Counterfactual Instruction Grounding for Vision-Language-Action Models

Shiyu Liu[1,2] **Yu Zhou**[1,2,*] **Meng Liu**[3,2] **Xuanming Guo**[4,2] **Liqiang Nie**[5,2,*]

[1]VCIP & TMCC & DISSec, College of Computer Science &
College of Cryptology and Cyber Science, Nankai University
[2]Zhongguancun Academy [3]Shandong University [4]Jilin University
[5]Harbin Institute of Technology (Shenzhen)

s-lsyu25@bza.edu.cn yzhou@nankai.edu.cn
mengliu.sdu@gmail.com guoxm24@mails.jlu.edu.cn
nieliqiang@gmail.com

Abstract

Vision-Language-Action (VLA) models inherit rich semantic priors from pre-trained vision-language models, yet these priors do not guarantee persistent instruction sensitivity after narrow-domain fine-tuning. We observe a systematic failure mode: although VLA models remain visually reactive, their later actions can become weakly constrained by the commanded instruction, leading to unstable object interaction and visually plausible but instruction-inconsistent behavior. We trace this behavior to a collapse of instruction-conditioned representations across network depth and task progress. More fundamentally, this collapse is enabled by the learning objective itself: standard imitation learning admits Bayes-optimal solutions that are invariant to language. We address this by recasting instruction grounding as identifying the commanded instruction among counterfactual alternatives. Based on this view, we propose Counterfactual Instruction Grounding (CIG), a contrastive objective that encourages the generated trajectory to remain identifiable with the commanded instruction among counterfactual alternatives. CIG applies to continuous-regression (CR) and flow-matching (FM) VLA models through a common energy-based scoring interface, using action regression error for CR policies and velocity regression residuals for FM policies without estimating intractable trajectory likelihoods. Practically, CIG can be applied directly to already fine-tuned models as a continuation-stage grounding objective, restoring instruction sensitivity without retraining from scratch or changing the architecture. Extensive experiments on RoboCasa and LIBERO-Plus demonstrate that CIG achieves strong instruction-following performance with success rates of 71.3% and 86.3%, respectively. Real-world deployment further shows that CIG transfers to kitchen manipulation and preserves instruction-conditioned behavior in visually ambiguous scenes. Code is available at `https://github.com/SeeYooLeo/CIG-RoboCasa-Kitchen`.

1 Introduction

VLA models have recently emerged as a promising paradigm for robotic control by combining the semantic priors of large vision-language models (VLMs) with action-generation modules trained on demonstrations. In manipulation tasks, such models can often follow natural-language commands reliably and exhibit strong transfer beyond the exact training distribution. However, much of this

*Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

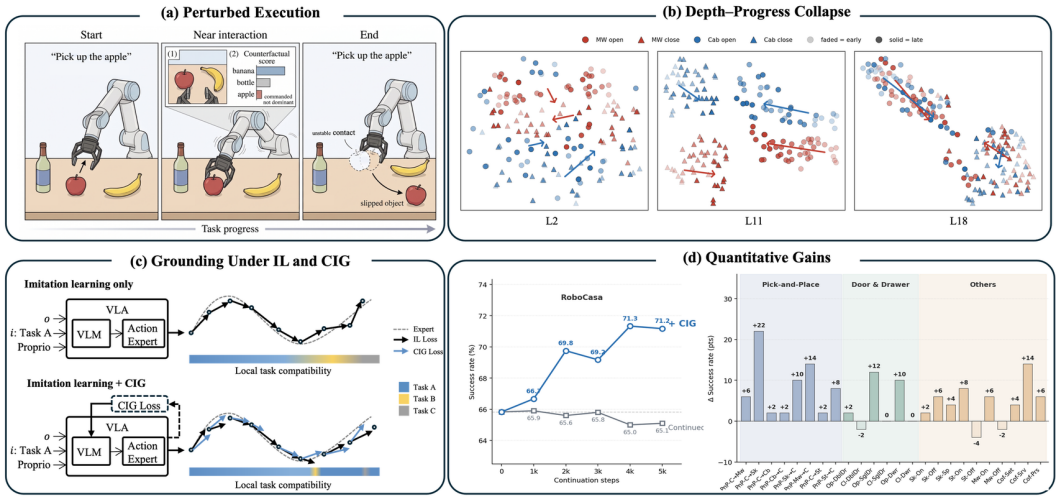


Figure 1: Instruction grounding failure in fine-tuned VLAs and its correction by CIG. (a) As execution progresses, the model drifts from the commanded task to perturbed execution: near interaction, competing alternatives interfere and the final grasp fails. (b) VLM features exhibit instruction-conditioned collapse across both network depth and task progress. (c) Imitation learning keeps trajectories close to expert demonstrations but does not guarantee instruction grounding; CIG adds grounding guidance that preserves local compatibility with the commanded task. (d) CIG yields sustained improvement during continuation training and broad gains across the 24 RoboCasa tasks.

apparent success arises in settings where the instruction is largely redundant with the scene: the target object, affordance, and required action are already strongly implied by observation alone. A harder and more realistic setting is when the same observation supports multiple plausible tasks, so the instruction must determine which behavior the model should execute. In this setting, instruction grounding becomes central: the model should not merely generate plausible behavior, but should consistently privilege the behavior specified by the commanded task.

In this setting, we observe a systematic failure mode in fine-tuned VLA models. A model may initially move toward the commanded target, yet as execution approaches object interaction it becomes increasingly sensitive to nearby alternatives. This raises a central question: **how can a VLA model consistently privilege the commanded task when a single scene admits multiple plausible task interpretations?** Rather than preserving a stable preference for the instruction, the model drifts toward behavior favored by the training distribution, leading to perturbed execution and eventual task failure, as illustrated in Fig. 1a. Importantly, this is not simply a perception problem or a random control error. Instead, it reflects a weakening of instruction grounding precisely when late-stage action refinement matters most.

To understand this failure, we probe instruction-conditioned VLM features across depth and task progress. We find that instruction-sensitive structure is not stably preserved: features may separate at intermediate layers but collapse in deeper layers, and distinct task interpretations drift toward overlap as rollouts progress (Fig. 1b). This suggests that language is not absent, but is not reliably retained where it matters for action generation.

Recent work has started to tackle language-following failures in VLA models through improved conditioning and stronger supervision. For example, OpenVLA-OFT [13] explicitly identifies language-following challenges and introduces FiLM-based [20] language conditioning to mitigate them, while CAST [10] strengthens supervision by augmenting demonstrations with counterfactual language labels. In parallel, diagnostic benchmarks [9, 1] have consistently shown that language information is often weakened or ignored during model execution. Together, these efforts indicate that instruction following remains a core challenge in VLA control. However, they still leave open the failure we study here: most methods do not explicitly require the resulting trajectory to remain identifiable with the commanded instruction. As a result, a model may achieve low imitation loss while relying only weakly on language. Figure 1c summarizes this gap: imitation learning constrains

trajectories to stay close to demonstrations, whereas CIG additionally enforces grounding to the commanded task.

Motivated by this view, we recast instruction grounding as a counterfactual identification problem: given an observation and the resulting behavior, the commanded task should remain distinguishable from alternative task interpretations throughout execution. Based on this formulation, we propose CIG, a contrastive objective that augments imitation learning (IL) with an explicit grounding objective. Rather than replacing imitation learning, CIG complements it by encouraging local behavior along the trajectory to remain semantically attributable to the commanded task instead of drifting toward competing alternatives. This view is agnostic to the specific action-generation family and can be applied to continuous-regression (CR) and flow-matching (FM) VLA models without changing the underlying architecture. Intuitively, imitation loss keeps the trajectory close to expert demonstrations, while CIG ensures that this trajectory remains grounded to the intended task.

We conduct extensive experiments across multiple simulation benchmarks and real-robot settings to evaluate the effectiveness of CIG. Across these evaluations, CIG consistently improves instruction-following performance over continued imitation-learning baselines, yielding gains that are both sustained during training and broadly distributed across tasks. These results suggest that preserving instruction grounding, rather than merely further minimizing imitation loss, is critical for reliable VLA control. Our contributions are summarized as follows:

- **Instruction-grounding failure in fine-tuned VLAs.** We identify and characterize a failure mode in which fine-tuned VLA models drift away from the commanded task, and show that this failure is reflected in instruction-conditioned representations across both network depth and task progress.
- **Counterfactual Instruction Grounding (CIG).** We propose CIG, a contrastive grounding objective that preserves the identifiability of the commanded task by enforcing consistency against counterfactual alternatives on top of imitation learning.
- **Consistent gains in simulation and real-robot settings.** We show that CIG improves instruction-following performance over continued imitation-learning baselines across simulation benchmarks and yields stronger instruction-conditioned behavior in real-robot evaluations.

2 Related Work

Vision-Language-Action models. VLA models integrate visual perception, language understanding, and low-level control for robot manipulation. Early systems such as RT-1 [4] and RT-2 [5] showed the benefit of scaling robot data and pretrained vision-language representations, while recent models such as OpenVLA [12], π_0 [3], and $\pi_{0.5}$ [21] further improve generalization with stronger pretraining and action-generation backbones. Adaptation methods such as OpenVLA-OFT [13] and RoboVLMs [15] improve fine-tuning efficiency and downstream success; further extensions target sequential reasoning, embodied pretraining, and flow-based regularization [24, 18, 26]. Complementary to these advances in pretraining and adaptation, we study instruction grounding as a fine-tuning objective, introducing an auxiliary loss that can be incorporated into both CR and FM policies.

Language neglect in embodied policies. Prior work shows that language-conditioned embodied policies can become weakly dependent on language after fitting narrow training distributions. OpenVLA-OFT [13] identifies language-following failures and strengthens language conditioning, while recent studies report related failures such as memory traps [25], modality collapse [16], and counterfactual cases where visual priors override language [8]. Mechanistic and sparse-autoencoder analyses further trace these effects to how visual, linguistic, and action features are weighted under fine-tuning [11, 22]. Building on these findings, we trace instruction sensitivity across network depth and task progress, relating changes in internal representations to behavioral drift during execution.

Contrastive and counterfactual supervision for instruction grounding. Recent work has begun to strengthen instruction following through contrastive or counterfactual supervision. CAST [10] augments demonstrations with counterfactual language labels to enrich semantic supervision, and Wang et al. [23] introduce synthetic counterfactual perturbations to better align language plans with demonstrations. LangForce [16] similarly points out that when instructions are overly predictable

from observations, the dependence between instructions and actions can collapse, and addresses this through Bayesian decomposition over latent action queries. While these methods enrich semantic supervision through counterfactual labels, perturbed plans, or latent action queries, CIG directly compares candidate instructions against the same demonstrated action chunk. This formulation requires no additional collection of counterfactual trajectories and applies to both CR and FM policies through policy-specific energy scores.

3 Preliminaries

3.1 VLA Policies

Vision-language-action (VLA) policies map visual observations, proprioceptive states, and language instructions to low-level robot actions. We consider two action-generation families: CR and FM. Let $h_t = (o_{\leq t}, a_{<t})$ denote the observation-action history and $\tau_t = a_{t:t+H}$ the action chunk.

