

Stable Language Guidance for Vision–Language–Action Models

Zhihao Zhan¹, Yuhao Chen¹, Jiaying Zhou¹, Qinhuan Lv¹, Hao Liu¹,
Keze Wang¹, Liang Lin^{1,2,3}, Guangrun Wang^{1,2,3†}

¹Sun Yat-sen University, ²Guangdong Key Lab of Big Data Analysis & Processing, ³X-Era AI Lab
Email: zhanzh6@mail2.sysu.edu.cn, wanggrun@gmail.com

Abstract

Vision-Language-Action (VLA) models have demonstrated impressive capabilities in generalized robotic control; however, they remain notoriously brittle to linguistic perturbations. We identify a critical “modality collapse” phenomenon where strong visual priors overwhelm sparse linguistic signals, causing agents to overfit to specific instruction phrasings while ignoring the underlying semantic intent. To address this, we propose **Residual Semantic Steering (RSS)**, a probabilistic framework that disentangles physical affordance from semantic execution. RSS introduces two theoretical innovations: (1) **Monte Carlo Syntactic Integration**, which approximates the true semantic posterior via dense, LLM-driven distributional expansion, and (2) **Residual Affordance Steering**, a dual-stream decoding mechanism that explicitly isolates the causal influence of language by subtracting the visual affordance prior. Theoretical analysis suggests that RSS effectively maximizes the mutual information between action and intent while suppressing visual distractors. Empirical results across diverse manipulation benchmarks demonstrate that RSS achieves state-of-the-art robustness, maintaining performance even under adversarial linguistic perturbations.

1 Introduction

The core promise of Embodied AI is the ability to map high-level intent to low-level control. Formally, we seek a policy $\pi(a|o, z)$ where o is the observation and z is the latent semantic intent. However, in practice, we only have access to linguistic realizations $l \sim p(l|z)$. Current Vision-Language-Action (VLA) models approximate $\pi(a|o, l)$, but often learn a degenerate mapping where $\pi(a|o, l_i) \not\approx \pi(a|o, l_j)$ even when l_i and l_j share the same intent z .

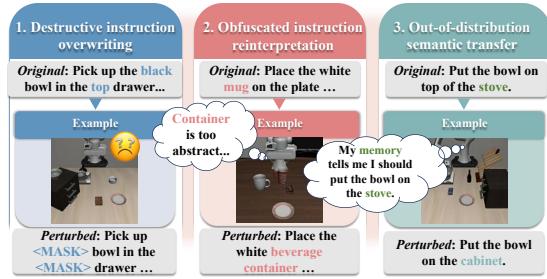


Figure 1: Taxonomy of Language Instruction Perturbations. We identify three distinct failure modes in VLA instruction following: (1) **Destructive Instruction Overwriting**, where critical semantic tokens are lost or masked (e.g., masking the drawer location); (2) **Obfuscated Instruction Reinterpretation**, where the model fails to ground synonymous or verbose descriptions (e.g., “beverage container” vs. “mug”); and (3) **Out-of-Distribution Semantic Transfer**, where the instruction targets a valid but unlearned goal configuration (e.g., placing on a “cabinet” instead of the training-set “stove”).

Recent empirical audits validate this failure mode. Analyses on the **Libero-Plus** benchmark reveal a pervasive “instruction blindness,” where models frequently ignore language inputs entirely, defaulting to the most probable action given the scene alone (Fei et al., 2025). Furthermore, evidence from **Libero-Pro** highlights a critical lack of grounding; even when language is processed, models exhibit an overdependence on rote pattern execution rather than true semantic interpretation, leading to catastrophic failure when instructions deviate from training templates (Zhou et al., 2025). As illustrated in Figure 1, these failures manifest through three primary categories of linguistic variation: destructive instruction overwriting, obfuscated reinterpretation of synonyms, and out-of-distribution semantic transfer.

We argue that this fragility stems from two sources:

1. Manifold Sparsity:

The training data covers

[†]Corresponding author.

a vanishingly small support of the syntactic distribution $p(l|z)$, leading to overfitting on surface statistics.

2. **Prior Dominance:** In the high-dimensional joint distribution $p(a|o, l)$, the visual signal o contains dense, high-frequency information (edges, textures) that dominates the gradient, causing the model to default to a “visual affordance prior” (e.g., “grasp the nearest object”) regardless of the text.

We introduce **Residual Semantic Steering (RSS)**, a framework that addresses these challenges through two theoretical innovations. First, to resolve manifold sparsity, we employ **Monte Carlo Syntactic Integration**. Instead of relying on single data points, we treat the instruction as a seed and utilize an Oracle Teacher (LLM) to generate a dense syntactic neighborhood. By optimizing an Expected Semantic Loss over this expanded distribution, we force the policy to marginalize out syntactic noise and approximate the true posterior $\pi(a|o, z)$. Second, to counteract prior dominance, we introduce **Residual Affordance Steering**. We reinterpret the “unconditional” forward pass not as a null baseline, but as the Base Affordance Distribution $s(a|o, \emptyset)$ —capturing physically feasible actions independent of intent. By subtracting this visual prior from the conditional logits, we isolate a Pure Semantic Signal that represents the causal influence of language. Unlike standard Classifier-Free Guidance (CFG) (Ho and Salimans, 2022) which acts as a “quality booster” in generative models, RSS functions as a Bias Suppressor in control, mathematically penalizing actions driven solely by visual instinct.

Contributions. Our main contributions are summarized as follows:

- We propose Residual Affordance Steering that acts as a Bias Suppressor. By subtracting the Base Affordance Distribution (visual prior) from the conditional logits, it isolates the Pure Semantic Signal, effectively restoring the rank of linguistic features relative to visual dominators.
- We introduce Monte Carlo Syntactic Integration, a training strategy that utilizes an Oracle Teacher to generate dense linguistic neighborhoods. By optimizing an Expected Semantic Loss over this expanded distribution, we

approximate the true semantic posterior, ensuring the policy is invariant to surface-level syntactic perturbations.

- We empirically demonstrate that RSS achieves state-of-the-art robustness, effectively mitigating “instruction blindness” and preventing rote pattern execution by decoupling semantic intent from visual affordances.

2 Related Work

Vision-Language-Action Architectures. The field has rapidly evolved from early large-scale imitators like RT-1 (Brohan et al., 2022) and RT-2 (Zitkovich et al., 2023) to efficient open-source models like OpenVLA (Kim et al., 2024). Recent autoregressive approaches focus on specific capabilities: SpatialVLA (Qu et al., 2025) incorporates 3D spatial cues, OpenVLA-OFT (Kim et al., 2025) optimizes continuous action tuning, and π_0 FAST (Pertsch et al., 2025) enhances training efficiency. Parallel work integrates chain-of-thought reasoning to guide planning (CoT-VLA (Zhao et al., 2025), GR-1 (Wu et al., 2023)). Simultaneously, diffusion-based control has matured from the seminal Diffusion Policy (Chi et al., 2023) to scalable transformers like CogACT (Li et al., 2024) and RDT (Liu et al., 2024). State-of-the-art generalist frameworks now employ flow matching (π_0 (Black et al., 2024), $\pi_{0.5}$ (Intelligence et al., 2025)), \mathcal{E}_0 (Zhan et al., 2025), dual-system designs (OneTwoVLA (Lin et al., 2025), GR00T N1 (Bjorck et al., 2025)), and decoupling approaches (Li et al., 2025) to achieve broad physical generalization.

Language Guidance and Modality Imbalance. Despite these architectural advances, balancing multi-modal inputs remains a critical bottleneck. Recent audits on the **Libero-Plus** (Fei et al., 2025) and **Libero-Pro** (Zhou et al., 2025) benchmarks reveal that VLA agents frequently suffer from “instruction blindness” or rote execution, as dense high-frequency visual signals tend to overwhelm sparse linguistic tokens. A notable architectural solution is **RDT-1B** (Liu et al., 2024), which mitigates this text overshadowing by eschewing simultaneous token injection. Instead, it employs an alternating cross-attention mechanism, injecting image and text tokens in successive layers. This strategy explicitly preserves the magnitude of linguistic gradients during deep fusion, ensuring that instruction signals are not drowned out by the vi-

sual affordance prior.

3 Methodology: The RSS Framework

3.1 Preliminaries

Vision-language-action(VLA) models are commonly trained via imitation learning on large-scale robot demonstration datasets \mathcal{D} . Given an observation o_t and a natural language task instruction l , the objective is to maximize the log-likelihood of an action (or an action chunk) $a_{t:t+H}$:

$$\max_{\theta} \mathbb{E}_{(a_{t:t+H}, o_t, l) \sim \mathcal{D}} [\log \pi_{\theta}(a_{t:t+H} | o_t, l)]. \quad (1)$$

The observation o_t typically comprises one or more visual inputs $\{I_t^1, \dots, I_t^n\}$ and a proprioceptive state q_t encoding the robot’s joint configurations.

