicted frames are not robot executions.

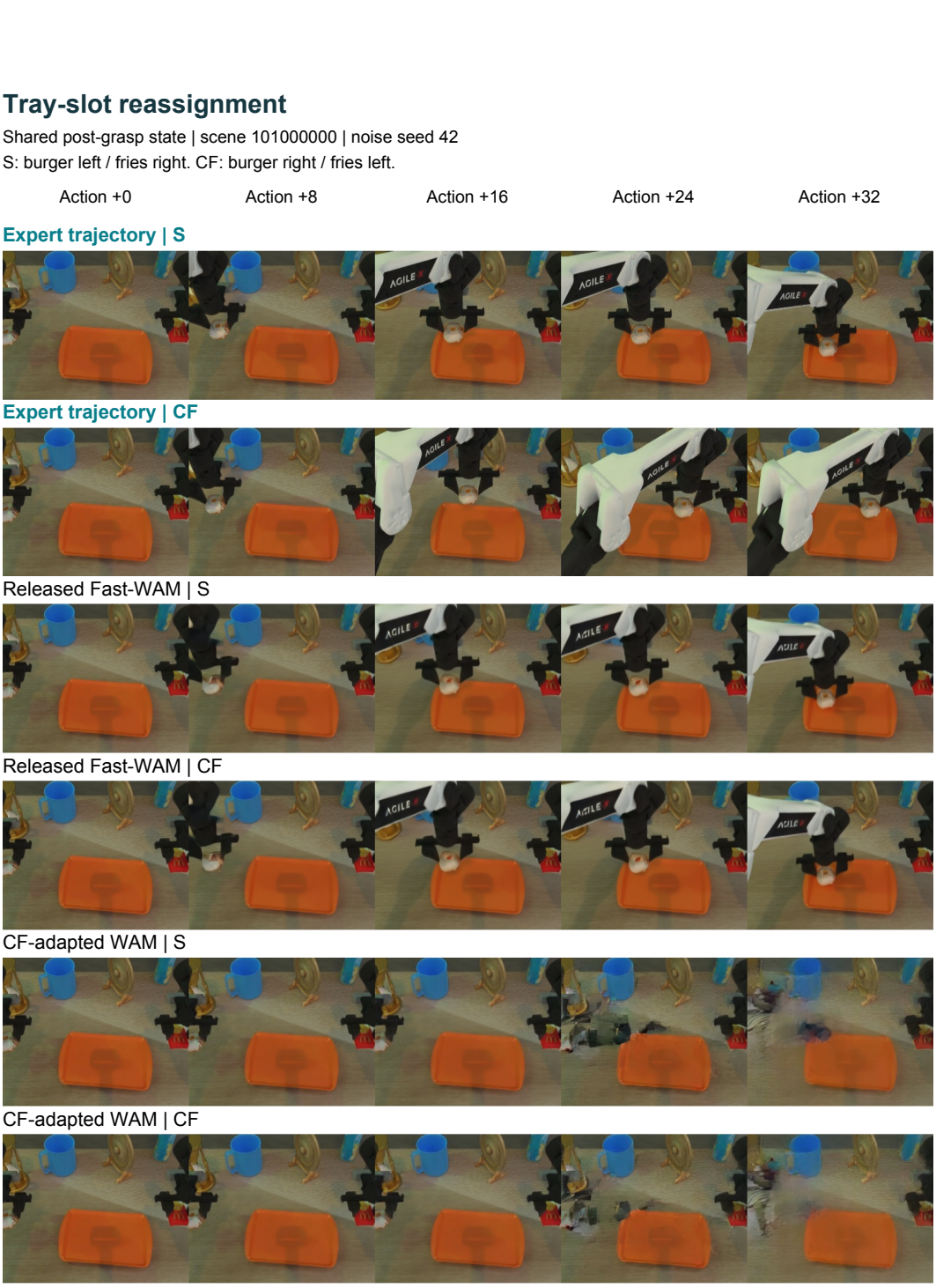

Figure 10: **Slot reassignment: a contrasting goal-alignment case.** The two expert continuations carry the burger toward different tray slots. Released predictions approach the source-side slot under both instructions. The adapted model's late frames blur or lose coherent hand/object structure, without a clear slot reversal. This task also has negative CF-reference margins in Table 20. Only the sampled short window is shown; the complete instruction requires both burger and fries assignments.