Continuous-regression policies. CR policies predict a continuous action chunk $\hat{\tau}_\theta(h_t, i)$ in parallel. We train these policies to match demonstrated actions by minimizing the ℓ_1 distance:

$$\mathcal{L}_{\mathrm{IL}}^{\mathrm{CR}} = \mathbb{E}_{(h_t, i, \tau_t)\sim\mathcal{D}} \left[\left\| \hat{\tau}_\theta(h_t, i) - \tau_t \right\|_1 \right]. \tag{1}$$

This objective directly supervises the predicted action values.

Flow-matching policies. FM policies instead learn a velocity field over action latents. Given an action chunk τ_t and noise ϵ, a noisy latent x_s is constructed by interpolating between noise and data at time $s \in [0, 1]$. The model predicts $v_\theta(x_s, s, h_t, i)$ and is trained with

$$\mathcal{L}_{\mathrm{IL}}^{\mathrm{FM}} = \mathbb{E}_{s,\epsilon} \left[\left\| v_\theta(x_s, s, h_t, i) - u^\star(x_s, s, \tau_t) \right\|_2^2 \right]. \tag{2}$$

FM policies generate continuous actions by integrating the learned velocity field and do not directly provide a normalized likelihood over action chunks.

3.2 Instruction Ambiguity

Both CR and FM policies are adapted with imitation learning, using action regression in Eq. (1) or velocity regression in Eq. (2), respectively. However, imitation learning can fit demonstrations without enforcing discriminative language use: when visual context already predicts the demonstrated behavior, the policy may rely weakly on the instruction.

We study ambiguous states where the same history h_t supports multiple plausible instructions:

$$C(h_t) = \{i^+, i_1^-, \ldots, i_K^-\}, \tag{3}$$

where i^+ is the commanded instruction and the rest are counterfactual alternatives. Instruction neglect occurs when different candidates induce nearly indistinguishable local behavior:

$$\pi_\theta(\tau_t \mid h_t, i) \approx \pi_\theta(\tau_t \mid h_t, i'), \qquad i \neq i',\ i, i' \in C(h_t). \tag{4}$$

The resulting action chunk may remain visually plausible, but is no longer reliably determined by the commanded instruction.

4 Method

4.1 Problem Formulation

Let h_t be the history at step t, τ_t the action chunk generated from h_t, and $C(h_t) = \{i^+, i_1^-, \ldots, i_K^-\}$ a candidate instruction set, where i^+ is the commanded instruction.

Definition 1 (Instruction grounding)**.** *A policy is grounded on $(h_t, C(h_t))$ if the commanded instruction is identifiable from the action chunk:*

$$i^+ = \arg\max_{i \in C(h_t)} s_\theta(h_t, \tau_t, i), \tag{5}$$

where $s_\theta(h_t, \tau_t, i)$ is an instruction-conditioned chunk score.

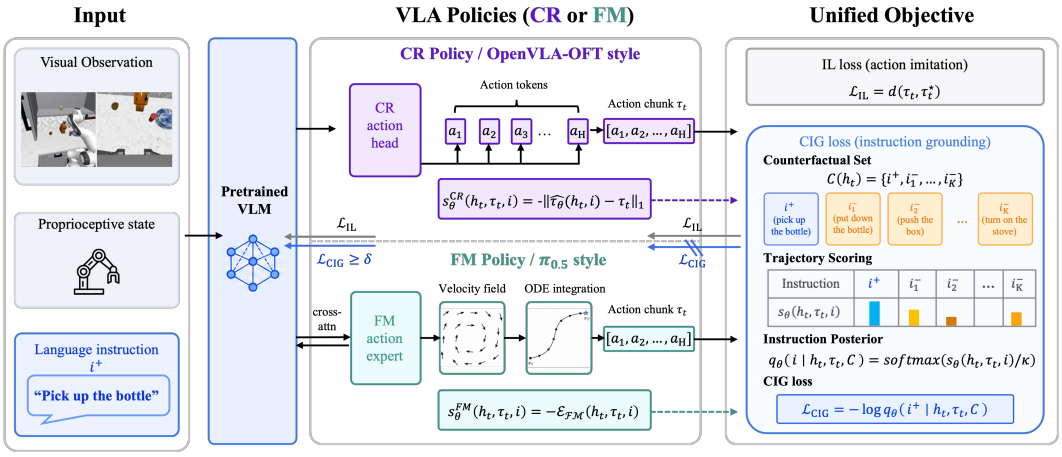


Figure 2: Overview of Counterfactual Instruction Grounding (CIG). Given multimodal inputs, continuous-regression (CR) and flow-matching (FM) VLA policies generate an action chunk and provide an energy-based trajectory score. CR uses negative mean L1 action error; FM uses negative velocity regression residual energy. CIG scores the same chunk against the commanded instruction and counterfactual alternatives, forming a contrastive posterior that enforces instruction identifiability. Imitation learning (IL) preserves action imitation, while CIG supplies a grounding signal routed to the VLM representation and activated when the grounding loss is at least the floor δ.

This separates imitation from grounding: imitation matches expert behavior, whereas grounding asks whether the same chunk is attributable to the commanded instruction.

Why imitation alone is insufficient. Imitation learning supervises the paired chunk (i^+, τ_t) but does not compare i^+ against alternatives in $C(h_t)$. If the chunk is predictable from visual context alone, this permits an instruction-invariant shortcut.

Proposition 1 (Instruction-invariant optima of imitation learning)**.** *Let $p_{\exp}(\tau_t \mid h_t, i)$ be the demonstrator distribution and $\mathcal{L}_{\mathrm{IL}}$ the population imitation objective. Suppose there exists a visual statistic $g(h_t)$ such that*

$$p_{\exp}(\tau_t \mid h_t, i) = p_{\exp}(\tau_t \mid g(h_t)) \quad \textit{a.s. over } (h_t, i). \tag{6}$$

Then any policy with

$$\hat{\tau}_\theta(h_t, i) = m(g(h_t)), \qquad v_\theta(x_s, s, h_t, i) = \bar{u}(x_s, s, g(h_t)), \tag{7}$$

is an optimum of the corresponding $\mathcal{L}_{\mathrm{IL}}$ and is invariant to i, where m and $\bar{u}$ are the conditional median action and conditional mean velocity, respectively.

Remark. Proposition 1 states that low imitation loss does not imply instruction grounding: when visual context determines the chunk, IL has no incentive to distinguish counterfactual instructions. The proof is given in Appendix A. CIG addresses this gap by making i^+ identifiable within $C(h_t)$.

4.2 Counterfactual Instruction Grounding

Eq. (5) defines grounding as a ranking condition over candidate instructions. We make this criterion differentiable by converting trajectory scores into a contrastive posterior over $C(h_t)$ and supervising it with the commanded instruction, yielding the Counterfactual Instruction Grounding (CIG) objective; Figure 2 illustrates the overall framework.

Contrastive posterior. Given the trajectory score $s_\theta(h_t, \tau_t, i)$, we define a posterior over the candidate instruction set:

$$q_\theta(i \mid h_t, \tau_t, C(h_t)) = \frac{\exp(s_\theta(h_t, \tau_t, i)/\kappa)}{\sum_{i' \in C(h_t)} \exp(s_\theta(h_t, \tau_t, i')/\kappa)}, \tag{8}$$

where $\kappa > 0$ is a temperature. CIG minimizes the negative log-probability of the commanded instruction:

$$\mathcal{L}_{\mathrm{CIG}} = \mathbb{E}\left[-\log q_\theta(i^+ \mid h_t, \tau_t, C(h_t))\right]. \tag{9}$$

The full training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{IL}} + \lambda \mathcal{L}_{\mathrm{CIG}}, \tag{10}$$

where λ controls the strength of the grounding term.

Theorem 1 (CIG as a contrastive information lower bound)**.** *Let $\mathcal{Q}$ be the contrastive training distribution induced by first sampling (h_t, i^+, τ_t) from the demonstration data and then sampling K negative instructions from $\rho(\cdot \mid h_t, i^+)$ to form $C(h_t) = \{i^+, i_1^-, \ldots, i_K^-\}$. Let $I = i^+$ be the demonstrated instruction and assume K distinct negatives, so $|C(h_t)| = K + 1$. Under the candidate-prior condition in Appendix A.2, the posterior q_θ defined in Eq.* (8) *satisfies*

$$\mathrm{I}_{\mathcal{Q}}(I; \tau_t \mid h_t, C(h_t)) \geq \log(K+1) - \mathcal{L}_{\mathrm{CIG}}. \tag{11}$$

Remark. CIG treats negative instructions as alternative hypotheses for the same demonstrated action chunk. Minimizing the contrastive loss encourages the commanded instruction to be distinguished from these alternatives.

Counterfactual instruction sampling. For each training chunk, $C(h_t)$ contains the commanded instruction i^+ and K negatives sampled from an instruction pool. We use category-aware sampling to include both same-category hard alternatives and cross-category alternatives, so that CIG learns both fine-grained referent discrimination and coarse task-type separation. The negatives are sampled from the instruction pool of the corresponding dataset and are not paired with counterfactual expert trajectories. They serve only as alternative instruction hypotheses for the same demonstrated chunk. The full sampling procedure is described in Appendix B.

4.3 Trajectory Scoring

A unified score interface. The contrastive posterior in Eq. (8) only requires a scalar trajectory score $s_\theta(h_t, \tau_t, i)$ for comparing candidate instructions on the same action chunk. We instantiate this interface for CR and FM policies below. The following propositions establish that these scores yield the instruction posterior in Eq. (8) through an auxiliary Laplace model for CR and an energy-based formulation for FM.

Continuous-regression energy score. For CR policies, we score the same demonstrated chunk under each candidate instruction using the negative ℓ_1 distance between predicted and demonstrated actions:

$$s_\theta^{\mathrm{CR}}(h_t, \tau_t, i) = -\left\|\hat{\tau}_\theta(h_t, i) - \tau_t\right\|_1. \tag{12}$$

Proposition 2 (Instruction posterior from CR scores)**.** *The CR score induces an auxiliary Laplace likelihood for the demonstrated action chunk:*

$$p_{\mathrm{aux},\theta}(\tau_t \mid h_t, i) \propto \exp\big(s_\theta^{\mathrm{CR}}(h_t, \tau_t, i)/\kappa\big). \tag{13}$$

With equal prior weight assigned to candidate instructions, Bayes' rule gives the instruction posterior in Eq. (8)*.*

Remark. Instructions whose predicted actions better match the demonstrated chunk receive higher posterior probability. This turns action-regression error into a signal for instruction grounding. The model specification and proof are given in Appendix A.3.

Flow-matching energy score. For FM policies, we score each instruction by how closely its predicted velocity matches the target velocity:

$$s_\theta^{\mathrm{FM}}(h_t, \tau_t, i) = -\mathbb{E}_{s,\epsilon}\left[\frac{1}{2\sigma^2}\left\|v_\theta(x_s, s, h_t, i) - u^\star(x_s, s, \tau_t)\right\|_2^2\right]. \tag{14}$$

Here $\sigma^2 > 0$ is a shared residual variance across candidate instructions. The score scales the velocity-regression error used by the FM imitation objective.