Modern VLA architectures inherit the design principles of large vision-language models, employing modality-specific tokenizers to map visual, linguistic, and action inputs into either discrete or continuous token representations (Brohan et al., 2022; Zitkovich et al., 2023; Pertsch et al., 2025; Black et al., 2024; Intelligence et al., 2025). These tokens are processed by a large autoregressive transformer backbone, whose parameters are commonly initialized from pretrained vision-language models. By encoding observations, instructions, and actions into a unified token sequence, the imitation learning objective can be reformulated as a standard next-token prediction problem, enabling the use of scalable training techniques from contemporary language modeling.

We consider a VLA policy parameterized by θ . Our goal is to ensure the policy is invariant to linguistic noise ϵ_l such that $\pi(a|o, l) \approx \pi(a|o, l + \epsilon_l)$.

To achieve this, we propose the **Residual Semantic Steering (RSS)** framework. As illustrated in Figure 2, RSS operates as a dual-stage mechanism: it densifies the linguistic supervision signal via *Monte Carlo Syntactic Integration* to enforce semantic invariance; it actively suppresses visual priors via *Residual Affordance Steering* to isolate the pure semantic intent.

3.2 Monte Carlo Syntactic Integration

Standard Maximum Likelihood Estimation (MLE) minimizes $\mathcal{L} = -\log p_{\theta}(a|o, l)$. However, a single instruction l is a noisy estimator of the true intent z . To learn the true semantic policy $p(a|o, z)$, we must

marginalize over the nuisance variable of syntax:

$$p(a|o, z) = \int p(a|o, l)p(l|z) dl \quad (2)$$

Since this integral is intractable, we employ a **Monte Carlo approximation** using an Oracle Teacher (e.g., a high-capability LLM). We treat the original instruction l_{orig} as a seed and generate a dense neighborhood $\mathcal{N}(l_{orig}) = \{l_1, \dots, l_K\}$ via the Teacher, sampling from the induced distribution $\hat{p}(l|z)$. We then optimize the **Expected Semantic Loss**:

$$\mathcal{L}_{RSS} = \mathbb{E}_{(o, a) \sim \mathcal{D}} \left[\frac{1}{K} \sum_{k=1}^K -\log \pi_{\theta}(a|o, l_k) \right] \quad (3)$$

This formulation forces the encoder to map disparate linguistic inputs $\{l_k\}$ to a unified region in the latent embedding space, explicitly minimizing the conditional entropy $H(A|L)$ with respect to syntactic variation.

3.3 Residual Affordance Steering

Standard CFG combines conditional and unconditional scores. We re-interpret this for the action space to distinguish our method from generative diffusion. Let $s(a|o, l)$ be the logit score of an action. We decompose the decision process into two components:

1. **The Affordance Prior** $s(a|o, \emptyset)$: The probability of an action based solely on visual scene geometry (what is *possible*).
2. **The Semantic Modulation**: The specific shift induced by the instruction.

Standard VLA inference uses $s(a|o, l)$. However, we observe that $s(a|o, l) \approx s(a|o, \emptyset)$ when the language signal is weak or the visual features are overpowering.

To extract the **Pure Semantic Signal**, we calculate the **Residual Vector** Δ_{sem} :

$$\Delta_{sem}(a, o, l) = s(a|o, l) - s(a|o, \emptyset) \quad (4)$$

This subtraction cancels out the visual bias. If the robot wants to grasp a red cup (a_{red}) simply because it is close (visual bias), $s(a_{red}|o, \emptyset)$ will be high. The residual Δ_{sem} removes this bias, leaving only the text’s contribution.

The final **Steered Policy** is:

$$\tilde{\pi}(a|o, l) \propto \exp(s(a|o, \emptyset) + \gamma \cdot \Delta_{sem}(a, o, l)) \quad (5)$$

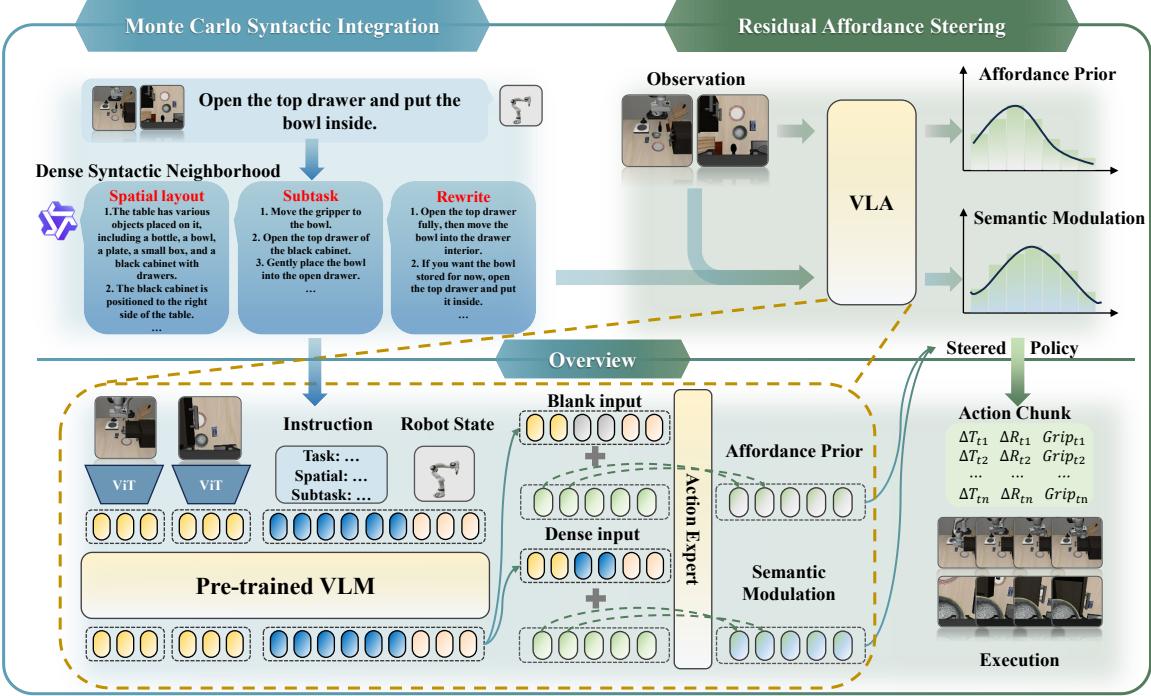


Figure 2: **Overview of Residual Semantic Steering (RSS).** To combat instruction blindness, RSS operates in two stages. **Left:** *Monte Carlo Syntactic Integration* utilizes an Oracle Teacher to generate a dense linguistic neighborhood around a seed instruction. Optimizing over this distribution forces the policy to learn representations that are invariant to syntactic perturbations. **Right:** *Residual Affordance Steering* mitigates visual prior dominance. By subtracting the unconditioned “visual instinct” (Base Affordance) from the standard prediction, we isolate and amplify the residual semantic signal, ensuring the policy follows the specific user intent rather than generic visual attractors.

where $\gamma > 1$ is the **Steering Coefficient**.

Novelty vs. Standard Classifier-Free Guidance (CFG) (Ho and Salimans, 2022): While mechanically similar to CFG, RSS differs conceptually. CFG in diffusion acts as a quality booster (trading diversity for fidelity). RSS acts as a **Bias Suppressor**. We use the null-text pass to explicitly model the “visual instinct” of the robot and then mathematically penalize actions that are driven *only* by that instinct and not confirmed by the text.

3.4 Theoretical Analysis

Proposition 1 (Visual Bias Decoupling). Let the logit function be approximated linearly as $S(o, l) = W_v\phi(o) + W_l\psi(l) + C$, where C represents confounding interactions, $\phi(o) \in \mathbb{R}^d$ is the visual embedding vector, $\psi(l) \in \mathbb{R}^d$ is the linguistic embedding vector, $W_v, W_l \in \mathbb{R}^d$ are the projection weights for the visual and linguistic modalities, respectively, ϵ represents higher-order interaction terms and bias, assumed to be negligible for this first-order analysis. The standard inference yields a signal-to-noise ratio dependent on $\|W_v\|/\|W_l\|$. In VLA tasks, $\|W_v\| \gg \|W_l\|$. The Residual Steering

yields:

$$\begin{aligned}\tilde{S} &= S(o, \emptyset) + \gamma(S(o, l) - S(o, \emptyset)) \\ &\approx W_v\phi(o) + \gamma W_l\psi(l)\end{aligned}\quad (6)$$

By setting $\gamma > 1$, we artificially restore the rank of the language features, effectively orthogonalizing the semantic vector from the visual manifold.

4 Experiment

4.1 Training Setting

We adopt π_0 (Black et al., 2024) and $\pi_{0.5}$ (Intelligence et al., 2025) as our baseline models. The pre-trained vision language model backbone is based on Gemma (Team et al., 2024). The initial base weights are obtained through large-scale pre-training on a diverse and heterogeneous collection of robotic datasets (O’Neill et al., 2024). Building upon these pretrained representations, we fine-tune the models on task-specific data and incorporate our proposed modifications.

