

VLA Grounder: Language-Conditioning Space Optimization for Black-Box VLA Models

Damir Shodiev², Nikita Kachaev^{1,4}, Alexey K. Kovalev^{1,2}✉, Aleksandr I. Panov^{1,2},
Alekssei Staroverov^{1,2,3},

¹AXXX, ²MIRAI, ³MISIS, ⁴ITMO University AI Talent Hub,

✉ Correspondence: kovalev.a@miriai.org

Abstract

Vision-Language-Action (VLA) models are commonly treated as end-to-end action policies conditioned on natural-language task descriptions. In practice, however, their behavior often depends sharply on how the instruction is phrased, suggesting that language is not merely a task label but an optimizable conditioning input. We study whether frozen VLA policies can be improved by optimizing language space rather than updating action weights. Our method introduces a language-conditioning space policy that translates a human instruction into a short VLA-grounded command using object appearance, spatial relations, and target-grounding cues. The language-conditioning space policy is optimized with reinforcement learning from sparse task-completion rewards, while the downstream VLA remains fully frozen. Experiments on RL4VLA and VL-Think show that language-conditioning space optimization improves success on instruction-sensitive, symbolic, and multi-object manipulation tasks, demonstrating that language can serve as an optimizable variable for robot foundation models.

Website: <https://cognitiveaisystems.github.io/VLA-Grounder/>

1 Introduction

Vision-Language-Action (VLA) models represent a promising paradigm in robotic learning, combining visual perception, natural language understanding, and action generation within a single policy. Recent VLA models leverage large-scale pre-training and robotic demonstrations to produce actions directly from images and language, enabling broad zero-shot manipulation across tasks and embodiments (Black et al., 2024; Zhang et al., 2025; Zhong et al., 2025).

Despite this progress, the language input to a VLA is often treated as a passive task description, even though it directly shapes the action

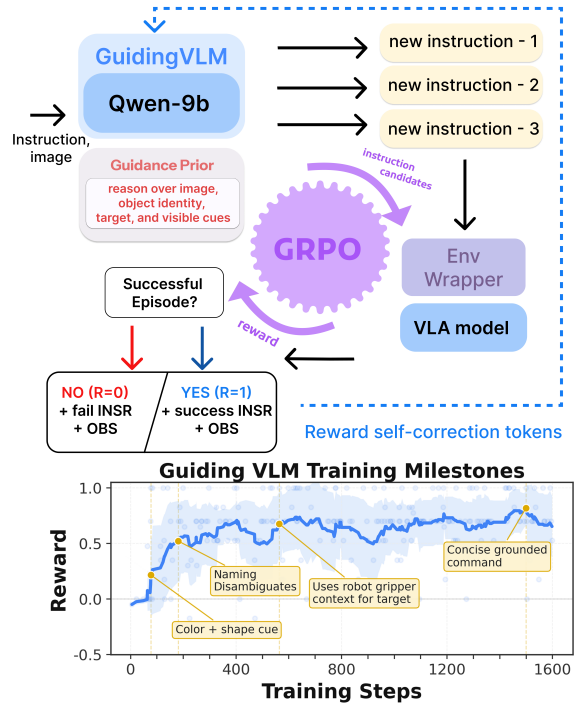


Figure 1: Visual overview of language-conditioning space optimization. A failure-derived command-space prior conditions a pre-trained VLM to translate the user instruction and scene image into a short VLA-grounded command. This command conditions a VLA policy, improving action success without updating VLA weights.

distribution of the policy. In practice, VLA behavior is highly sensitive to how a goal is expressed (Moskalenko et al., 2026; Kachaev et al., 2025; Fang et al., 2025). A command such as "put bread on plate" may fail even when the scene contains a visually clear target, while a more grounded command such as "pick up the brown round object and put it on the yellow plate" can succeed. Similar effects appear when an object name is visually misleading: "put champagne glass on plate" can be improved by referring to the same object as "the tall white glass". These examples suggest that many VLA failures arise from an instruction-

to-grounding mismatch between human semantic intent and the perceptual categories available to the robot policy.

This mismatch is amplified by the limited linguistic diversity of robot demonstration datasets (Walke et al., 2023; O’Neill et al., 2024). Most datasets contain only short, templated instructions and everyday household objects, while user requests may include ambiguous objects, abstract references, or uncommon categories. As a result, the language channel of a VLA is underused: models reserve substantial capacity for text, but the instruction often fails to provide the visual and spatial cues needed for robust action selection (Zhang et al., 2025; Liu et al., 2026; Black et al., 2024; Staroverov et al., 2025).