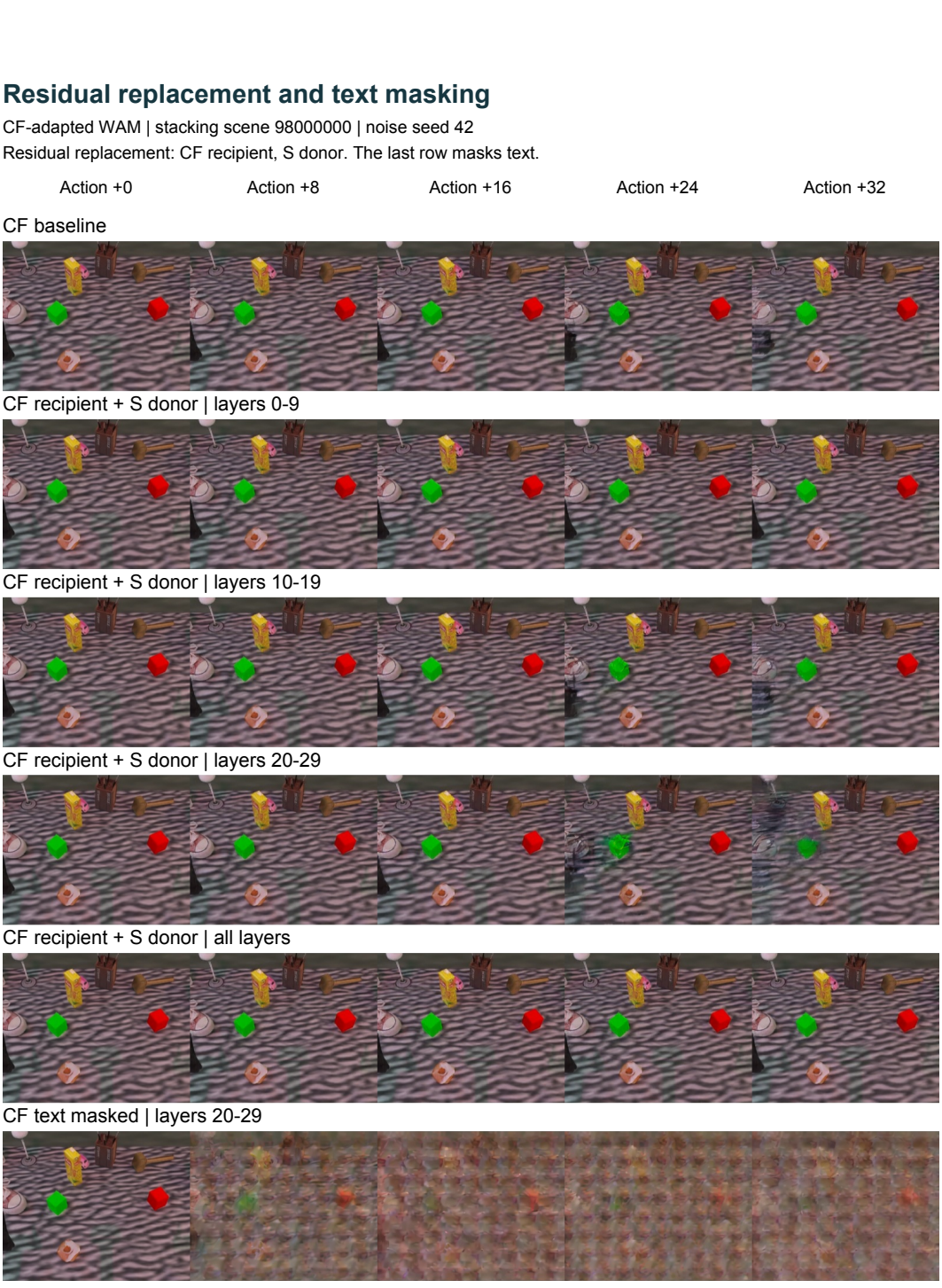

Figure 11: **Residual replacement and text masking in the stacking example.** With CF text retained, selected cross-attention residuals are replaced by S-conditioned donor residuals. Partial replacement can introduce local fragments or green-block ghosting; full replacement recovers the largely static S baseline shown in Figure 8. The last row instead masks text in layers 20–29 and disrupts the broader image structure. All rows use the same input state, seed, view, and sampled time offsets.