To implement *Monte Carlo Syntactic Integration*, we utilize Qwen2.5-VL (Bai et al., 2025) as

Table 1: **Robustness evaluation under Destructive Instruction Overwriting.** Results are reported as task Success Rates (SR) in percentages (%). **Bold** and underlined entries denote the best and second-best performance per column, respectively. Values in parentheses represent the absolute improvement over the corresponding baseline. **RAS**: Residual Affordance Steering; **MCSI**: Monte Carlo Syntactic Integration.

Model	Origin SR (%) ↑	Multi SR (%) ↑	Blank SR (%) ↑	Rand SR (%) ↑	M2 SR (%) ↑	M4 SR (%) ↑	M6 SR (%) ↑	M8 SR (%) ↑	Simple SR (%) ↑	Average SR (%) ↑
π_0 (Black et al., 2024)										
base	94.15	91.30	25.20	89.35	72.65	42.50	22.15	7.80	26.25	52.37
+ RAS	90.65	89.90	63.40	88.70	78.45	55.05	33.55	17.95	62.50	64.46 (+12.09)
+ MCSI	94.55	93.45	41.85	92.95	88.85	77.75	63.85	52.80	39.85	71.77 (+19.40)
+ RAS & MCSI	93.35	91.35	69.65	94.95	92.05	85.85	77.95	69.90	64.90	82.22 (+29.85)
$\pi_{0.5}$ (Intelligence et al., 2025)										
base	95.15	95.45	50.05	95.20	92.65	82.70	69.55	55.45	46.90	75.90
+ RAS	<u>96.65</u>	96.80	70.50	<u>95.95</u>	93.90	86.65	<u>79.30</u>	71.05	<u>69.10</u>	84.43 (+8.53)
+ MCSI	98.25	98.00	46.20	97.45	<u>96.05</u>	<u>87.20</u>	73.40	57.95	46.20	77.86 (+1.96)
+ RAS & MCSI	96.60	97.50	70.25	97.45	<u>96.35</u>	92.00	<u>84.55</u>	<u>77.50</u>	70.60	86.98 (+11.08)

Table 2: **Obfuscated instruction reinterpretation result.** **Bold** entries denote the best results per column, and underlined entries indicate the second-best. Values in parentheses report the absolute SR improvement (in percentage points) over the corresponding baseline under the same setting. All numbers are percentages (%). **RAS**: Residual Affordance Steering; **MCSI**: Monte Carlo Syntactic Integration.

Model	R0 SR (%) ↑	R1 SR (%) ↑	R2 SR (%) ↑	R3 SR (%) ↑	R4 SR (%) ↑	Average SR (%) ↑
π_0 (Black et al., 2024)						
base	91.4	55.8	7.4	28.4	42.4	45.08
+ RAS	90.0	<u>59.0</u>	11.4	40.2	46.0	49.32 (+4.24)
+ MCSI	92.6	83.6	28.0	71.2	58.6	66.80 (+21.72)
+ RAS & MCSI	88.4	85.4	26.8	80.0	47.0	65.52 (+20.44)
$\pi_{0.5}$ (Intelligence et al., 2025)						
base	95.0	93.2	<u>30.4</u>	90.6	68.6	75.56
+ RAS	<u>97.0</u>	<u>94.6</u>	26.4	86.4	<u>77.6</u>	76.40 (+0.84)
+ MCSI	<u>97.0</u>	<u>94.6</u>	31.4	84.4	70.6	75.60 (+0.04)
+ RAS & MCSI	97.6	97.2	30.2	<u>89.4</u>	79.0	78.68 (+3.12)

an oracle teacher to generate dense semantic neighborhoods during training. At evaluation time, we employ ChatGPT-5.2 (OpenAI, 2025) to systematically rewrite all task instructions, ensuring consistent and controlled linguistic variations across experiments.

Each model was trained for a total of 30,000 steps with a batch size of 32. The learning rate followed a cosine decay schedule, implemented as *CosineDecaySchedule*, with a warm-up phase of 10,000 steps, a peak learning rate of 5×10^{-5} and a final learning rate of 5×10^{-5} . An exponential moving average (EMA) with a decay rate of 0.999 was applied. During inference, a single NVIDIA RTX 3090 GPU was used for model evaluation and deployment on the server side.

4.2 Simulation Experiment

We adopt LIBERO (Liu et al., 2023) as our primary simulation benchmark for evaluating VLA models.

It comprises four task categories (*LIBERO-Spatial*, *LIBERO-Object*, *LIBERO-Goal*, and *LIBERO-10*) which jointly challenge VLA models along multiple dimensions, including spatial reasoning, object-centric manipulation, goal specification, and multi-step task execution. As a result, it has become a highly competitive benchmark, with many state-of-the-art methods employing sophisticated architectural designs and advanced training strategies to maximize performance.

To further investigate how natural language instructions guide VLA models, we extend LIBERO with a set of *controlled instruction variants* that explicitly probe linguistic robustness and generalization. Specifically, we consider three categories of instruction perturbations (See Figure 1). (1) **Destructive instruction overwriting**, where critical semantic components of the original instruction are intentionally corrupted or removed, thereby disrupting the instruction’s compositional mean-

Table 3: **Examples of obfuscated instruction reinterpretation variants (R0–R4) on LIBERO-Goal.** All variants preserve the original task semantics while introducing different forms of linguistic perturbations.

Variant	Function	Examples
Original	/	Put the wine bottle on top of the cabinet.
R0	Multiword Substitution	Move the wine bottle onto the cabinet top.
R1	Distraction	Once you’ve got a steady hold, put the wine bottle on the cabinet top.
R2	Common Sense	Set the sealed container typically used for pouring grape-based drinks on the upper face of the standing storage furniture.
R3	Reasoning Chain	Move the bottle over the cabinet, then release it once it is stable on top.
R4	Confusion	Regardless of the drawers being open or closed, place the wine bottle on top of the cabinet.

Table 4: **Out-of-distribution semantic transfer result.** **Bold** entries denote the best results per column. The base model is $\pi_{0.5}$ (Intelligence et al., 2025). **RAS**: Residual Affordance Steering; **MCSI**: Monte Carlo Syntactic Integration.

Model	10-steps	100-steps	1000-steps	Average
	SR (%) ↑	SR (%) ↑	SR (%) ↑	SR (%) ↑
base	27.0	31.0	91.0	49.67
+ RAS	17.0	29.0	98.0	48.00
+ MCSI	28.0	42.0	97.0	55.67
+ RAS & MCSI	31.0	31.0	97.0	53.00

ing. (2) **Obfuscated instruction reinterpretation**, which introduces syntactically valid yet semantically peripheral or potentially distracting descriptions while preserving the original task intent to evaluate robustness against semantic distraction rather than outright corruption. (3) **Out-of-distribution (OOD) semantic transfer**, where novel task instructions are constructed by recombining object concepts observed during training into previously unseen compositions, resulting in tasks that do not appear in the original training distribution.

Destructive instruction overwriting. To explicitly probe the extent to which VLA policies rely on genuine language grounding rather than spurious correlations, we design a set of *destructive instruction overwriting* variants that deliberately corrupt or erase critical semantic information in the original task instructions. Specifically, we consider five types of perturbations: **Blank**, where the instruction is replaced by an empty string, completely removing linguistic input; **Simple**, where all instructions are substituted with a generic and uninformative phrase (e.g., “Do something”); **Multi**, where we leverage a large language model to generate multiple paraphrases of the original instruction via word substitution or rephrasing, and randomly

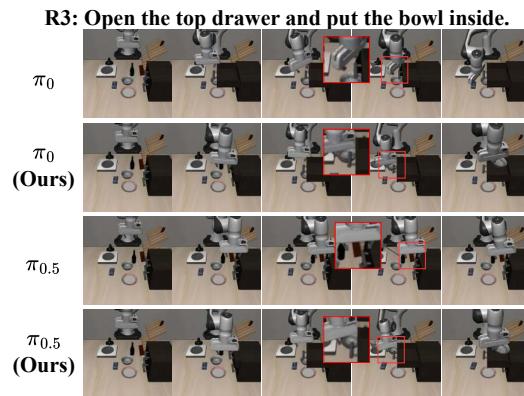


Figure 3: **Comparison on the LIBERO variant R3 Reasoning Chain.** In the “open the top drawer and put the bowl inside” task, our model consistently outperforms the baseline under reasoning-chain-perturbed instructions, demonstrating a stronger ability to follow multi-step semantic constraints and accurately complete the task despite increased linguistic complexity.

sample one variant during evaluation; **Rand**, which randomly permutes the word order within the instruction, disrupting syntactic structure while preserving the vocabulary; and **Mask**, where each word is independently replaced by a `<MASK>` token with varying probabilities, progressively degrading semantic content.