Proposition 3 (Instruction posterior from FM scores)**.** *The FM score induces an instruction-conditioned energy potential:*

$$\tilde{p}_{\theta,\kappa}(\tau_t \mid h_t, i) = \exp\big(s_\theta^{\mathrm{FM}}(h_t, \tau_t, i)/\kappa\big). \tag{15}$$

Normalizing this potential over $C(h_t)$ yields the energy-based instruction posterior in Eq. (8)*.*

Remark. Instructions with smaller expected velocity errors receive higher posterior probability. Both policy families therefore use their regression errors to compare candidate instructions for the same demonstrated chunk. The construction and proof are given in Appendix A.4.

4.4 Optimization

Let $\theta = (\theta_{\mathrm{VLM}}, \theta_{\mathrm{Exp}})$ denote the VLM backbone and action-expert parameters. We optimize the combined objective in Eq. (10) by controlling where and when the CIG gradient is applied.

VLM-only routing. CIG is designed to restore instruction-sensitive representations, while the action expert should remain primarily governed by imitation learning. We therefore allow the CIG gradient to update only the VLM backbone and prevent it from directly updating the action expert. This confines the grounding correction to the representation side, where instruction-conditioned collapse is observed, rather than forcing the action head to compensate for weak language features.

Saturation-gated update. CIG should correct insufficient instruction separation, not dominate late-stage action refinement. Once candidate instructions are already separated, continuing to optimize the grounding loss may over-sharpen language discrimination and interfere with action fidelity. We therefore activate the CIG gradient only when $\mathcal{L}_{\mathrm{CIG}}$ remains above a floor δ.

Combining VLM-only routing and the saturation gate, the actual update is

$$\nabla_{\theta_{\mathrm{VLM}}}\mathcal{L} = \nabla_{\theta_{\mathrm{VLM}}}\mathcal{L}_{\mathrm{IL}} + \lambda \mathbf{1}\{\mathcal{L}_{\mathrm{CIG}} \geq \delta\}\nabla_{\theta_{\mathrm{VLM}}}\mathcal{L}_{\mathrm{CIG}}, \qquad \nabla_{\theta_{\mathrm{Exp}}}\mathcal{L} = \nabla_{\theta_{\mathrm{Exp}}}\mathcal{L}_{\mathrm{IL}}. \tag{16}$$

When $\mathcal{L}_{\mathrm{CIG}} < \delta$, training reduces to imitation refinement; when $\mathcal{L}_{\mathrm{CIG}} \geq \delta$, CIG additionally reshapes the VLM representation. Full pseudocode is provided in Appendix C.

5 Experiments

5.1 Setup

Implementation. We conduct RoboCasa continuation training on 4 NVIDIA A100 GPUs (80GB). We report task success rate (SR), use OpenVLA-OFT and $\pi_{0.5}$ as the CR and FM baselines, respectively, and apply CIG without changing the architecture or collecting additional demonstrations. Full implementation details are provided in Appendix C.

Simulation benchmarks. We evaluate on RoboCasa [19] and LIBERO-Plus [9]. RoboCasa tests long-horizon kitchen manipulation under task ambiguity, while LIBERO-Plus tests robustness under controlled distribution shifts. Together, they evaluate whether CIG preserves instruction grounding when visually plausible alternatives or shifted conditions are present.

Real-world experiments. We conduct real-world experiments on a Hello Robot Stretch 3 using VR-collected demonstrations in kitchen scenes. The tasks cover common household manipulation scenarios, including operating articulated fixtures, picking and placing objects, and retrieving items from appliances, where multiple plausible actions may be available in the same scene.

5.2 Simulation Experiments

RoboCasa benchmark. Table 1 reports SR on RoboCasa. CIG improves $\pi_{0.5}$ from 65.8% to 71.3% average SR, with gains across all five task categories. The largest improvements appear in Insertion ($+9.0$ pp) and Pick-and-Place ($+8.3$ pp), where multiple visually plausible targets or receptacles coexist and language is needed for disambiguation. CIG also improves OpenVLA-OFT from 40.7% to 47.7% average SR, with especially strong gains on Button ($+31.3$ pp) and Open-and-Close ($+9.6$ pp). These results show that CIG consistently improves instruction-following performance across both CR and FM backbones, while the strongest gains appear in categories where trajectory scores provide clear instruction-discriminative signals.

LIBERO-Plus benchmark. Table 2 reports SR under seven controlled perturbation dimensions. CIG improves both backbones, raising OpenVLA-OFT by $+3.0$ pp and $\pi_{0.5}$ by $+1.9$ pp overall. The gains are concentrated on instruction-relevant perturbations: Robot Initial States and Language improve by $+11.7$ pp and $+3.4$ pp on OpenVLA-OFT, and by $+6.5$ pp and $+3.1$ pp on $\pi_{0.5}$. This pattern indicates that CIG improves instruction-conditioned robustness, especially when shifted states or language changes alter the salient affordance or the commanded behavior.

Table 1: **RoboCasa simulation benchmark results.** We report success rate (SR, %) on the 24-task RoboCasa benchmark, grouped into five task categories and averaged over 50 rollouts per task, with "*" denoting reported average SR from prior work where category-wise results are unavailable. Values in parentheses denote absolute SR changes in percentage points.

Method	Pick & Place	Open & Close	Knob & Lever	Button	Insertion	Avg.
GR00T-N1 [2]	50.3	81.2	59.6	74.3	46.0	62.6
Video Policy [17]	47.3	83.0	56.0	**97.3**	**64.0**	65.7
Cosmos Policy [14]	–	–	–	–	–	67.1*
OpenVLA-OFT [13]	14.8	63.7	50.4	54.0	31.0	40.7
OpenVLA-OFT + CIG	20.3 (+5.5)	73.3 (+9.6)	48.4 (-2.0)	85.3 (+31.3)	22.0 (-9.0)	47.7 (+7.0)
$\pi_{0.5}$ [21]	56.5	83.0	62.4	80.7	38.0	65.8
$\pi_{0.5}$ + CIG	**64.8** (+8.3)	**86.7** (+3.7)	**65.6** (+3.2)	84.0 (+3.3)	47.0 (+9.0)	**71.3** (+5.5)

Table 2: **LIBERO-Plus simulation benchmark results.** We report success rate (SR, %) across seven controlled perturbation dimensions over all 10,030 evaluation episodes, with one rollout per episode following [9]. Values in parentheses denote absolute SR changes in percentage points.

Method	Camera	Robot	Language	Light	Background	Noise	Layout	Avg.
WorldVLA [7]	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0
UniVLA [6]	1.8	46.2	69.6	69.0	81.0	21.2	31.9	43.9
OpenVLA-OFT [13]	91.5	30.3	83.8	97.3	93.9	93.4	75.2	79.6
OpenVLA-OFT + CIG	**92.5** (+1.0)	42.0 (+11.7)	**87.2** (+3.4)	**97.5** (+0.2)	94.0 (+0.1)	**93.5** (+0.1)	78.0 (+2.8)	82.6 (+3.0)
$\pi_{0.5}$ [21]	82.5	70.3	80.9	95.5	95.1	89.3	83.6	84.4
$\pi_{0.5}$ + CIG	82.6 (+0.1)	**76.8** (+6.5)	84.0 (+3.1)	96.4 (+0.9)	**95.4** (+0.3)	88.4 (-0.9)	**86.3** (+2.7)	**86.3** (+1.9)

5.3 Ablation Study

Table 3 adds CIG components incrementally on both $\pi_{0.5}$ and OpenVLA-OFT backbones. Adding the CIG loss improves SR on both benchmarks, with substantially larger gains on RoboCasa than on LIBERO-Plus ($+2.4$ vs $+0.9$ pp on $\pi_{0.5}$; $+4.2$ vs $+2.1$ pp on OpenVLA-OFT), indicating that counterfactual grounding mainly helps where instruction ambiguity dominates. Restricting the CIG gradient to the VLM representation improves over direct CIG updates, suggesting that grounding is most effective when applied to the instruction-conditioned representation rather than the action expert. Adding the saturation gate yields the best performance on both benchmarks, showing that CIG should correct insufficient instruction separation without continuously dominating action refinement.

Table 3: Ablation on $\pi_{0.5}$ and OpenVLA-OFT. We report success rate (SR, %) as CIG components are added incrementally.

Variant	RoboCasa	LIBERO-Plus
$\pi_{0.5}$	65.8	84.4
+ CIG loss	68.2	85.3
+ VLM routing	69.1	85.7
+ Saturation gate	**71.3**	**86.3**
OpenVLA-OFT	40.7	79.6
+ CIG loss	44.9	81.7
+ VLM routing	46.2	82.2
+ Saturation gate	**47.7**	**82.6**

5.4 Training Dynamics

Figure 3 analyzes CIG during continuation training on the $\pi_{0.5}$ backbone. Positive and counterfactual FM energies separate early, showing that CIG quickly increases the margin between the commanded instruction and its alternatives. Once $\mathcal{L}_{\text{CIG}}$ falls below the floor δ, the active rate drops as the saturation gate suppresses direct CIG updates; nonetheless $\mathcal{L}_{\text{CIG}}$ and SR keep improving, indicating that IL preserves the early CIG-induced separation while refining actions.

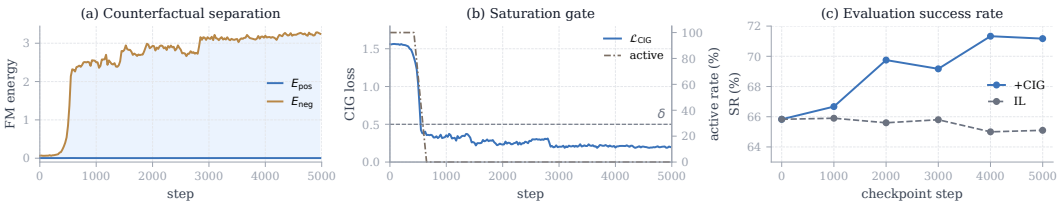


Figure 3: **Training dynamics of CIG on the $\pi_{0.5}$ backbone.** (a) Positive and counterfactual FM energies separate early. (b) $\mathcal{L}_{\text{CIG}}$ falls below the floor δ as the active rate decays. (c) Evaluation SR continues to improve after CIG activity decreases.

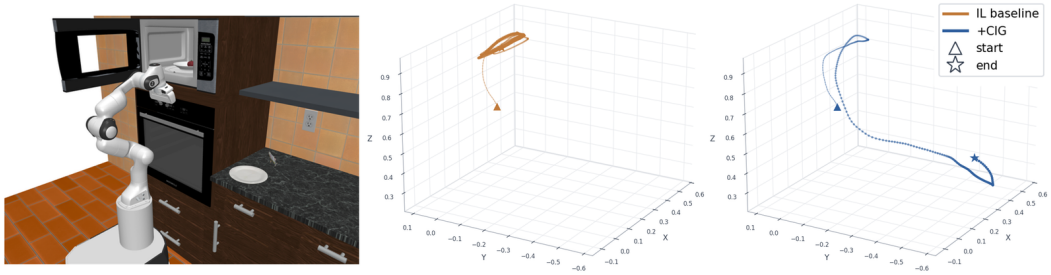


Figure 4: **Qualitative end-effector trajectories on RoboCasa PnPMicrowaveToCounter.** CIG produces a more directed trajectory than continued IL under a similar scene configuration. Triangles and stars denote start and end waypoints, respectively.