B.11 Instruction Swap in Closed-Loop Policy Rollouts

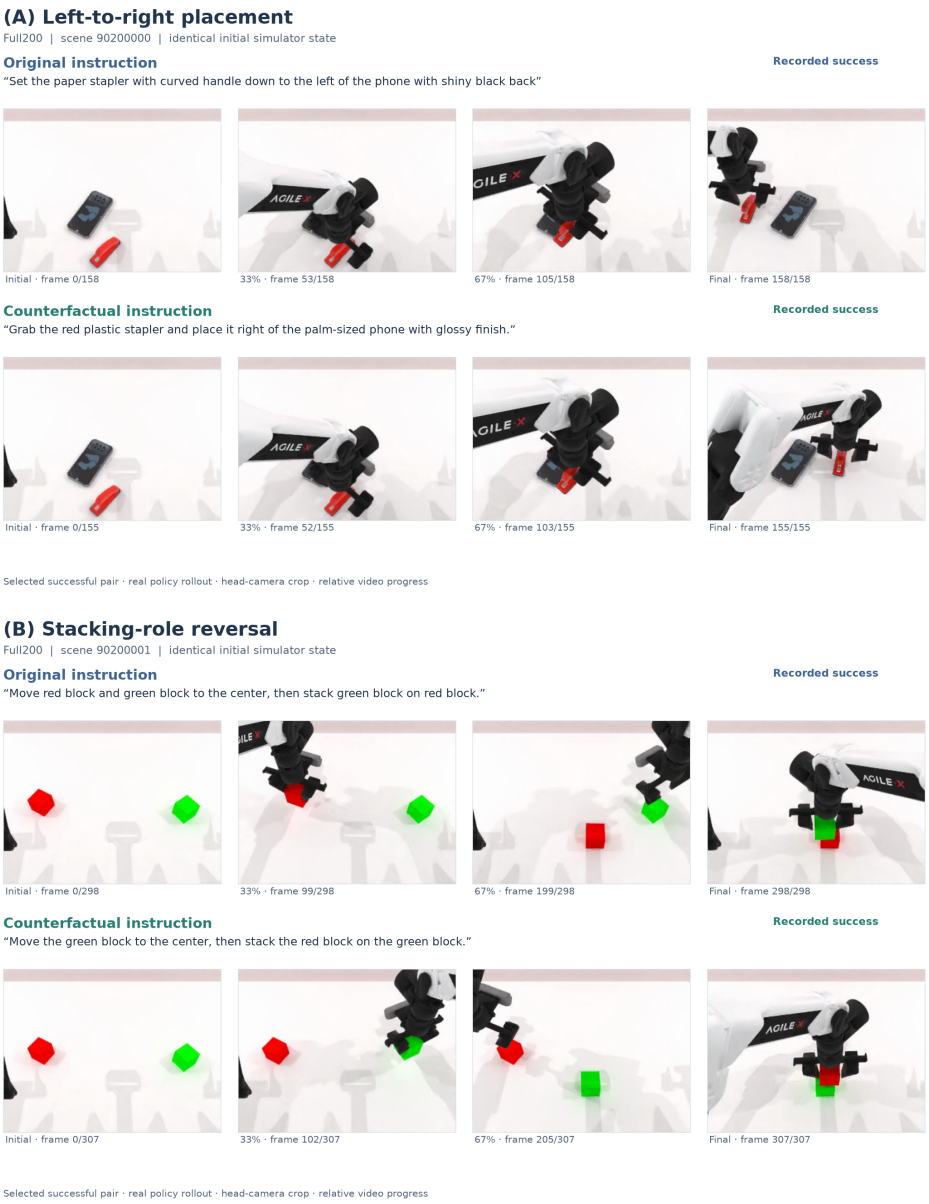

Figure 12: **Paired instruction-swap rollout examples.** Top: left-to-right placement. Bottom: stacking-role reversal. Within each pair, the checkpoint and initial simulator state are identical. Images are recorded closed-loop policy frames at four relative video positions; columns are not synchronized physical times. Cases were selected for recorded success under both instructions and visual clarity.

C Real-World Results and Visualizations

Table 24 reports the per-task results on the UNT David A1C after real-robot fine-tuning. Each of the eight task conditions is evaluated over 10 episodes, for 80 episodes in total.