As shown in Table 1, these destructive perturbations lead to substantial performance degradation across most settings, particularly under **Blank**, **Simple**, and high-ratio **Mask** conditions, indicating that removing or severely corrupting linguistic cues significantly hinders task execution. Notably, models augmented with RAS and MCSI demonstrate improved resilience under destructive overwriting, achieving higher average success rates across all variants. This observation implies that richer vision-language alignment can partially mitigate the reliance on brittle instruction patterns, encouraging

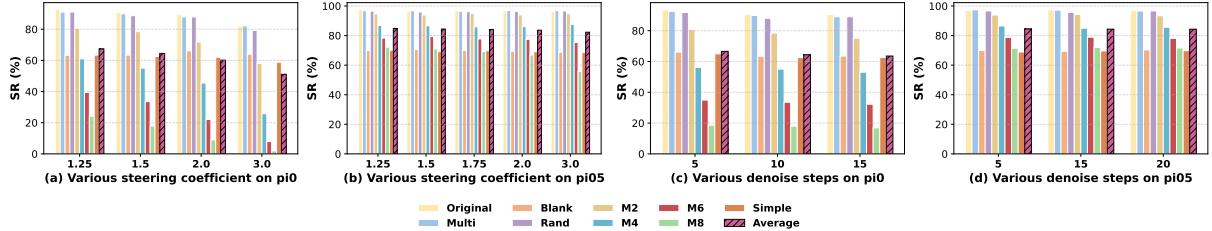


Figure 4: **Ablation of steering coefficient and denoising steps on destructive instruction overwritten.** Success rates (SR, %) across instruction variants under different steering coefficients for π_0 (a) and $\pi_{0.5}$ (b), and different denoising steps for π_0 (c) and $\pi_{0.5}$ (d), illustrating the effect of guidance and generation depth on robustness to instruction perturbations.

more semantically grounded policy behavior rather than shortcut exploitation.

Obfuscated instruction reinterpretation. We introduce a set of semantically preserved but potentially obfuscated instruction variants to evaluate linguistic robustness on the LIBERO-Goal benchmark. Specifically, we consider: **R0 (Multiword Substitution)**, which applies lightweight synonym or phrase replacements; **R1 (Distraction)**, which augments instructions with task-irrelevant contextual content; **R2 (Common Sense)**, which replaces object names with commonsense-based descriptive phrases; **R3 (Reasoning Chain)**, which reformulates instructions to emphasize implicit reasoning or final-state constraints; and **R4 (Confusion)**, which introduces distractor objects via explicit negation while preserving the original task goal.

As shown in Table 2, there are clear performance disparities across different reinterpretation variants, highlighting how VLA models respond to semantically preserved yet linguistically challenging instructions. Among all settings, **R0** incurs only minor degradation, indicating that most models are largely insensitive to superficial lexical substitutions. In contrast, **R1** and **R2** lead to more pronounced drops, suggesting that additional irrelevant context and abstract commonsense descriptions substantially increase the difficulty of extracting task-relevant semantics.

Notably, **R3** and **R4** constitute the most challenging conditions. **R3** requires the model to correctly interpret implicit reasoning structures and final-state constraints, while **R4** explicitly introduces distractor objects that co-occur in other tasks, stressing the model’s ability to resist spurious correlations. Performance under these two variants, therefore, serves as a stronger indicator of true language grounding rather than shallow pattern match-

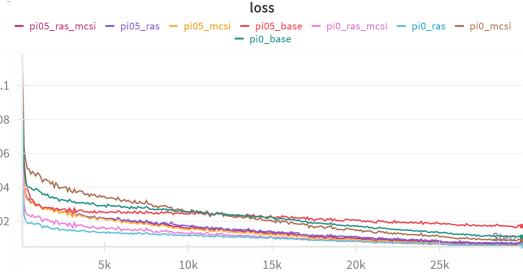


Figure 5: **Training loss curves.** We report the training loss trajectories of different model variants throughout optimization. **RAS**: Residual Affordance Steering; **MCSI**: Monte Carlo Syntactic Integration.

ing.

Across most variants, models augmented with RAS and MCSI demonstrate improved robustness, achieving the highest average success rate. This trend suggests that richer vision–language alignment encourages policies to rely less on brittle lexical cues and more on semantically grounded representations, improving generalization under obfuscated yet valid instructions.

Out-of-distribution semantic transfer. To construct a controlled out-of-distribution (OOD) evaluation setting, we remove two target tasks from the original LIBERO-Goal training set, while ensuring that the objects involved still appear in other training tasks. This design isolates distribution shift at the level of *object–goal composition*, rather than introducing entirely unseen objects. To avoid trivial failure due to complete task absence, we first fine-tune the baseline model for 6,000 steps using the remaining in-distribution data, corresponding to approximately 20% of full training. We then perform few-shot adaptation using a small number of demonstrations from the two held-out OOD tasks, and evaluate performance under varying shot budgets.

As shown in Table 4, the vanilla model exhibits limited generalization under low-shot regimes, particularly with 10-shot adaptation, indicating difficulty in transferring previously learned object semantics to novel task compositions. Meanwhile, although the average success rate of the baseline appears relatively high (in 100-shot and 1,000-shot adaptation), a closer inspection reveals that this performance is dominated by overfitting to a single held-out task, while consistently failing on the other OOD task. In contrast, our method achieves successful execution across both held-out tasks, reflecting improved robustness to compositional distribution shifts.

Notably, RAS and MCSI improve few-shot performance to different extents across regimes, with MCSI consistently achieving stronger gains. While combining RAS with MCSI offers additional benefits, MCSI alone delivers the most effective improvement, indicating its dominant role in enhancing semantic transfer and reducing reliance on task-specific memorization.

4.3 Ablation Study

We conduct a comprehensive ablation study on two key hyperparameters in RSS: the Residual Affordance Steering (RAS) coefficient and the number of denoising steps. All evaluations are performed under destructive instruction overwriting benchmarks.

Residual Affordance Steering. Figure 4 (a) and Figure 4 (b) report performance under different RAS coefficients with the number of denoising steps fixed to 10. We observe that moderate steering coefficients consistently improve robustness under semantic-preserving perturbations, by strengthening the alignment between language conditions and action generation. However, excessively large coefficients significantly amplify sensitivity to corrupted or obfuscated instructions, leading to pronounced performance drops under destructive perturbations. This trend indicates that over-strong RAS encourages over-conditioning on unreliable linguistic signals, thereby exacerbating shortcut exploitation.

Denoising steps. Figure 4 (c) and Figure 4 (d) analyze the effect of varying the number of denoising steps with the RAS coefficient fixed at 1.5 and 1.25, respectively. While increasing the number of steps can improve success rates in certain individual settings, we observe that larger step counts do not consistently translate to higher overall perfor-

mance, and in some cases lead to a slight degradation in average success rate. This suggests that excessive denoising primarily refines low-level action details without providing additional semantic benefits.

Joint effect. When considering the interaction between RAS and denoising steps, we find that the overall impact of varying the number of denoising steps is relatively limited under smaller steering coefficients. In this regime, the model is less sensitive to sampling depth, as weaker conditional forcing mitigates over-reliance on potentially noisy language inputs. Overall, the optimal configuration reflects a trade-off between conditional strength and sampling stability, with moderate RAS coefficients and a moderate number of denoising steps yielding the most robust performance across language perturbations.

4.4 Training Loss Curves

Figure 5 illustrates the training loss trajectories of different model variants. Across all settings, the loss decreases rapidly during the early training stage and gradually stabilizes as optimization proceeds, indicating stable convergence behavior. Compared to their corresponding baseline models, variants equipped with RAS and MCSI achieve lower loss values when training. This suggests that the proposed components provide a more effective training signal and facilitate smoother optimization, leading to improved learning efficiency rather than merely affecting the final performance.

5 Conclusion

We identify a core limitation of current VLA models: their failure to robustly ground linguistic intent, resulting in instruction blindness and over-reliance on visual affordance priors. To address this, we propose **Residual Semantic Steering (RSS)**, which combines **Monte Carlo Syntactic Integration** to overcome instruction manifold sparsity with **Residual Affordance Steering** to suppress visually driven biases. By explicitly disentangling semantic intent from visual priors, RSS restores consistent language-action alignment and enables robust generalization under diverse linguistic perturbations. Empirical results demonstrate that RSS significantly improves instruction robustness and semantic grounding, establishing a principled path toward reliable language-conditioned robotic control.

Limitations

A limitation of Residual Affordance Steering (RAS) is its conservative behavior when presented with extremely vague or underspecified instructions (e.g., generic prompts like "do something"). Because our inference mechanism explicitly suppresses the "Base Affordance Distribution"—which drives standard models to aggressively execute the most common visual trajectory regardless of input—our model may exhibit hesitation or inaction when the language signal lacks sufficient semantic content to steer the policy. Unlike baseline methods that tend to ignore linguistic ambiguity and revert to rote visual pattern matching (often executing an action simply because it was frequent in the training set), our framework effectively requires semantically meaningful commands to initiate motion. This dependency prevents the model from "hallucinating" actions based on visual priors alone, ensuring that the robot does not default to unsafe autopilot behaviors when the user's intent is unclear.