5.5 Qualitative Trajectory Analysis

Figure 4 provides a behavior-level view of the improvement behind the aggregate SR gains. Under a similar RoboCasa scene, continued IL hesitates near the interaction region, alternating between approaching and moving away, consistent with ambiguity among visually plausible actions. In contrast, CIG yields a directed reach-and-place trajectory toward task completion, suggesting that counterfactual grounding helps the policy commit to the positive instruction during execution.

5.6 Direct Grounding Diagnostic

We assess instruction grounding on RoboCasa rollout chunks using the same frozen pre-CIG IL checkpoint as a judge for both IL and CIG policies. The evaluated policies generate the trajectories, while the judge retains its pre-CIG parameters throughout evaluation. Instruction Identification Accuracy (IIA) measures whether the commanded instruction receives the highest judge score among sampled alternatives, while Conf. denotes its posterior probability under the judge. Table 4 shows improvements across all task categories, raising overall IIA from 85.8% to 91.5% and confidence from 0.409 to 0.514. With the evaluator held fixed, these gains reflect changes in the generated trajectories and support improved instruction grounding. Full protocol details are provided in Appendix E.2.

Table 4: Grounding diagnostic on RoboCasa rollouts using a shared frozen IL-baseline judge.

Category	IIA (%) ↑			Conf. ↑		
	IL	CIG	Δ	IL	CIG	Δ
Pick & Place	73.3	84.2	+10.9	0.389	0.445	+0.056
Open & Close	98.9	99.2	+0.3	0.498	0.587	+0.089
Knob & Lever	89.3	93.9	+4.6	0.437	0.681	+0.244
Button	88.9	92.4	+3.5	0.327	0.395	+0.068
Insertion	83.3	90.1	+6.8	0.272	0.332	+0.060
Overall	85.8	**91.5**	+5.7	0.409	**0.514**	+0.105

5.7 Real-World Deployment

We deploy CIG on a Hello Robot Stretch 3 in a real kitchen scene. As shown in Fig. 5, we evaluate counter-to-rack placement, washer-to-basin retrieval, and faucet operation, which require selecting

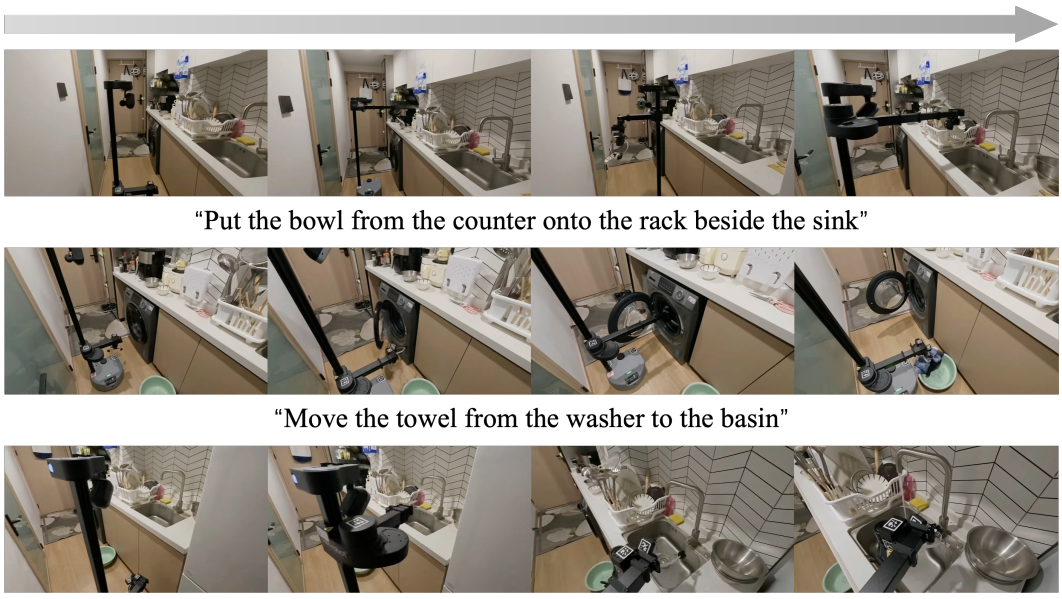


Figure 5: Real-world deployment on Hello Robot Stretch 3.

among tabletop, appliance, and articulated-fixture affordances in the same environment. We report held-out rollout SR and provide the full protocol in Appendix F.

6 Conclusion and Limitations

We identify instruction-conditioned feature collapse in fine-tuned VLA models and recast grounding as identifying the commanded instruction among counterfactual alternatives. CIG improves instruction sensitivity without architecture changes or additional demonstrations, yielding a $+5.5$ pp gain on RoboCasa, 86.3% SR on LIBERO-Plus, and transfer to real-world kitchen manipulation.

CIG is most effective when instruction disambiguation determines task success, with limited benefits when perception or low-level control is the dominant bottleneck. Within this scope, the grounding signal depends on the availability and diversity of alternative instructions in the dataset, while the FM energy supports their relative ranking without providing a calibrated trajectory likelihood. Our real-world evaluation remains limited to one robot and three tasks, with 20 trials per task and method. Future work will explore richer instruction supervision together with complementary perception and control capabilities to extend CIG to a broader range of tasks. We will also investigate calibrated trajectory scoring and evaluate these extensions across more diverse robots and environments.

Acknowledgments and Disclosure of Funding

This research was funded by the Zhongguancun Academy (Grant No. 20240306) and the National Natural Science Foundation of China (Grant Nos. 62376266 and 62406318).

References

[1] Abrar Anwar, Rohan Gupta, and Jesse Thomason. Contrast sets for evaluating language-guided robot policies. *arXiv preprint arXiv:2406.13636*, 2024.

[2] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[3] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems (RSS)*, 2025.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[6] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions. In *Robotics: Science and Systems (RSS)*, 2025.

[7] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. WorldVLA: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[8] 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.

[9] 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.

[10] 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.

[11] Bryce Grant, Xijia Zhao, and Peng Wang. Not all features are created equal: A mechanistic study of vision-language-action models. *arXiv preprint arXiv:2603.19233*, 2026.

[12] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning (CoRL)*, 2024.

[13] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems (RSS)*, 2025.

[14] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

[15] Xinghang Li, Peiyan Li, Long Qian, Minghuan Liu, et al. What matters in building vision-language-action models for generalist robots. *arXiv preprint arXiv:2412.14058*, 2024.

[16] 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. *arXiv preprint arXiv:2601.15197*, 2026.

[17] Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. Video generators are robot policies. *arXiv preprint arXiv:2508.00795*, 2025.

[18] Xiaopeng Lin, Shijie Lian, Bin Yu, Ruoqi Yang, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Yurun Jin, Yukun Shi, Jiyan He, Cong Huang, Bojun Cheng, and Kai Chen. PhysBrain: Human egocentric data as a bridge from vision language models to physical intelligence. *arXiv preprint arXiv:2512.16793*, 2025.

[19] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

[20] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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

[22] Aiden Swann, Lachlain McGranahan, Hugo Buurmeijer, Monroe Kennedy III, and Mac Schwager. Sparse autoencoders reveal interpretable and steerable features in vla models. *arXiv preprint arXiv:2603.19183*, 2026.

[23] Yanwei Wang, Tsun-Hsuan Wang, Jiayuan Mao, Michael Hagenow, and Julie Shah. Grounding language plans in demonstrations through counterfactual perturbations. In *International Conference on Learning Representations (ICLR)*, 2024.

[24] Lei Xiao, Jifeng Li, Juntao Gao, Feiyang Ye, Yan Jin, Jingjing Qian, Jing Zhang, Yong Wu, and Xiaoyuan Yu. AVA-VLA: Improving vision-language-action models with active visual attention. *arXiv preprint arXiv:2511.18960*, 2025.

[25] Siyu Xu, Zijian Wang, Yunke Wang, Chenghao Xia, Tao Huang, and Chang Xu. Affordance field intervention: Enabling VLAs to escape memory traps in robotic manipulation. *arXiv preprint arXiv:2512.07472*, 2025.

[26] Wanpeng Zhang, Ye Wang, Hao Luo, Haoqi Yuan, Yicheng Feng, Sipeng Zheng, Qin Jin, and Zongqing Lu. DiG-Flow: Discrepancy-guided flow matching for robust VLA models. *arXiv preprint arXiv:2512.01715*, 2025.

Counterfactual Instruction Grounding for Vision-Language-Action Models

Appendix

A	Proofs	14
B	Counterfactual Set Construction	15
C	Implementation Details	16
D	Sensitivity to CIG Hyperparameters	18
E	Additional Results and Analysis	18
	E.1 Full RoboCasa Results	18
	E.2 Direct Grounding Diagnostic	18
	E.3 Additional Representation Analysis	19
	E.4 RoboCasa Qualitative Results	20
	E.5 LIBERO-Plus Qualitative Results	20
F	Stretch 3 Real-World Protocol	23

A Proofs

A.1 Proof of Proposition 1

Assumptions. The CR targets have finite first moments and the FM targets have finite second moments. For FM, noise and interpolation time are sampled independently of (h_t, i, τ_t), with a common interpolation rule depending only on the chunk, noise and time. The model classes contain the corresponding Bayes regression predictors.