Real-robot data and fine-tuning. We fine-tune LIFT-WAM on a dataset of 161 teleoperated demonstrations comprising 54,756 frames for the UNT David A1C. Using random seed 42, we split the dataset into 145 training and 16 validation episodes. The recordings contain camera images, robot states, and demonstrated arm and gripper actions, and are converted into episode-based observation–action sequences with timestamps assigned at 15 fps. Images are resized from 640×480 to 256×192 (width $\times$ height), preserving their aspect ratio. Real-robot fine-tuning uses two NVIDIA A800 GPUs with fully sharded data parallelism (FSDP), a per-GPU batch size of 1, and 16 gradient accumulation steps, giving an effective batch size of 32. The learning rate is 1×10^{-4} and the configured training budget is 1,200 optimizer updates. Checkpoints are saved every 100 updates, and the checkpoint at update 300 is used for deployment.

Closed-loop deployment. A local client acquires camera observations and robot states and sends the policy inputs to a remote GPU inference service. In this deployment configuration, the policy predicts a chunk of 16 actions under the task instruction. The client converts the predictions into David A1C arm and gripper commands and executes the first 8 actions at a playback rate of 15 Hz before refreshing the observations for the next inference cycle. The low-level controller runs at 50 Hz and interpolates the arm commands between policy action targets. Remote inference and observation processing contribute additional time between action segments. This closed-loop procedure conditions subsequent predictions on updated observations from physical execution.

Table 24: Real-world evaluation on the UNT David A1C after real-robot fine-tuning. Each task condition is evaluated over 10 episodes. Near and far refer to the filming viewpoint. Task IDs correspond to the appendix visualizations.

ID	Task condition	Successes	SR (%)
Destination selection			
T1	Yellow block into near container	2/10	20
T2	Yellow block into far container	3/10	30
Stacking-role reversal			
T3	Blue block on yellow block	3/10	30
T4	Yellow block on blue block	2/10	20
Target-object selection			
T5	Blue block into container	3/10	30
T6	Yellow block into container	4/10	40
Relative-position placement			
T7	Blue block on near side of yellow	4/10	40
T8	Blue block on far side of yellow	2/10	20
Overall / task macro-average		23/80	28.75

Task definitions. The following descriptions specify the scene, manipulation goal, and intended terminal configuration for each task. Task names are descriptive labels rather than verbatim logged prompts. Near and far are defined from the filming viewpoint. Each task is evaluated over 10 episodes, and task IDs match the results table and figures.

T1: Yellow block into the near container. The scene contains a yellow block and two wooden containers: a near container with four compartments and a far container with two compartments. The robot must grasp the yellow block, transport it to the near container, and release it inside. This task examines destination selection when multiple receptacles are available. The success rate is 20% (2/10).

T2: Yellow block into the far container. The scene again contains a yellow block and both containers, but the target destination is the far, two-compartment container. The robot must pick up

the yellow block and deposit it inside that container. Together, T1 and T2 distinguish two placement destinations for the same manipulated object. The success rate is 30% (3/10).

T3: Blue block on yellow block. A blue block and a yellow block initially rest separately on the table. The robot must grasp the blue block, lift and align it above the yellow block, and release it to form a stack. The intended terminal configuration has the yellow block supporting the blue block after release. This task requires assigning the blue block the manipulated role and the yellow block the supporting role. The success rate is 30% (3/10).

T4: Yellow block on blue block. The scene contains the same two block colors, initially unstacked. The robot must pick up the yellow block and place it on the blue block, producing a stack with yellow above blue after release. Relative to T3, the manipulated and supporting roles are exchanged while the stacking relation is retained. The success rate is 20% (2/10).

T5: Blue block into the container. A blue block, a yellow block, and a wooden two-compartment container are present on the table. The robot must select the blue block, grasp and transport it to the container, and release it inside. The yellow block provides a non-target alternative. This task examines target-object selection with a single available receptacle. The success rate is 30% (3/10).

T6: Yellow block into the container. The scene contains both colored blocks and the wooden container. The robot must select the yellow block and deposit it inside the container, with the blue block serving as the non-target alternative. Together, T5 and T6 distinguish which object should be manipulated while retaining a common placement destination. The success rate is 40% (4/10).

T7: Blue block on the near side of yellow. The blue and yellow blocks initially rest separately on the table. The robot must pick up the blue block and place it on the camera-near side of the yellow block, which serves as the spatial reference. In the intended terminal configuration, both blocks are supported by the tabletop, with the blue block nearer to the camera than the yellow block. The success rate is 40% (4/10).