References

- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.
- Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, and 1 others. 2025. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*.
- Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, and 1 others. 2024. Pi0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*.
- Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, and 1 others. 2022. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*.
- Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. 2023. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668.
- Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, and 1 others. 2025. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*.
- Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.
- Zhi Hou, Tianyi Zhang, Yuwen Xiong, Haonan Duan, Hengjun Pu, Ronglei Tong, Chengyang Zhao, Xizhou Zhu, Yu Qiao, Jifeng Dai, and 1 others. 2025. Dita: Scaling diffusion transformer for generalist vision-language-action policy. *arXiv preprint arXiv:2503.19757*.
- Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, and 1 others. 2025. Pi0.5: A Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*.
- Moo Jin Kim, Chelsea Finn, and Percy Liang. 2025. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*.
- Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, and 1 others. 2024. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.
- Xixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, and 1 others. 2024. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*.
- Weiqi Li, Quande Zhang, Ruifeng Zhai, Liang Lin, and Guangrun Wang. 2025. Vla models are more generalizable than you think: Revisiting physical and spatial modeling. *arXiv preprint arXiv:2512.02902*.
- Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. 2025. Onetwovla: A unified vision-language-action model with adaptive reasoning. *arXiv preprint arXiv:2505.11917*.
- Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. 2023. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791.
- Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. 2024. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*.

- Oier Mees, Dibya Ghosh, Karl Pertsch, Kevin Black, Homer Rich Walke, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, and 1 others. 2024. Octo: An open-source generalist robot policy. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*.
- OpenAI. 2025. Chatgpt. <https://chat.openai.com/>. Version 5.2.
- Abby O’Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, and 1 others. 2024. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE.
- Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. 2025. FAST: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*.
- Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and 1 others. 2025. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*.
- Moritz Reuss, Ömer Erdinç Yağmurlu, Fabian Wenzel, and Rudolf Lioutikov. 2024. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. *arXiv preprint arXiv:2407.05996*.
- Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.
- Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. 2023. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*.
- Zhihao Zhan, Jiaying Zhou, Likui Zhang, Qinhan Lv, Hao Liu, Jusheng Zhang, Weizheng Li, Ziliang Chen, Tianshui Chen, Keze Wang, and 1 others. 2025. *mathcal{E}_0*: Enhancing generalization and fine-grained control in vla models via continuized discrete diffusion. *arXiv preprint arXiv:2511.21542*.
- Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, and 1 others. 2025. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1702–1713.
- Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. 2024. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*.
- Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. 2025. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*.
- Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, and 1 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR.

A Proof of Proposition 1: Visual Bias Decoupling via Residual Steering

In this section, we provide a formal derivation of Proposition 1, demonstrating how Residual Semantic Steering (RSS) restores the rank of linguistic features in the presence of dominant visual priors.

A.1 Setup and Definitions

Let $S(a|o, l) \in \mathbb{R}$ denote the logit score for a specific action a , given observation o and instruction l . We analyze the behavior of the model’s final layer. We assume the logit can be approximated as a linear combination of the disentangled features in the penultimate layer:

$$S(a|o, l) = W_v^\top \phi(o) + W_l^\top \psi(l) + \epsilon \quad (7)$$

where:

- $\phi(o) \in \mathbb{R}^d$ is the visual embedding vector.
- $\psi(l) \in \mathbb{R}^d$ is the linguistic embedding vector.
- $W_v, W_l \in \mathbb{R}^d$ are the projection weights for the visual and linguistic modalities, respectively.
- ϵ represents higher-order interaction terms and bias, assumed to be negligible for this first-order analysis.

Assumption 1 (Visual Dominance): In VLA models, the gradient flow during training is dominated by the dense visual signal, leading to spectral norms such that $\|W_v\| \gg \|W_l\|$. This implies that standard inference is driven primarily by $\phi(o)$.

Assumption 2 (Null-Text Baseline): When the instruction is dropped (denoted as \emptyset), the linguistic feature vector collapses to a null or bias state, denoted as $\psi(\emptyset) \approx \mathbf{0}$. Thus, the unconditional forward pass represents the pure visual affordance:

$$S(a|o, \emptyset) \approx W_v^\top \phi(o) \quad (8)$$

A.2 Derivation of Residual Steering

The Residual Semantic Steering formula is defined as:

$$\tilde{S}(a) = S(a|o, \emptyset) + \gamma (S(a|o, l) - S(a|o, \emptyset)) \quad (9)$$

where γ is the steering coefficient.

Substituting the linear approximations into the residual term:

$$\Delta_{sem} = S(a|o, l) - S(a|o, \emptyset) \quad (10)$$

$$= (W_v^\top \phi(o) + W_l^\top \psi(l)) \quad (11)$$

$$- (W_v^\top \phi(o) + W_l^\top \psi(\emptyset)) \quad (12)$$

$$= W_l^\top (\psi(l) - \mathbf{0}) \quad (13)$$

$$= W_l^\top \psi(l) \quad (14)$$

Note that the visual term $W_v^\top \phi(o)$ is explicitly cancelled out in the residual term.

Now, we reconstruct the full steered logit $\tilde{S}(a)$:

$$\tilde{S}(a) = S(a|o, \emptyset) + \gamma \cdot \Delta_{sem} \quad (15)$$

$$= W_v^\top \phi(o) + \gamma (W_l^\top \psi(l)) \quad (16)$$

A.3 Analysis of Signal-to-Noise Ratio (SNR)

We define the Semantic Signal-to-Noise Ratio (SNR) as the ratio of the linguistic contribution to the visual contribution.

Case 1: Standard Inference ($\gamma = 1$)

$$\text{SNR}_{std} = \frac{|W_l^\top \psi(l)|}{|W_v^\top \phi(o)|} \quad (17)$$

Given Assumption 1 ($\|W_v\| \gg \|W_l\|$), $\text{SNR}_{std} \rightarrow 0$. The text has minimal impact on the action ranking.

Case 2: Residual Steering ($\gamma > 1$)

$$\text{SNR}_{rss} = \frac{|\gamma W_l^\top \psi(l)|}{|W_v^\top \phi(o)|} = \gamma \cdot \text{SNR}_{std} \quad (18)$$

A.4 Conclusion

By choosing $\gamma \gg 1$, we linearly amplify the linguistic contribution without altering the visual affordance landscape. Effectively, we synthesize a new weight vector $\tilde{W}_l = \gamma W_l$, artificially restoring the balance between modalities. This proves that RSS orthogonalizes the semantic intent from the visual prior, as stated in Proposition 1. \square

B Detailed Results

This section provides a comprehensive breakdown of task-level performance across all LIBERO sub-tasks. Unlike the aggregated results reported in the main paper, the tables in this section present success rates for each individual subtask, enabling a fine-grained examination of model behavior under different instruction conditions.

Specifically, Table 5 reports performance under the original, unmodified instructions, serving as a reference for canonical task execution. Tables 6, 7, 8, 9, and 10 report detailed subtask-level success rates under different instruction perturbation settings.

C Detailed Ablation Results

Tables 11, 12, and 13 present detailed ablation results on the steering coefficient (SC) and the number of denoising steps under destructive instruction overwriting. All results are reported as average success rates across different language perturbation benchmarks.

As shown in Table 11, the steering coefficient plays a critical role in balancing robustness and over-steering. Excessively large coefficients lead to noticeable performance degradation. This indicates that overly strong residual affordance steering can dominate the policy update, thereby reducing adaptability under corrupted language inputs.

Tables 12 and 13 further show the effect of denoising steps for π_0 and $\pi_{0.5}$, respectively. Notably, $\pi_{0.5}$ exhibits stronger overall stability across different step configurations, indicating that a stronger base language grounding reduces sensitivity to inference-time hyperparameters.

Overall, these results demonstrate that robustness under destructive instruction overwriting depends on a careful balance between steering strength and inference depth. Moderate steering coefficients combined with sufficient denoising steps yield the most stable and consistent performance across language perturbations.

D Qualitative Analysis under Destructive Instruction Overwriting

Figures 6 and 7 present qualitative rollouts under *destructive instruction overwriting*, where key linguistic tokens are removed from the command. In this setting, the vanilla policies exhibit brittle behavior: although the scene remains visually unchanged, the policy fails to consistently localize

Table 5: **LIBERO (original instruction) task performance (success rate, SR) results.** Results are reported on the four standard LIBERO sub-tasks (Libero-Spatial, Libero-Object, Libero-Goal, and Libero-Long) along with the overall average. This setting evaluates model performance under unmodified, canonical task instructions. **RAS:** Residual Affordance Steering; **MCSI:** Monte Carlo Syntactic Integration.