For CR, the population L1 risk separates over action coordinates. A conditional median minimizes the expected absolute error in each coordinate. By the assumed sufficiency of $g(h_t)$, each conditional target distribution depends only on $g(h_t)$, so an instruction-invariant choice of these medians is Bayes-optimal:

$$\hat{\tau}^*(h_t, i) = m(g(h_t)).$$

For FM, squared velocity error is minimized by the conditional mean of the target $u^\star$:

$$v^*(x_s, s, h_t, i) = \mathbb{E}[u^\star \mid x_s, s, h_t, i].$$

The demonstrated chunk distribution depends only on $g(h_t)$, and the noise and time are sampled independently of (h_t, i, τ_t) under a common interpolation rule. The joint conditional distribution of $(x_s, u^\star)$ therefore depends on (h_t, i) only through $g(h_t)$ (and s). Consequently,

$$v^*(x_s, s, h_t, i) = \mathbb{E}[u^\star \mid x_s, s, g(h_t)] = \bar{u}(x_s, s, g(h_t)).$$

The stated moment and model-capacity assumptions ensure these Bayes predictors are admissible. Both imitation objectives thus admit an optimum that ignores the instruction. □

A.2 Proof of Theorem 1

Write $Z = (h_t, C(h_t))$ and let $I = i^+$ under the contrastive training distribution $\mathcal{Q}$. The candidate-prior condition for Theorem 1 is

$$p_{\mathcal{Q}}(I = i \mid Z) = \frac{1}{K+1}, \qquad i \in C(h_t).$$

This condition concerns the data-induced candidate prior before observing the action chunk. With $p = p_{\mathcal{Q}}$, the non-negativity of conditional KL divergence gives

$$\mathbb{E}_{\mathcal{Q}}[\log p(I \mid Z, \tau_t)] \geq \mathbb{E}_{\mathcal{Q}}[\log q_\theta(I \mid Z, \tau_t)] = -\mathcal{L}_{\mathrm{CIG}}.$$

Consequently,

$$\begin{aligned} \mathrm{I}_{\mathcal{Q}}(I; \tau_t \mid Z) &= \mathbb{E}_{\mathcal{Q}}\left[\log \frac{p(I \mid Z, \tau_t)}{p(I \mid Z)}\right] \\ &= \log(K+1) + \mathbb{E}_{\mathcal{Q}}[\log p(I \mid Z, \tau_t)] \\ &\geq \log(K+1) - \mathcal{L}_{\mathrm{CIG}}. \end{aligned}$$

This proves Eq. (11). For a non-uniform candidate prior, the general lower bound additionally subtracts $\mathbb{E}_Z[\mathrm{KL}(p(\cdot \mid Z) \| U_{C(h_t)})]$, where $U_{C(h_t)}$ is uniform on the candidate set. The category-aware sampler is not assumed to satisfy the candidate-prior condition. □

A.3 Proof of Proposition 2

Fix $\kappa > 0$. Let N be the number of scalar entries in an action chunk, flatten these entries into coordinates $j = 1, \ldots, N$, and set $b = \kappa$. Assume independent Laplace residuals with this shared scale and a uniform prior over the candidate instructions. The auxiliary Laplace observation density is

$$p_{\mathrm{aux},\theta}(\tau_t \mid h_t, i) = (2b)^{-N} \exp\left(-\frac{1}{b} \sum_{j=1}^{N} |\tau_{t,j} - \hat{\tau}_{\theta,j}(h_t, i)|\right).$$

Using Eq. (12),

$$\log p_{\mathrm{aux},\theta}(\tau_t \mid h_t, i) = \frac{s_\theta^{\mathrm{CR}}(h_t, \tau_t, i)}{\kappa} - N \log(2\kappa).$$

For fixed chunk dimensions and temperature, the second term is independent of the candidate instruction. With a uniform prior over $C(h_t)$, Bayes' rule gives

$$p_{\mathrm{aux},\theta}(i \mid h_t, \tau_t, C(h_t)) = \frac{\exp(s_\theta^{\mathrm{CR}}(h_t, \tau_t, i)/\kappa)}{\sum_{i' \in C(h_t)} \exp(s_\theta^{\mathrm{CR}}(h_t, \tau_t, i')/\kappa)},$$

which is Eq. (8). This is a posterior under the stated auxiliary observation model, not a calibrated rollout-density claim. □

A.4 Proof of Proposition 3

Consider an isotropic Gaussian residual model with variance $\sigma^2 > 0$ shared across candidate instructions. Its negative log-likelihood is $\frac{1}{2\sigma^2}\|v - u\|_2^2$ up to an additive constant independent of the instruction. Let s_θ^{FM} be its negative expectation as in Eq. (14). For any fixed $\kappa > 0$, define the tempered, unnormalized potential

$$\tilde{p}_{\theta,\kappa}(\tau_t \mid h_t, i) = \exp\big(s_\theta^{\mathrm{FM}}(h_t, \tau_t, i)/\kappa\big)\,.$$

With a uniform prior over the finite candidate set, normalizing this potential over instructions gives

$$q_\theta(i \mid h_t, \tau_t, C(h_t)) = \frac{\tilde{p}_{\theta,\kappa}(\tau_t \mid h_t, i)}{\sum_{i' \in C(h_t)} \tilde{p}_{\theta,\kappa}(\tau_t \mid h_t, i')},$$

which is Eq. (8). An additive energy constant independent of i cancels in this ratio. This construction normalizes over candidate instructions only; it makes no claim about a normalized density over action chunks. □

B Counterfactual Set Construction

For each training chunk, CIG constructs a candidate instruction set $C(h_t) = \{i^+, i_1^-, \ldots, i_K^-\}$ containing the commanded instruction and K negative instructions. The negative instructions are not paired with counterfactual expert trajectories; they are used only as alternative instruction hypotheses for the same observed demonstration chunk. Across all experiments, the instruction pool is built from the task instructions already available in the corresponding benchmark or real-robot dataset. Thus, CIG does not require additional demonstrations, additional environment interaction, or counterfactual trajectory collection.

RoboCasa. For RoboCasa, we collect a pool of 4183 natural-language task instructions from the existing RoboCasa task descriptions and training instructions. We assign each instruction to one of seven coarse task categories: pick-and-place (4113 instructions, covering the surface-to-surface pick-and-place variants), door/drawer open-close (28), stove on/off (21), sink on/off (6), sink spout (3), microwave on/off (6), and coffee operations (6). The pool is naturally imbalanced because RoboCasa contains many more pick-and-place variants than other task types. To avoid simply sampling negatives according to this global imbalance, we use category-aware sampling: for each positive instruction, 50% of negatives are drawn from the same category, 25% from different categories, and 25% uniformly at random. Same-category negatives provide hard referent or target contrasts, while cross-category negatives preserve coarse action-type discrimination. Category-aware sampling prevents the negative distribution from being dominated purely by pick-and-place frequency. For $K = 4$, the sampler selects two same-category instructions, one cross-category instruction, and one random instruction, excluding the positive and previously selected instructions. For a non-pick-and-place positive, the cross-category pool excludes the dominant pick-and-place category; the final random draw may come from any category. If a category has too few candidates or category metadata is unavailable, the sampler fills the remaining slots uniformly from the unused pool. The released 4,183-instruction pool supports sampling all four negatives without replacement.

LIBERO-Plus. For LIBERO-Plus, the negative pool is constructed from the instructions in the LIBERO-Plus training task set only; evaluation-set instructions are never seen during training. We use the same candidate-set construction rule as RoboCasa whenever category labels are available. When a benchmark task does not provide a natural coarse category, we use task templates and object/receptacle/action keywords to assign coarse categories for sampling. These categories are used only for negative sampling and are never provided to the model.

Algorithm 1 Counterfactual Instruction Grounding (CIG)

Require: Imitation-learned VLA policy π_θ with parameters $\theta = (\theta_{\mathrm{VLM}}, \theta_{\mathrm{Exp}})$; demonstration dataset $\mathcal{D}$; instruction pool $\mathcal{P}$; number of negatives K; temperature κ; grounding weight λ; saturation floor δ.
Ensure: Updated policy parameters θ.

for each training step **do**
  Sample a training tuple (h_t, i^+, τ_t) from $\mathcal{D}$.
  Construct $C(h_t) = \{i^+, i_1^-, \ldots, i_K^-\}$ by sampling negatives from $\mathcal{P}$.
  Compute the imitation loss $\mathcal{L}_{\mathrm{IL}}$ on (h_t, i^+, τ_t) and backpropagate through all trainable modules.
  for each $i \in C(h_t)$ **do**
    if the policy is a CR policy **then**
      Compute s_θ^{CR} as negative mean L1 action error using Eq. (12).
    else if the policy is an FM policy **then**
      Compute s_θ^{FM} as negative expected squared velocity error using Eq. (14).
    end if
  end for
  Compute $q_\theta(i \mid h_t, \tau_t, C(h_t))$ using Eq. (8).
  Average $-\log q_\theta(i^+ \mid h_t, \tau_t, C(h_t))$ over the local microbatch to obtain $\mathcal{L}_{\mathrm{CIG}}$.
  if $\mathcal{L}_{\mathrm{CIG}} \geq \delta$ **then**
    Backpropagate $\lambda \mathcal{L}_{\mathrm{CIG}}$ while updating only θ_{VLM}.
  else
    Skip the CIG gradient.
  end if
  Average accumulated gradients across GPUs and take one optimizer step.
end for

Stretch 3. For the real-robot Stretch 3 experiments, the instruction pool is built from the three real-world task instructions and their paraphrases: counter-to-rack placement, washer-to-basin retrieval, and faucet operation. For each positive instruction, negatives are sampled from the other task types and from paraphrases of competing tasks in the same kitchen scene. This creates candidate sets in which all alternatives are physically plausible in the shared environment but correspond to different affordances.

Role of category labels. The coarse category labels are used only to balance the negative sampling distribution. They are not used as model inputs, action labels, reward signals, or additional supervision for the policy. Therefore, CIG does not change the demonstration data or require counterfactual trajectories; it only changes how the existing instruction annotations are organized into contrastive candidate sets.

Alternative designs considered. We also tested two alternatives which performed worse. *Uniform random:* drop category-aware sampling entirely; this loses about 0.5 pp peak compared to the category-aware default. *Per-scene scene-conditional filtering:* compute a runtime scene-applicability filter (as in the early version of this work); in practice, the real instruction pool at RoboCasa scale is already dominated by in-distribution alternatives, so scene filtering never excluded more than $\sim 2\%$ of the pool and did not move the final number, while adding considerable engineering complexity. We therefore dropped it in favor of the flat pool.

C Implementation Details

C.1 CIG Training Step

Algorithm 1 summarizes one continuation-training step with CIG. Starting from an imitation-learned VLA policy, each update accumulates the standard imitation-learning gradient and, when the saturation condition is active, the routed counterfactual grounding gradient before a single optimizer step. The counterfactual instruction set is constructed using the procedure described in Appendix B.

Table 5: Main training configuration for CIG continuation training.

Setting	OpenVLA-OFT	$\pi_{0.5}$
GPUs	4×A100	4×A100
Effective batch size	16	128
Learning rate	5×10^{-5}	5×10^{-5} after 1k warmup
CIG steps	5k	5k
Grounding weight λ	0.1	0.1
Saturation floor δ	0.5	0.5
Temperature κ	1.0	1.0
Negatives K	4	4
Negative sampling	50% same-category, 25% cross-category, 25% random	

C.2 Gradient Routing

We implement VLM-only routing with a two-pass update. The first pass computes $\mathcal{L}_{\mathrm{IL}}$ and accumulates gradients for all trainable modules, including both the VLM backbone and the action expert. The second pass recomputes trajectory scores over the candidate instruction set, but excludes the action expert parameters from the CIG optimizer update. Thus, the action expert is still used to evaluate the instruction-conditioned score, while the CIG gradient is applied only to the VLM-side representation. If $\mathcal{L}_{\mathrm{CIG}} < \delta$, the saturation gate skips the CIG gradient. For OFT, the action head and proprioceptive projector retain only their IL gradients. The gate acts on each GPU microbatch; gradients are averaged across GPUs after both passes and any gradient accumulation, then one optimizer step is taken. This keeps action imitation governed by $\mathcal{L}_{\mathrm{IL}}$ while using CIG to reshape instruction-sensitive representations.