T8: Blue block on the far side of yellow. The scene again contains the blue and yellow blocks. The robot must move the blue block to the camera-far side of the yellow block and release it onto the tabletop. The intended terminal configuration places the blue block farther from the camera than the yellow block, with both blocks supported by the table. T7 and T8 retain the manipulated and reference objects while reversing the desired relative position. The success rate is 20% (2/10). Figures 13 and 14 illustrate all eight tasks in Table 24. Each row shows initial, intermediate, and final frames from one video; timestamps refer to the source clip. Frames are cropped to the workspace.

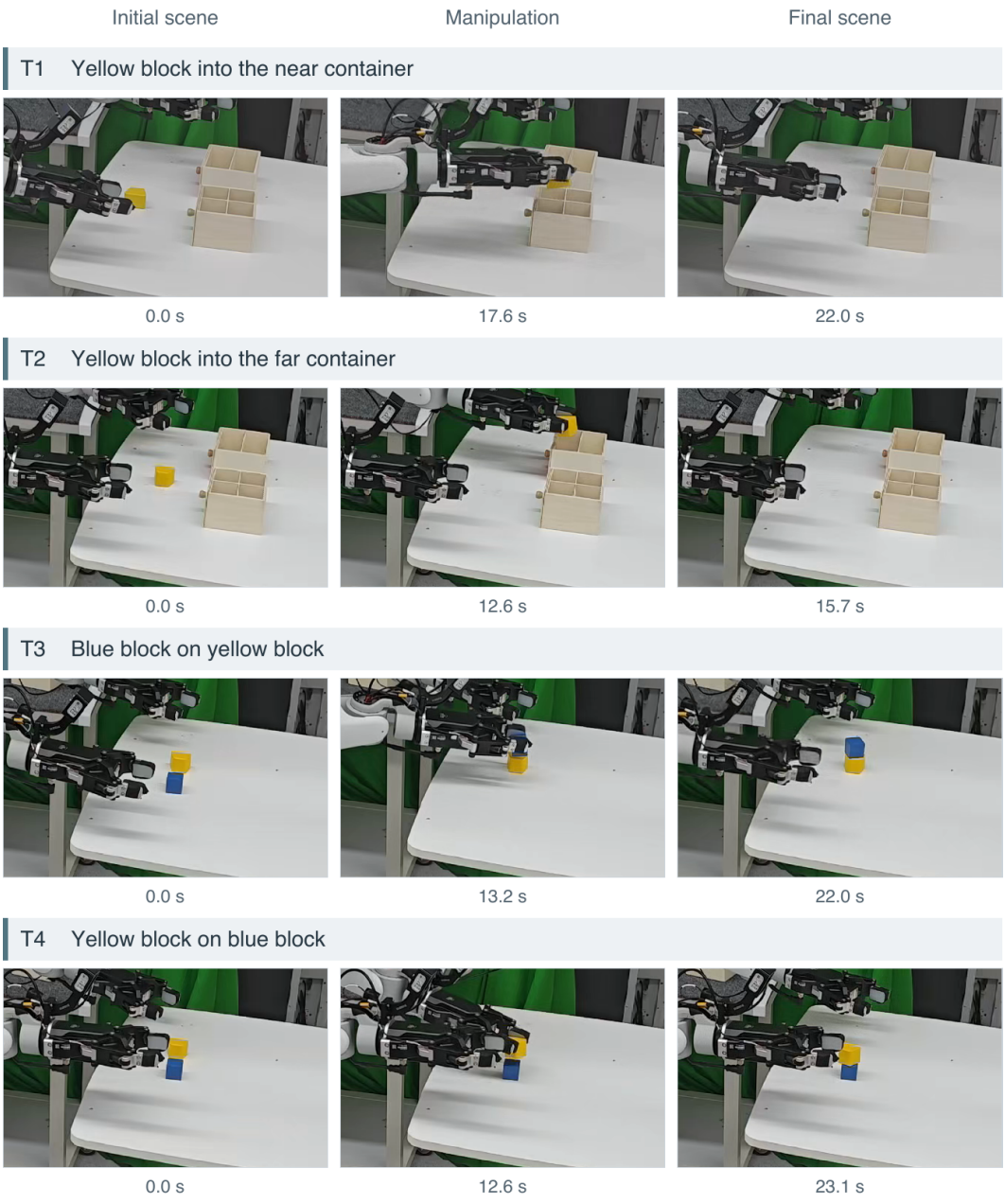

Figure 13: Real-world executions on the UNT David A1C: destination selection (T1–T2) and stacking-role reversal (T3–T4). T1–T2 place the yellow block into different containers; T3–T4 exchange the roles of the blue and yellow blocks in stacking.