Model	Libero-Spatial SR (%) ↑	Libero-Object SR (%) ↑	Libero-Goal SR (%) ↑	Libero-Long SR (%) ↑	Average SR (%) ↑
Diffusion Policy (Chi et al., 2023)	78.3	92.5	68.3	50.5	72.40
MDT (Reuss et al., 2024)	78.5	87.5	73.5	64.8	76.10
OpenVLA (Kim et al., 2024)	84.7	88.4	79.2	53.7	76.50
Octo (Mees et al., 2024)	78.9	85.7	84.6	51.1	75.10
Dita (Hou et al., 2025)	84.2	96.3	85.4	63.8	82.40
TraceVLA (Zheng et al., 2024)	84.6	85.2	75.1	54.1	74.80
SpatialVLA (Qi et al., 2025)	88.2	89.9	78.6	55.5	78.10
π_0 FAST (Pertsch et al., 2025)	96.4	96.8	88.6	60.2	85.50
π_0 (Black et al., 2024)	96.8	98.8	95.8	85.2	94.15
$\pi_0 + \text{RAS}$	91.4	97.4	92.2	81.6	90.65
$\pi_0 + \text{MCSI}$	96.6	99.8	94.8	87.0	94.55
$\pi_0 + \text{RAS \& MCSI}$	97.4	98.8	93.4	83.8	93.35
$\pi_{0.5}$ (Intelligence et al., 2025)	95.4	98.4	97.2	89.6	95.15
$\pi_{0.5} + \text{RAS}$	97.8	99.4	96.6	92.8	96.65
$\pi_{0.5} + \text{MCSI}$	99.8	99.6	99.6	94.0	98.25
$\pi_{0.5} + \text{RAS \& MCSI}$	98.6	99.2	95.8	92.8	96.60

Table 6: **LIBERO (blank instruction) task performance (success rate, SR) results.** In this variant, task instructions are fully blanked to remove explicit linguistic guidance. **RAS:** Residual Affordance Steering; **MCSI:** Monte Carlo Syntactic Integration.

Model	Libero-Spatial SR (%) ↑	Libero-Object SR (%) ↑	Libero-Goal SR (%) ↑	Libero-Long SR (%) ↑	Average SR (%) ↑
π_0 (Black et al., 2024)	29.8	41.2	4.6	25.2	25.20
$\pi_0 + \text{RAS}$	77.4	95.4	6.8	74.0	63.40
$\pi_0 + \text{MCSI}$	57.0	50.8	9.8	49.8	41.85
$\pi_0 + \text{RAS \& MCSI}$	88.0	98.8	9.8	82.0	69.65
$\pi_{0.5}$ (Intelligence et al., 2025)	65.0	51.8	11.4	72.0	50.05
$\pi_{0.5} + \text{RAS}$	87.0	99.4	10.8	84.8	70.50
$\pi_{0.5} + \text{MCSI}$	43.4	50.2	13.8	77.4	46.20
$\pi_{0.5} + \text{RAS \& MCSI}$	88.4	99.4	10.2	83.0	70.25

Table 7: **LIBERO (simple-word instruction) task performance (success rate, SR) results.** All task instructions are replaced by a single generic command (e.g., "do something"), completely removing the original semantic content. **RAS:** Residual Affordance Steering; **MCSI:** Monte Carlo Syntactic Integration.

Model	Libero-Spatial SR (%) ↑	Libero-Object SR (%) ↑	Libero-Goal SR (%) ↑	Libero-Long SR (%) ↑	Average SR (%) ↑
π_0 (Black et al., 2024)	29.2	43.4	2.2	30.2	26.25
$\pi_0 + \text{RAS}$	79.0	96.2	5.4	69.4	62.50
$\pi_0 + \text{MCSI}$	58.6	48.8	9.0	43.0	39.85
$\pi_0 + \text{RAS \& MCSI}$	86.6	93.4	9.4	70.2	64.90
$\pi_{0.5}$ (Intelligence et al., 2025)	47.8	56.2	10.8	72.8	46.90
$\pi_{0.5} + \text{RAS}$	86.6	99.2	9.2	81.4	69.10
$\pi_{0.5} + \text{MCSI}$	43.2	48.2	15.0	78.4	46.20
$\pi_{0.5} + \text{RAS \& MCSI}$	88.0	100.0	10.0	84.4	70.60

Table 8: **LIBERO (multi-word instruction) task performance (success rate, SR) results.** Instructions are rewritten by replacing words in the original sentence with simple lexical alternatives, while largely preserving the overall task semantics. **RAS**: Residual Affordance Steering; **MCSI**: Monte Carlo Syntactic Integration.

Model	Libero-Spatial SR (%) ↑	Libero-Object SR (%) ↑	Libero-Goal SR (%) ↑	Libero-Long SR (%) ↑	Average SR (%) ↑
π_0 (Black et al., 2024)	88.6	97.6	91.4	87.6	91.30
$\pi_0 + \text{RAS}$	91.2	98.4	90.0	80.0	89.90
$\pi_0 + \text{MCSI}$	96.0	99.8	92.6	85.4	93.45
$\pi_0 + \text{RAS \& MCSI}$	97.2	99.0	88.4	80.8	91.35
$\pi_{0.5}$ (Intelligence et al., 2025)	95.2	99.0	95.0	92.6	95.45
$\pi_{0.5} + \text{RAS}$	98.4	98.6	97.0	93.2	96.80
$\pi_{0.5} + \text{MCSI}$	99.8	99.8	97.0	95.4	98.00
$\pi_{0.5} + \text{RAS \& MCSI}$	97.6	99.0	97.6	95.8	97.50

Table 9: **LIBERO (random-lang instruction) task performance (success rate, SR) results.** Instructions are constructed by randomly shuffling the word order of the original sentence, disrupting syntactic structure while retaining the same lexical content. **RAS**: Residual Affordance Steering; **MCSI**: Monte Carlo Syntactic Integration.

Model	Libero-Spatial SR (%) ↑	Libero-Object SR (%) ↑	Libero-Goal SR (%) ↑	Libero-Long SR (%) ↑	Average SR (%) ↑
π_0 (Black et al., 2024)	83.0	97.4	91.6	85.4	89.35
$\pi_0 + \text{RAS}$	91.6	94.6	89.8	78.8	88.70
$\pi_0 + \text{MCSI}$	94.2	99.2	97.0	81.4	92.95
$\pi_0 + \text{RAS \& MCSI}$	96.8	99.8	95.4	87.8	94.95
$\pi_{0.5}$ (Intelligence et al., 2025)	96.4	98.8	94.8	90.8	95.20
$\pi_{0.5} + \text{RAS}$	97.4	99.0	93.2	94.2	95.95
$\pi_{0.5} + \text{MCSI}$	98.2	100.0	95.6	96.0	97.45
$\pi_{0.5} + \text{RAS \& MCSI}$	98.6	99.8	97.0	94.4	97.45

the wine bottle or align it with the cabinet-top affordance, resulting in unstable grasps or premature terminations.

Applying **MCSI** improves robustness to syntactic variation by aggregating multiple rewritten instructions; however, when critical semantic content is missing, **MCSI** alone is insufficient to recover the correct action intent. In contrast, **RAS** explicitly injects residual affordance signals derived from visual observations, steering the policy toward task-relevant object-target configurations even when language supervision is severely degraded.

Notably, the combination of **RAS+MCSI** achieves the most reliable performance. By jointly mitigating linguistic uncertainty and reinforcing visual affordance alignment, the policy consistently executes the correct placement behavior across all evaluated rollouts. This result highlights that robustness to destructive language perturbations requires not only syntactic integration but also explicit affordance-level correction.

E Instruction Rewriting Examples on Obfuscated Instruction Reinterpretation

As illustrated in Figures 8, 9, 10, and 11, we employ ChatGPT-5.2 (OpenAI, 2025) to automatically rewrite the original task instructions into a set of semantically equivalent but linguistically obfuscated variants. These examples demonstrate how different rewriting strategies introduce increasing levels of surface variation, contextual distraction, and semantic indirection while strictly preserving the underlying task goal. Specifically, R1 and R2 introduce task-irrelevant context and commonsense-based object descriptions, respectively. R3 further reformulates instructions by emphasizing implicit reasoning or final-state constraints, and R4 explicitly injects object-level distractors via negation.

Table 10: **LIBERO (random-mask instruction) task performance (success rate, SR) results.** Each word in the instruction is independently masked with a predefined probability, introducing controlled semantic degradation while preserving the original word order. **RAS**: Residual Affordance Steering; **MCSI**: Monte Carlo Syntactic Integration.