C.3 Main Training Configuration

Table 5 summarizes the RoboCasa continuation-training configuration for the CR and FM backbones. The RoboCasa settings use four GPUs, with global batch sizes shown below. The two backbones share the grounding weight, temperature, saturation floor, negative count and sampling proportions.

For RoboCasa, OFT uses a per-GPU batch size of 4 with no gradient accumulation. Its 5k continuation updates start from the imitation checkpoint labelled 100k and end at update 105k. The $\pi_{0.5}$ batch size is global; its learning-rate schedule warms up for 1k updates and has equal peak and terminal rates of 5×10^{-5}. These continuation settings are distinct from initial imitation training.

Optimizer and initialization. Both continuation recipes use AdamW with $\epsilon = 10^{-8}$ and a fresh optimizer initialized from the corresponding imitation checkpoint. OFT trains rank-128 LoRA adapters plus the action head and proprioceptive projector, with $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay 0.01, no warmup and a constant learning rate over the 5k continuation updates. The $\pi_{0.5}$ recipe uses $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 10^{-10} and global gradient-norm clipping at 1.0. Its baseline is trained with the official openpi JAX loop and converted to PyTorch for continuation and inference. The separate release baseline recipes use 100k OFT updates (global batch 32, learning rate 5×10^{-4}) and 15k $\pi_{0.5}$ updates (global batch 64, 1k warmup to 5×10^{-5}).

RoboCasa data and evaluation configuration. The release uses official RoboCasa v0.1 image demonstrations for the 24 single-stage tasks. The three RGB views (left, right and wrist) are stored at 128×128 and resized to 224×224 for the policy. The raw state/action dimensions are 27/12; stationary manipulation retains the first 25 state entries and the first 7 action entries, with $\pi_{0.5}$ padding model tensors to 32 dimensions. Both simulator policies predict 10-step chunks and execute 5 steps before replanning at a 20 Hz control rate. The evaluation recipe uses object split B, layout/style pairs $(1, 1), (2, 2), (4, 4), (6, 9), (7, 10)$, seed 42, 50 trials per task and the task-specific horizons supplied with the code; camera randomization and generative textures are disabled.

Release scope. The release contains the RoboCasa training/evaluation code and four simulator checkpoints (OFT and $\pi_{0.5}$, each with and without CIG). The Stretch 3 VR tools support the teleoperation, recording and command replay used for the real-world data-collection workflow in Appendix F. They provide timestamped source recordings; conversion to the processed policy dataset

Table 6: Sensitivity to CIG hyperparameters. We vary one hyperparameter at a time and report success rate (SR, %) on RoboCasa and LIBERO-Plus. Default settings are marked in bold.

Hyperparameter	Setting	RoboCasa SR ↑	LIBERO-Plus SR ↑
λ	0.05	70.6	84.0
	0.10	**71.3**	**86.3**
	0.30	69.9	86.0
δ	0.3	69.8	85.9
	0.5	**71.3**	**86.3**
	0.7	71.0	85.8
K	2	70.5	84.7
	4	**71.3**	**86.3**
	8	71.2	86.1

and the real-robot policy deployment loop are outside the released module. The release does not include LIBERO-Plus training/evaluation, real-robot policy checkpoints, or the real-robot training dataset.

D Sensitivity to CIG Hyperparameters

We study three CIG-specific hyperparameters: the grounding weight λ, the saturation floor δ, and the number of counterfactual negatives K. The default setting used in the main experiments is $\lambda = 0.1$, $\delta = 0.5$, and $K = 4$. Table 6 varies one hyperparameter at a time while keeping the others fixed. Overall, CIG is not highly sensitive to moderate changes in these parameters, while the default setting provides the strongest average performance across RoboCasa and LIBERO-Plus.

The sweep shows a consistent but mild preference for the default configuration. For λ, a smaller value weakens the grounding signal, while a larger value can over-emphasize counterfactual discrimination and slightly reduce RoboCasa performance. For δ, the intermediate floor works best: a smaller floor keeps CIG active more aggressively, while a larger floor suppresses grounding updates earlier. For K, using four negatives gives the best trade-off between contrastive strength and training cost; increasing to eight negatives yields similar performance but adds computation.

E Additional Results and Analysis

E.1 Full RoboCasa Results

Table 7 reports per-task success rates for the 24 RoboCasa tasks for the four evaluated OFT and $\pi_{0.5}$ variants in Table 1. Tasks are grouped into the same five categories as the main text. Each number is the mean SR over 50 rollouts. Category-average rows are italicized and the overall average row is bolded.

E.2 Direct Grounding Diagnostic

We provide the protocol for the score-level grounding diagnostic reported in Table 4. Both IL and CIG rollouts are scored by the same frozen pre-CIG IL checkpoint, with parameters denoted by $\bar{\theta}$. The judge is held fixed throughout evaluation and has received only the IL objective. Each evaluated policy supplies its own rollout histories and action chunks, so the comparison measures instruction identifiability in generated behavior under a common evaluator. For each of the 24 RoboCasa tasks, we sample 5 evaluation rollouts and extract 3 non-idle chunks at different task-progress stages, yielding 360 chunks per evaluated policy. For each chunk (h_t, τ_t), we construct a candidate set $C(h_t)$ containing the commanded instruction and $K = 4$ negative instructions sampled with the same category-aware rule used for CIG training. For flow-matching scoring, we estimate the energy score with $n_{\text{noise}} = 8$ noise samples and use diagnostic temperature $\kappa_{\text{diag}} = 0.01$ when computing the posterior over candidate instructions. This score-level evaluation temperature is distinct from the continuation-training temperature $\kappa = 1.0$; it does not change the argmax used for IIA.

Table 7: Per-task success rate (%) on RoboCasa for the four evaluated OFT and $\pi_{0.5}$ variants in Table 1. Tasks are grouped into the same five categories as the main table; italicized rows are per-category averages and the final bolded row is the 24-task average.

Category	Task	$\pi_{0.5}$	$\pi_{0.5}$+CIG	OpenVLA-OFT	OpenVLA-OFT+CIG
Pick & Place	PnPCounterToMicrowave	34.0	40.0	6.0	12.0
	PnPCounterToSink	50.0	72.0	18.0	24.0
	PnPCounterToCab	64.0	66.0	20.0	24.0
	PnPCabToCounter	60.0	62.0	2.0	10.0
	PnPSinkToCounter	62.0	72.0	34.0	40.0
	PnPMicrowaveToCounter	42.0	56.0	6.0	6.0
	PnPCounterToStove	70.0	72.0	16.0	22.0
	PnPStoveToCounter	70.0	78.0	16.0	24.0
	Category avg.	*56.5*	*64.8*	*14.8*	*20.3*
Open & Close	OpenDoubleDoor	98.0	100.0	46.0	76.0
	CloseDoubleDoor	84.0	82.0	64.0	82.0
	OpenSingleDoor	60.0	72.0	30.0	50.0
	CloseSingleDoor	82.0	82.0	76.0	72.0
	OpenDrawer	76.0	86.0	68.0	64.0
	CloseDrawer	98.0	98.0	98.0	96.0
	Category avg.	*83.0*	*86.7*	*63.7*	*73.3*
Knob & Lever	TurnOnSinkFaucet	76.0	78.0	50.0	40.0
	TurnOffSinkFaucet	70.0	76.0	82.0	88.0
	TurnSinkSpout	84.0	88.0	74.0	76.0
	TurnOnStove	58.0	66.0	34.0	24.0
	TurnOffStove	24.0	20.0	12.0	14.0
	Category avg.	*62.4*	*65.6*	*50.4*	*48.4*
Button	TurnOnMicrowave	78.0	84.0	36.0	76.0
	TurnOffMicrowave	94.0	92.0	82.0	94.0
	CoffeePressButton	70.0	76.0	44.0	86.0
	Category avg.	*80.7*	*84.0*	*54.0*	*85.3*
Insertion	CoffeeSetupMug	22.0	26.0	14.0	16.0
	CoffeeServeMug	54.0	68.0	48.0	28.0
	Category avg.	*38.0*	*47.0*	*31.0*	*22.0*
Overall average (24 tasks)		**65.8**	**71.3**	**40.7**	**47.7**

Instruction Identification Accuracy (IIA) measures whether the commanded instruction receives the highest trajectory score:

$$\mathrm{IIA} = \mathbb{E}\left[\mathbf{1}\left(i^{+} = \arg\max_{i \in C(h_t)} s_{\bar{\theta}}(h_t, \tau_t, i)\right)\right].$$

Confidence measures the posterior probability assigned to the commanded instruction:

$$\mathrm{Conf.} = \mathbb{E}\left[q_{\bar{\theta}}(i^{+} \mid h_t, \tau_t, C(h_t))\right].$$

E.3 Additional Representation Analysis

Figure 6 provides an additional representation-level visualization of the depth-progress effect discussed in Fig. 1(b). We compare continued IL and CIG using two counterfactual instructions from the same microwave scene, *MW open* and *MW close*. For each method, we extract instruction-conditioned VLM features from layers L11, L14, and L18 and project them with t-SNE. Marker opacity indicates task progress, with lighter points corresponding to earlier execution and darker points corresponding to later execution.

Compared with continued IL, CIG maintains a more structured separation between the two instruction conditions across depth and progress. In the baseline, the feature trajectories become less consistently instruction-dependent as execution proceeds, especially in deeper layers. With CIG, the two instruction-conditioned trajectories remain more organized and retain clearer task-specific structure. This qualitative evidence is consistent with our claim that CIG mitigates the instruction-conditioned feature collapse observed during fine-tuning.

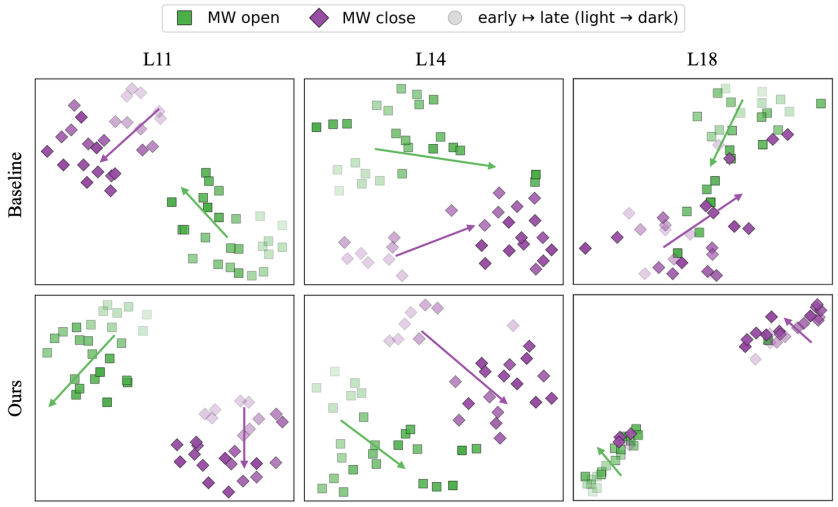


Figure 6: Additional t-SNE visualization of instruction-conditioned VLM features. We compare continued IL and CIG using two counterfactual microwave instructions, *MW open* and *MW close*, across layers L11, L14, and L18. Opacity indicates progress from early to late execution. CIG preserves more instruction-dependent structure across depth and progress than continued IL.