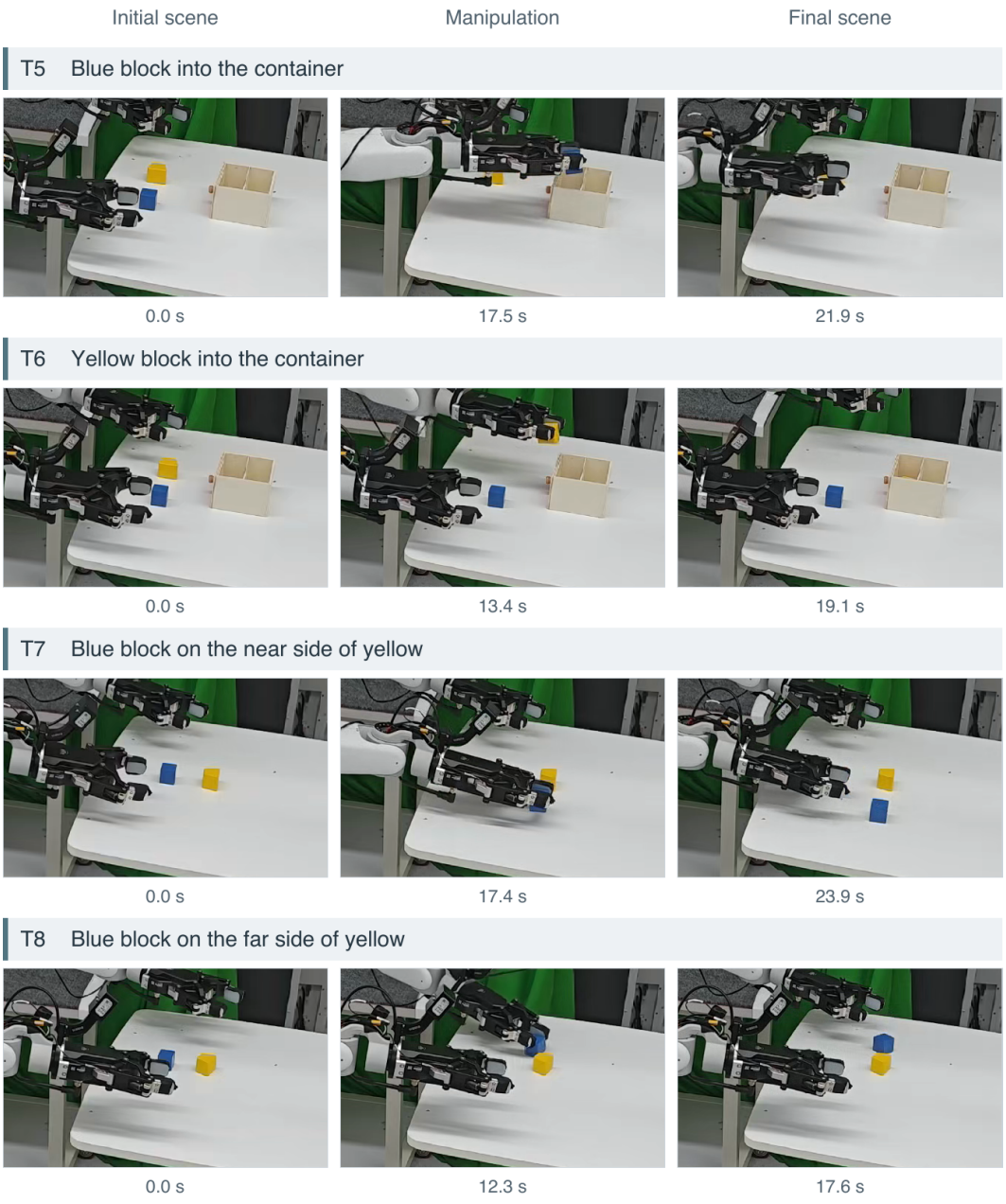

Figure 14: Real-world executions on the UNT David A1C: target-object selection (T5–T6) and relative-position placement (T7–T8). T5–T6 place different blocks into a common container; T7–T8 place the blue block on opposite sides of the yellow block.