Model	Libero-Spatial SR (%) ↑	Libero-Object SR (%) ↑	Libero-Goal SR (%) ↑	Libero-Long SR (%) ↑	Average SR (%) ↑
Random Mask Rate = 0.2					
π_0 (Black et al., 2024)	53.4	90.2	74.8	72.2	72.65
$\pi_0 + \text{RAS}$	69.6	94.6	75.4	74.2	78.45
$\pi_0 + \text{MCSI}$	94.2	97.4	81.0	82.8	88.85
$\pi_0 + \text{RAS \& MCSI}$	96.4	99.8	92.6	79.4	92.05
$\pi_{0.5}$ (Intelligence et al., 2025)	95.0	97.4	89.6	88.6	92.65
$\pi_{0.5} + \text{RAS}$	96.0	98.6	88.2	92.8	93.90
$\pi_{0.5} + \text{MCSI}$	98.4	99.2	91.8	94.8	96.05
$\pi_{0.5} + \text{RAS \& MCSI}$	98.4	99.6	92.0	95.4	96.35
Random Mask Rate = 0.4					
π_0 (Black et al., 2024)	15.2	75.0	33.0	46.8	42.50
$\pi_0 + \text{RAS}$	34.2	83.8	44.8	57.4	55.05
$\pi_0 + \text{MCSI}$	87.8	93.6	51.8	77.8	77.75
$\pi_0 + \text{RAS \& MCSI}$	93.6	100.0	71.8	78.0	85.85
$\pi_{0.5}$ (Intelligence et al., 2025)	87.2	90.8	67.6	85.2	82.70
$\pi_{0.5} + \text{RAS}$	89.6	98.8	64.6	93.6	86.65
$\pi_{0.5} + \text{MCSI}$	87.6	95.4	72.0	93.8	87.20
$\pi_{0.5} + \text{RAS \& MCSI}$	97.0	100.0	75.6	95.4	92.00
Random Mask Rate = 0.6					
π_0 (Black et al., 2024)	7.4	51.0	10.0	20.2	22.15
$\pi_0 + \text{RAS}$	18.2	66.8	20.4	28.8	33.55
$\pi_0 + \text{MCSI}$	81.4	84.0	24.8	65.2	63.85
$\pi_0 + \text{RAS \& MCSI}$	92.2	97.4	50.6	71.6	77.95
$\pi_{0.5}$ (Intelligence et al., 2025)	71.8	79.2	43.4	83.8	69.55
$\pi_{0.5} + \text{RAS}$	87.4	99.4	44.0	86.4	79.30
$\pi_{0.5} + \text{MCSI}$	70.8	85.2	46.6	91.0	73.40
$\pi_{0.5} + \text{RAS \& MCSI}$	93.6	100.0	50.6	94.0	84.55
Random Mask Rate = 0.8					
π_0 (Black et al., 2024)	0.8	13.6	5.4	11.4	7.80
$\pi_0 + \text{RAS}$	9.0	36.0	8.8	18.0	17.95
$\pi_0 + \text{MCSI}$	73.4	69.2	12.8	55.8	52.80
$\pi_0 + \text{RAS \& MCSI}$	89.0	95.0	29.2	66.4	69.90
$\pi_{0.5}$ (Intelligence et al., 2025)	60.0	63.2	24.6	74.0	55.45
$\pi_{0.5} + \text{RAS}$	83.0	97.4	23.0	80.8	71.05
$\pi_{0.5} + \text{MCSI}$	50.8	69.4	27.0	84.6	57.95
$\pi_{0.5} + \text{RAS \& MCSI}$	89.0	99.6	29.8	91.6	77.50

Table 11: **Ablation on steering coefficient (SC) across different language perturbation benchmarks.** Denoising steps are fixed to 10. All values are average success rate (SR, %).

SC	Origin SR (%) ↑	Multi SR (%) ↑	Blank SR (%) ↑	Rand SR (%) ↑	M2 SR (%) ↑	M4 SR (%) ↑	M6 SR (%) ↑	M8 SR (%) ↑	Simple SR (%) ↑	Average SR (%) ↑
π_0 (Black et al., 2024) + RAS										
1.25	92.45	91.00	63.25	91.05	80.40	61.00	39.45	24.15	63.40	67.35
1.5	90.65	89.90	63.40	88.70	78.45	55.05	33.55	17.95	62.50	64.46
2.0	89.50	87.95	66.15	87.90	71.75	45.45	22.05	9.00	62.05	60.20
3.0	81.95	82.15	64.00	79.20	58.00	25.80	7.90	1.90	58.80	51.08
$\pi_{0.5}$ (Intelligence et al., 2025) + RAS										
1.25	97.40	96.80	70.00	96.55	94.70	86.85	78.30	72.20	70.10	84.77
1.5	96.65	96.80	70.50	95.95	93.90	86.65	79.30	71.05	69.10	84.43
1.75	96.90	96.50	70.00	96.10	94.85	85.90	77.75	69.20	69.75	84.11
2.0	96.70	97.00	69.45	96.25	94.00	86.00	77.40	66.90	69.15	83.65
3.0	96.45	96.90	68.80	96.60	94.75	87.60	75.35	55.65	68.55	82.29

Table 12: **Ablation on denoising steps across different *destructive instruction overwriting* benchmarks.** The base model is π_0 (Black et al., 2024). The steering coefficient is fixed to 1.5. All values are average success rate (SR, %).

Steps	Origin	Multi	Blank	Rand	M2	M4	M6	M8	Simple	Average
	SR (%) ↑									
5	93.55	92.50	66.10	91.80	80.65	56.15	34.95	18.65	65.00	66.59
10	90.65	89.90	63.40	88.80	78.45	55.05	33.55	17.95	62.50	64.46
15	90.60	89.05	63.50	89.05	75.15	53.05	32.25	16.90	62.55	63.57

Table 13: **Ablation on denoising steps across different *destructive instruction overwriting* benchmarks.** The base model is $\pi_{0.5}$ (Intelligence et al., 2025). The steering coefficient is fixed to 1.25. All values are average success rate (SR, %).

Steps	Origin	Multi	Blank	Rand	M2	M4	M6	M8	Simple	Average
	SR (%) ↑									
5	96.90	97.35	69.90	96.65	93.90	86.50	78.70	71.45	68.90	84.47
15	97.40	97.20	69.35	95.60	94.30	85.00	78.90	72.00	69.60	84.37
20	96.85	96.60	70.30	96.65	93.45	85.60	78.05	71.60	69.80	84.32

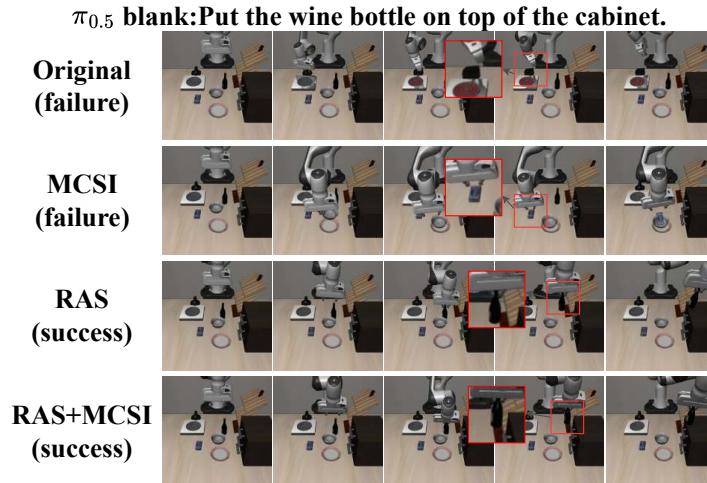


Figure 6: **Qualitative comparison under destructive instruction overwriting ($\pi_{0.5}$).** We visualize representative rollout trajectories for the task “Put the wine bottle on top of the cabinet” when the instruction is partially blanked. The base model is $\pi_{0.5}$ (Intelligence et al., 2025). **RAS:** Residual Affordance Steering; **MCSI:** Monte Carlo Syntactic Integration.

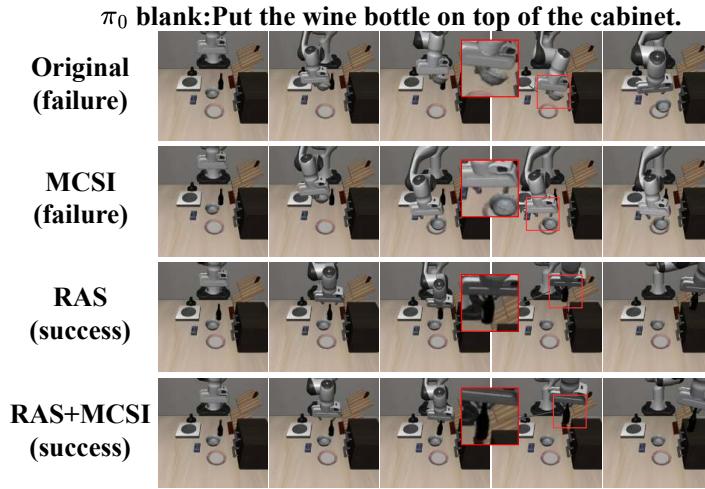


Figure 7: **Qualitative comparison under destructive instruction overwriting (π_0)**. We visualize representative rollout trajectories for the task “Put the wine bottle on top of the cabinet” when the instruction is partially blanked. The base model is $\pi_{0.5}$ (Black et al., 2024). **RAS**: Residual Affordance Steering; **MCSI**: Monte Carlo Syntactic Integration.