E.4 RoboCasa Qualitative Results

Figure 7 shows additional qualitative rollout examples on RoboCasa. We select representative tasks from different categories, including pick-and-place, fixture operation, and longer-horizon manipulation. These examples complement the aggregate SR results in Table 1 by showing how CIG changes the execution behavior in ambiguous scenes. Compared with continued IL, CIG more consistently commits to the commanded object, receptacle, or fixture instead of hesitating or drifting toward visually plausible alternatives.

E.5 LIBERO-Plus Qualitative Results

Figure 8 shows representative LIBERO-Plus rollouts under controlled perturbations. We focus on language- and layout-sensitive settings because these dimensions most directly test whether the policy preserves the commanded behavior when the scene remains visually plausible but the instruction-relevant decision changes. The qualitative examples are consistent with the quantitative trend in Table 2: CIG is most beneficial when the perturbation changes which object, target, or action should be selected, rather than only testing generic low-level robustness.

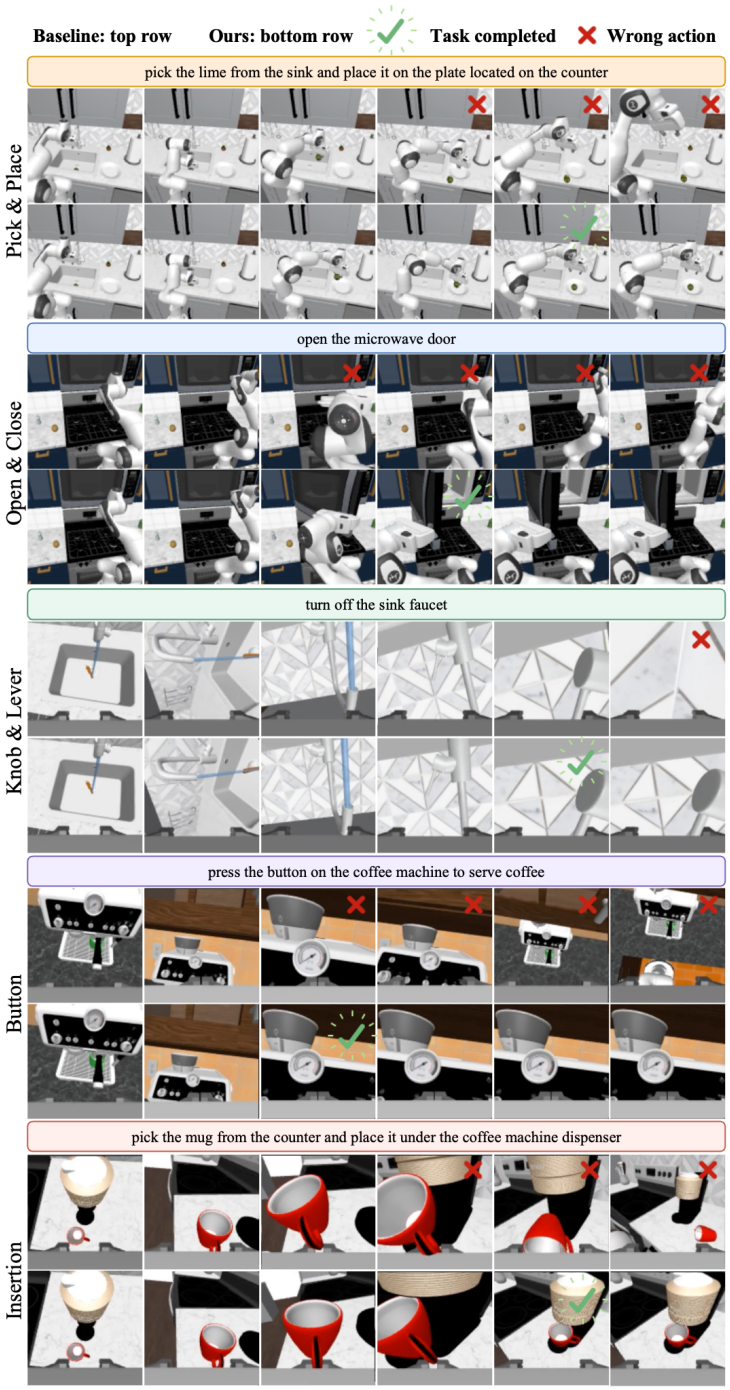


Figure 7: Additional qualitative results on RoboCasa. We show key frames from representative tasks covering pick-and-place, fixture operation, and long-horizon manipulation. Continued IL often hesitates, repeats local motions, or drifts toward visually plausible alternatives, while CIG more consistently follows the commanded object, target, or fixture.

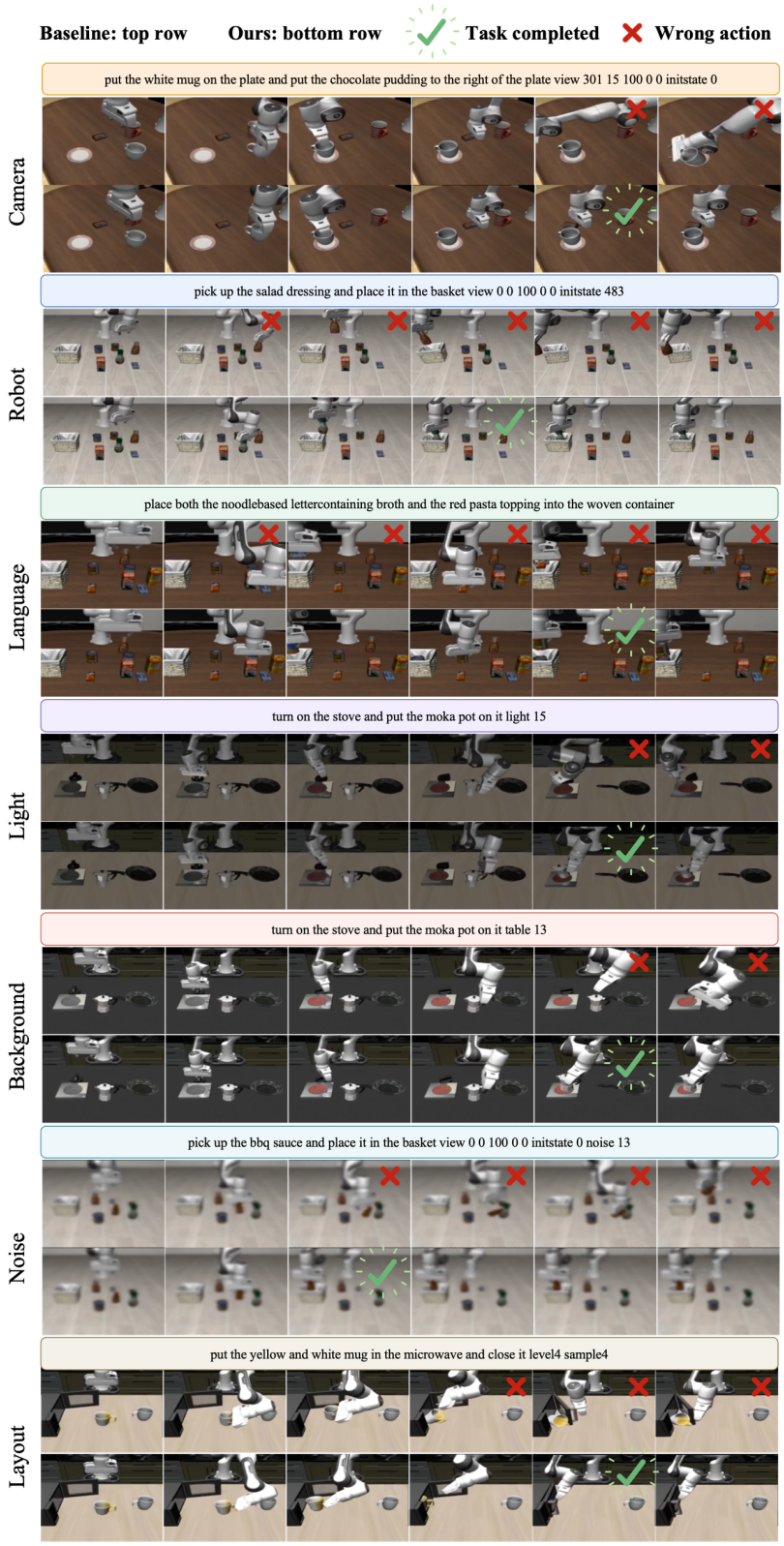


Figure 8: Additional qualitative results on LIBERO-Plus. We show representative rollouts under language, layout, and viewpoint or robot-state perturbations. CIG better preserves the commanded behavior under instruction-relevant shifts, consistent with improved performance in the Language, Layout and Robot Initial States dimensions.



Figure 9: Real-world hardware and kitchen setup. Left: Hello Robot Stretch 3 platform with head and wrist cameras. Right: kitchen scene used for real-world evaluation. Task 1 is counter-to-rack placement, Task 2 is washer-to-basin retrieval, and Task 3 is faucet operation.

F Stretch 3 Real-World Protocol

Robot and scene setup. We conduct real-world experiments on a Hello Robot Stretch 3 mobile manipulator in a kitchen environment. The robot consists of a differential-drive mobile base, a vertical lift, an extendable arm, a DexWrist3 wrist, and a parallel gripper. For perception, we use a head-mounted Intel RealSense D435if camera and a wrist-mounted Intel RealSense D405 camera. The kitchen scene contains a counter, a sink with a faucet, a dish drying rack beside the sink, and a washer under the counter. As shown in Fig. 9, the three tasks share the same scene but require selecting different affordances according to the instruction.

Data collection. We collect 50 VR-teleoperated demonstrations for each of the three kitchen tasks, with an average episode duration of approximately 2 minutes. Demonstrations are collected using a Quest 3S VR headset and a WebXR teleoperation interface, through which the operator controls the mobile base, lift, arm extension, wrist pose, and gripper state while viewing the robot cameras. The VR system records timestamped RGB observations from the head and wrist cameras, proprioceptive states, gripper states, and low-level robot actions. For the reported experiments, these streams are temporally aligned to a common 15 Hz demonstration timeline. Incomplete, unsafe, or clearly failed teleoperation episodes are discarded.

Preprocessing. VR-recorded RGB frames from both cameras are center-cropped and resized to 224×224. Robot states and actions are temporally aligned with the camera frames at 15 Hz. The processed policy dataset uses the 14D state and 18D action representation in Table 8; videos are encoded with H.264 at CRF 15. We remove episodes with severe occlusion, missing sensor streams, or safety-stop events. We also trim initial and terminal idle segments where the robot remains stationary for multiple consecutive steps. Only RGB observations are used by the policy.

Deployment. Table 8 summarizes the real-robot deployment settings. Inference is performed on a single RTX 5090 workstation, and control is executed at 15 Hz, matching the aligned VR demonstration frequency. The $\pi_{0.5}$ policy predicts action chunks of length 12 and replans every 5 executed steps. This corresponds to a 0.8 s action horizon and a 0.33 s replanning period. To compensate for inference and communication latency, we skip the first 3 predicted actions and execute the remaining actions with overlap across consecutive chunks. The policy uses a 14D proprioceptive state and outputs an 18D robot action command. The base pose component (x, y, θ) is normalized in the episode frame, with each episode initialized at $(0, 0, 0)$.

Table 8: VR demonstration preprocessing and policy deployment settings on Stretch 3.

Item	Setting
Collection interface	Quest 3S VR / WebXR
Aligned VR data / deployment rate	15 Hz
Deployment GPU	RTX 5090
Action horizon	12 steps
Replanning interval	5 steps
Deployment skip	3 steps
Chunk duration	0.8 s
Replanning period	0.33 s
State / action dimension	14D / 18D
Head RGB image	$224 \times 224 \times 3$
Wrist RGB image	$224 \times 224 \times 3$
Image encoding	H.264, CRF 15

Task definitions and success criteria. We design three real-world kitchen tasks to evaluate instruction-conditioned behavior in a shared scene where multiple physically plausible actions are visible.

1. **Counter-to-rack placement.** The robot is instructed to pick up a bowl from the kitchen counter and place it onto the dish drying rack beside the sink. A trial is successful if the bowl is grasped from the counter and stably placed on the drying rack. A trial fails if the robot grasps the wrong object, places the bowl outside the drying rack, drops the bowl, or fails to reach the target region.
2. **Washer-to-basin retrieval.** The robot is instructed to move a towel from the washer to a basin on the floor. A trial is successful if the towel is removed from the washer and stably placed in the basin. A trial fails if the robot interacts with the wrong source, fails to retrieve the towel, places it outside the basin, drops it, or exceeds the time limit.
3. **Faucet operation.** The robot is instructed to turn on the kitchen faucet. A trial is successful if the faucet is in the on state at the end of the rollout. A trial fails if the robot reaches toward an unrelated object, interacts with the wrong fixture, fails to actuate the faucet, or triggers a safety stop.

Instructions. For the main evaluation, we use one canonical instruction for each task:

- "Put the bowl from the counter onto the rack beside the sink."
- "Move the towel from the washer to the basin."
- "Turn on the faucet."

These instructions are chosen so that the correct behavior depends on selecting the appropriate source, target, or fixture within the same kitchen scene.

Trial protocol and randomization. For each task, we evaluate both the continued-IL baseline and CIG on 20 held-out rollouts. Before each rollout, we randomize object poses within a bounded reachable region while preserving task feasibility. For counter-to-rack placement, the bowl is placed at different positions on the counter, while the drying rack remains beside the sink. For washer-to-basin retrieval, the towel is placed at different reachable positions inside the washer, and the basin is slightly perturbed on the floor. For faucet operation, the faucet starts from the same initial state, while nearby countertop objects remain visible as distractors. Each rollout is a single attempt with a fixed time horizon and no human intervention after execution starts. Outcomes are recorded as binary success or failure according to the criteria above.

Results and visualization. Table 9 reports per-task success and failure counts together with the overall success rate. CIG raises overall SR from 23.3% (IL) to 56.7%, with consistent gains on all three tasks: counter-to-rack ($20\% \rightarrow 60\%$), washer-to-basin ($10\% \rightarrow 35\%$), and faucet ($40\% \rightarrow 75\%$). Figure 10 visualizes representative real-world rollouts for the three kitchen tasks. Each task is shown with synchronized head and wrist camera views, where rows progress from

Table 9: Real-world results on Hello Robot Stretch 3. We report success/failure counts and success rate (SR) over 20 held-out rollouts per task.

Task	IL			CIG		
	Success	Failure	SR (%)	Success	Failure	SR (%)
Counter-to-rack placement	4	16	20.0	12	8	60.0
Washer-to-basin retrieval	2	18	10.0	7	13	35.0
Faucet operation	8	12	40.0	15	5	75.0
Total	14	46	23.3	34	26	56.7

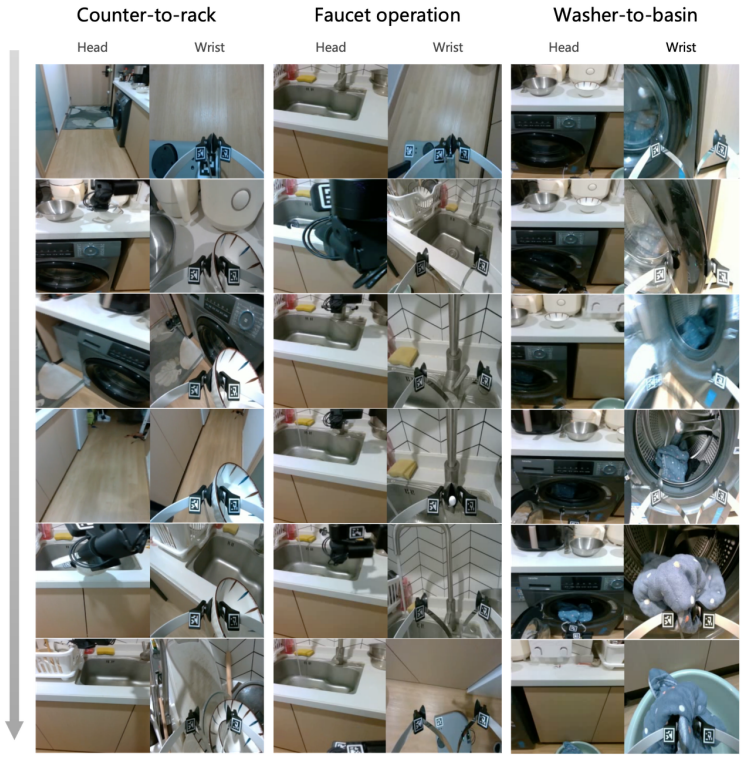


Figure 10: **Real-world kitchen rollouts.** We visualize three real-world tasks with synchronized head and wrist camera views. Columns correspond to tasks and camera views, while rows show task progress from top to bottom.

top to bottom. The three columns correspond to counter-to-rack placement, faucet operation, and washer-to-basin retrieval, respectively.

The visualization highlights that the evaluation requires more than executing a visually plausible local action. In the counter-to-rack task, the policy must pick the object from the counter and move it toward the dish rack rather than interacting with nearby appliances or countertop objects. In the faucet task, the policy must select and actuate the correct articulated fixture in a cluttered sink region. In the washer-to-basin task, the policy must retrieve the cloth from the washer and place it into the basin, requiring both correct source selection and stable transport. Across these examples, successful behavior depends on preserving the commanded instruction across perception, approach, interaction, and placement stages.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state both the empirical claims (average gains on RoboCasa and LIBERO-Plus, preliminary real-world results) and the theoretical scope (instruction-invariant IL optima under sufficiency, and a contrastive mutual-information lower bound under the contrastive distribution $\mathcal{Q}$ with fixed candidate-set size). Limitations and uneven category-level effects are discussed across Section 5 and Section 6 (the Conclusion and Limitations section).

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 discusses the limited benefit on perception- or control-dominated tasks, dependence on the availability and diversity of instruction annotations, the FM energy as a ranking proxy rather than a calibrated trajectory likelihood, and the limited real-world evaluation.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Proposition 1 (instruction-invariant IL optima under sufficiency, CR conditional-median and FM conditional-mean cases), Proposition 2 (CR posterior under an auxiliary Laplace model with common scale κ and a uniform instruction prior), Proposition 3 (FM pseudo-posterior under isotropic, instruction-independent variance), and Theorem 1 (contrastive mutual-information lower bound under the contrastive distribution, with the $\log(K+1)$ form additionally requiring a uniform conditional candidate prior) all state their assumptions inside the statement. Full proofs are given in Appendix A.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5.1 reports backbones, benchmarks, and rollout counts. Appendix C provides the full CIG training algorithm, the gradient-routing two-pass scheme, and the main training configuration table; Appendix B describes counterfactual-set construction; Appendix D reports hyperparameter sensitivity. Appendix F provides the full Stretch 3 protocol (hardware, data collection, preprocessing, deployment, task definitions, success criteria, and trial randomization).

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The prepared release covers RoboCasa training/evaluation, four simulator checkpoints and supplementary Stretch 3 VR collection/replay tools. It does not cover all paper experiments: LIBERO-Plus code, the real-robot policy and the real-robot demonstration dataset are not included. Benchmark download and conversion instructions are supplied; no additional public repository URL is asserted here.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5.1 summarizes the high-level setup; Appendix C.3 gives the full continuation-training configuration (effective batch size, learning rate, CIG steps, λ, δ, κ, K, negative sampling), Appendix D reports a one-at-a-time sensitivity sweep, and Appendix F reports deployment-side hyperparameters (control rate, action chunk length, replanning interval).

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Following common practice for VLA benchmarking, we report point-estimate SR averaged over 50 rollouts per RoboCasa task and the full 10,030-episode LIBERO-Plus protocol. Per-task and per-dimension breakdowns describe performance heterogeneity, but do not estimate run-to-run variance. We do not report confidence intervals or repeated-training-seed uncertainty.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Section 5.1 states RoboCasa continuation training is conducted on $4\times$NVIDIA A100 (80 GB) GPUs. Appendix C.3 gives effective batch sizes and CIG-step counts; Appendix F states real-world inference uses a single RTX 5090 workstation at 15 Hz. Wall-clock runtimes and total compute across all experiments are not reported.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work studies general-purpose vision-language-action policies on standard public manipulation benchmarks and a self-collected real-robot dataset that contains no human subjects, personally identifiable information, or sensitive data. We have reviewed the NeurIPS Code of Ethics and find no deviations.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper studies instruction following for manipulation, but does not provide a dedicated analysis of societal impacts; it does not establish safety guarantees for deployment.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The prepared release provides fine-tuned RoboCasa simulator policies and supplementary robot tools, rather than newly pretrained foundation models or scraped datasets. All training data is benchmark data or self-collected manipulation demonstrations in a controlled kitchen scene.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original works for all benchmarks (RoboCasa [19], LIBERO-Plus [9]), backbones ($\pi_{0.5}$ [21], OpenVLA-OFT [13], π_0 [3], OpenVLA [12]), and software components used in the experimental pipeline. Each asset is used in compliance with its publicly stated license.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The prepared RoboCasa package documents installation, official-data conversion, training, checkpoint formats, inference, evaluation, licenses and supplementary VR recording/replay. Its four simulator checkpoints include normalization statistics and a checksum manifest. The real-robot dataset and policy are outside the release scope.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects. Real-world demonstrations were performed by the authors themselves through VR teleoperation in a controlled kitchen scene.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects beyond the authors performing teleoperation, and therefore does not require IRB approval.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not an original component of the proposed CIG methodology. The vision-language-action backbones used as baselines ($\pi_{0.5}$, OpenVLA-OFT) are pretrained vision-language models adapted for robot control and are properly cited; CIG itself only changes the training objective and gradient routing, not the language-model component.