Please paraphrase the core instruction: "Open the middle drawer of the cabinet." into 10 different variations. Make each version longer and more human-like by adding conversational fillers, environmental context, or specific execution tips. The core task (opening that specific drawer) must not change, and the added context should not conflict with the main action.

1. When you get a second, go ahead and open the cabinet's middle drawer.
2. Before worrying about anything else on the table, open the middle drawer of the cabinet.
3. Just to get things ready, pull open the top-middle drawer in the cabinet.
4. If you're already facing the cabinet, open its middle drawer now.
5. Take a quick look at the cabinet and open the drawer in the middle.
6. Without moving any items around, simply open the cabinet's middle drawer.
7. After you orient yourself, slide the cabinet's middle drawer open.
8. To keep things organized, open the middle drawer of the cabinet first.
9. Even if it feels like a small step, open the middle drawer on the cabinet.
10. Once you're in position, open the middle drawer of the cabinet carefully.

Please paraphrase the core instruction: "Put the wine bottle on top of the cabinet." into 10 different variations. Make each version longer and more human-like by adding conversational fillers, environmental context, or specific execution tips. The core task (opening that specific drawer) must not change, and the added context should not conflict with the main action.

1. When you have a moment, place the wine bottle on top of the cabinet.
2. Even if it's not used right now, set the wine bottle on the cabinet's top surface.
3. To clear some space, put the wine bottle up on top of the cabinet.
4. Carefully lift the wine bottle and rest it on the cabinet's top.
5. Without changing anything else, move the wine bottle onto the cabinet top.
6. If you're done looking around, go ahead and place the wine bottle on top of the cabinet.
7. For a tidier setup, put the wine bottle on the top of the cabinet.
8. Take it slow—set the wine bottle on the cabinet's upper surface.
9. Just as a small organizing step, place the wine bottle on top of the cabinet.
10. Once you've got a steady hold, put the wine bottle on the cabinet top.

Figure 8: **R1-Distraction**. The instruction is augmented with task-irrelevant conversational or contextual content, such as background descriptions or auxiliary remarks, while keeping the core action and target unchanged.

16


Please paraphrase the core instruction: "**Put the bowl on the stove.**" into 10 different variations. Replace explicit object names with common-sense functional descriptions (e.g., changing "plate" to "a flat container for food") to challenge the model's information extraction and filtering. The core action and intended target must remain strictly unchanged.



1. Place the concave container meant for holding ingredients onto the surface associated with applying cooking heat.
2. Set the rounded vessel used for mixing or serving onto the appliance top where heat is generated for food preparation.
3. Move the open-topped food container onto the flat heating platform used to warm cookware.
4. Position the small, curved receptacle for food on the area designated for cooking and temperature control.
5. Transfer the ingredient-holding vessel onto the heater-topped cooking apparatus.
6. Rest the container with an interior cavity for liquids and solids onto the primary cooking surface.
7. Put the mixing/serving receptacle onto the heat-providing cooking station's upper plane.
8. Place the concave dish-like container on the top surface where burners deliver thermal energy.
9. Set the food-holding vessel on the cooking appliance's heating region.
10. Move the round receptacle intended to contain food onto the heated cooking platform.


Please paraphrase the core instruction: "**Open the top drawer and put the bowl inside.**" into 10 different variations. Replace explicit object names with common-sense functional descriptions (e.g., changing "plate" to "a flat container for food") to challenge the model's information extraction and filtering. The core action and intended target must remain strictly unchanged.



1. Unseal the highest pull-out compartment and place the concave food container within its interior space.
2. Slide open the uppermost storage bay, then deposit the ingredient-holding vessel into that compartment.
3. Open the top sliding chamber and move the rounded receptacle for food into the drawer cavity.
4. Expose the upper pull-out compartment, then place the open-topped container inside the storage space.
5. Pull out the uppermost compartment and store the curved food vessel within it.
6. Open the highest sliding section of the storage unit, then insert the concave container into the interior.
7. Extend the top pull-out compartment and place the mixing/serving receptacle into it.
8. Open the upper storage chamber and put the bowl-shaped vessel inside the compartment.
9. Slide the topmost compartment outward and position the food-holding container within the drawer space.
10. Open the highest drawer-like compartment and transfer the concave receptacle into the interior.

Figure 9: **R2-Common Sense.** Object names are replaced with commonsense-based descriptive phrases that implicitly convey their functional or physical properties. Although the task intent remains unchanged, this variant requires the model to extract relevant semantics from more abstract and verbose descriptions.



Please paraphrase the core instruction: "**Put the bowl on top of the cabinet.**" into 10 different variations. Incorporate multi-step reasoning or state constraints by either describing the desired final outcome (focusing on the result state rather than the action) or adding lightweight logical steps such as sequencing (first/then), conditions (if/then), or verifications. The core task must not change, and no new physical tasks should be introduced.



1. Locate the cabinet's top surface, then lift and place the bowl onto that top area.
2. Ensure the bowl is stable, then set it down on top of the cabinet.
3. Pick up the bowl, move it above the cabinet, and lower it onto the cabinet top.
4. Goal-state: the bowl should end up resting on the cabinet top surface.
5. Align the bowl with the cabinet top, then place it down gently to avoid sliding.
6. If the bowl is elsewhere, transfer it to the cabinet top and confirm placement.
7. Put the bowl on the cabinet top and verify it is not on the table afterward.
8. Move the bowl to the highest surface of the cabinet, then release it once steady.
9. Identify the cabinet, then place the bowl on its topmost surface.
10. Bring the bowl to the cabinet top and make sure the bowl remains on that surface.



Please paraphrase the core instruction: "**Push the plate to the front of the stove.**" into 10 different variations. Incorporate multi-step reasoning or state constraints by either describing the desired final outcome (focusing on the result state rather than the action) or adding lightweight logical steps such as sequencing (first/then), conditions (if/then), or verifications. The core task must not change, and no new physical tasks should be introduced.



1. Identify the stove's front edge, then push the plate until it reaches that front position.
2. Ensure the plate stays on the stove surface while you push it forward to the front.
3. Push the plate forward in a straight line until it is clearly at the front of the stove.
4. Goal-state: the plate should end up at the stove's front—push it until that condition is met.
5. Align your push direction toward the stove's front, then move the plate forward without tipping.
6. If the plate is not at the front, nudge it forward and confirm its final position is front-of-stove.
7. Push the plate toward the front edge, stopping once it's closest to you on the stove.
8. Move the plate forward; verify it is nearer the front than before.
9. Push the plate and check that it ends up positioned at the stove's front area.
10. First locate the plate on the stove, then push it forward until it's at the front.

Figure 10: **R3-Reasoning Chain.** The instruction is reformulated to emphasize implicit reasoning, execution order, or final-state constraints, either by introducing lightweight reasoning cues or by abstracting intermediate steps. The target task remains identical, but the linguistic form encourages reasoning-based interpretation.



Please paraphrase the core instruction: "**Put the cream cheese in the bowl.**" into 10 different variations. Introduce distractor objects (e.g., wine bottle, stove, plate) using exclusion logic such as "ignore the X," "not X but Y," or "regardless of X." The core task (opening that specific drawer) must not change, and no interaction with the distractor objects should occur.



1. Ignore the wine bottle and put the cream cheese in the bowl.
2. Not on the plate—place the cream cheese into the bowl.
3. Even if the stove is in front, put the cream cheese inside the bowl.
4. Don't turn on the stove yet; first put the cream cheese in the bowl.
5. Regardless of the drawers, move the cream cheese into the bowl.
6. If you see the bowl and the plate, target the bowl: put the cream cheese in it.
7. Not onto the cabinet top—place the cream cheese into the bowl.
8. With the rack as a distraction, put the cream cheese inside the bowl.
9. Even if the bowl later goes elsewhere, right now put the cream cheese in the bowl.
10. Ignore the stove controls and place the cream cheese into the bowl.



Please paraphrase the core instruction: "**Turn on the stove.**" into 10 different variations. Introduce distractor objects (e.g., wine bottle, stove, plate) using exclusion logic such as "ignore the X," "not X but Y," or "regardless of X." The core task (opening that specific drawer) must not change, and no interaction with the distractor objects should occur.



1. Ignore the bowl and wine bottle, and turn on the stove.
2. Regardless of what's on the plate, turn on the stove.
3. Don't open any drawers right now—turn on the stove.
4. Even if the rack is visible, switch the stove on.
5. Not placing objects first: simply turn on the stove.
6. Whether or not cream cheese is in the bowl, turn on the stove.
7. Ignore the cabinet top; activate the stove.
8. With the wine bottle as a distraction, turn on the stove.
9. After checking the burner area, turn on the stove.
10. If you're deciding between moving items and controls, choose the control: turn on the stove.

Figure 11: **R4-Confusion.** The instruction explicitly introduces distractor objects or actions through negation or contrast, while still specifying the correct target object and goal. This variant probes the model's ability to resist object-level confusion and focus on task-relevant semantics.