A natural response is to modify the instruction rather than the VLA weights. We therefore propose to insert an additional language-conditioning space policy between the human instruction and the frozen VLA, and ask whether command selection can provide an effective adaptation mechanism for large robot foundation models without updating their action weights.

We instantiate this idea with a language-conditioning space policy placed upstream of a frozen VLA, as shown in Figure 1. Given the current image and the original human instruction, the language-conditioning space policy produces a short VLA-grounded command that exposes execution-relevant cues: object appearance, color, shape, spatial position, relative location, and simplified target descriptions. The generated command is then used as the language input to the unchanged VLA policy. Because the downstream action model is kept fixed, the method adapts the conditioning signal rather than the action policy, making it non-invasive for different frozen VLA models.

We train the language-conditioning space policy (VLA Grounder) with Reinforcement Learning (RL) from sparse task-completion rewards while keeping the downstream VLA fully frozen (Figure 1). This casts adaptation as *language-conditioning space optimization*: the learned decision variable is not a continuous robot action or a VLA weight update, but the VLA-grounded command through which the frozen action model receives and grounds the task. To make this optimization meaningful, we use a failure-derived command-space prior that restricts exploration to concise, visually grounded, and VLA-compatible commands. Within this structured command space,

RL provides an action-grounded signal for discovering which commands most reliably elicit successful behavior from the existing policy. Experiments on VL-Think (Kachaev et al., 2025) and RL4VLA (Liu et al., 2026) benchmarks show that our method creates better grounded guidance commands for frozen π_0 and OpenVLA models, increasing downstream task success across object grounding and multi-object manipulation settings.

The contributions of this work are as follows:

1. **Language-conditioning space optimization for VLA models:** We formulate VLA adaptation as learning a language-conditioning space policy that maps human instructions to VLA-grounded commands while keeping the downstream action policy fixed.
2. **RL in language-conditioning space:** We introduce VLA Grounder, language-conditioning space policy that maps an image and human instruction to a concise VLA-grounded command. We optimize only the language-conditioning space policy from sparse task-completion rewards, showing that downstream robot behavior can improve without updating VLA action weights.
3. **Empirical and diagnostic evidence:** We evaluate on VL-Think and RL4VLA with frozen π_0 and OpenVLA backbones. The results show improved success on symbolic grounding, multi-object manipulation, and fixed-object settings, and our analyses indicate that optimized commands provide more grounded conditioning signals for the downstream VLA.

2 Related Work

2.1 Language-Conditioned VLA Policies

Large-scale robot policies increasingly separate high-level semantic control from low-level motor generation. ACT introduced action chunking for manipulation (Zhao et al., 2023), RT-2 transferred web-scale vision-language pretraining into robot action generation (Brohan et al., 2023), and recent generalist policies such as OpenVLA and π_0 scale this recipe to broad zero-shot manipulation (Kim et al., 2024; Black et al., 2024). Embodied chain-of-thought and $\pi_{0.5}$ further push toward richer intermediate reasoning and semantic control over action generation (Zawalski et al., 2024; Intelligence et al., 2025b). Another line of work makes a VLA more

controllable by enriching its conditioning signal. Higher-level semantic prediction in $\pi_{0.5}$, semantic guidance methods for VLA robustness, and prompt optimization methods in vision-language settings all support the broader idea that behavior depends strongly on how context is represented (Intelligence et al., 2025b; Zhan et al., 2026; Lee et al., 2023). These approaches treat language, metadata, or intermediate semantic variables as mechanisms for conditioning behavior.

Our paper asks a complementary post-training question: given an already frozen VLA, can we learn a better VLA-grounded command for control?

2.2 RL Adaptation of VLA Models

Reinforcement learning has become a central tool for improving VLA policies after imitation learning. RL4VLA studies PPO (Schulman et al., 2017)-, GRPO (Shao et al., 2024)-, and preference-based fine-tuning for VLA generalization (Liu et al., 2026); π_{RL} extends online RL to flow-based VLA models (Chen et al., 2025); and $\pi_{0.6}^*$ (Intelligence et al., 2025a) shows how deployment and corrective data can substantially improve a VLA through continued learning. More broadly, reward optimization for language policies, including ArCher, PReWrite, and StablePrompt, shows that sparse reward can shape text-producing policies in useful ways (Zhou et al., 2024; Kong et al., 2024; Kwon et al., 2024).

In contrast, our method adapts the language policy itself, not the VLA model weights, enabling parameter-efficient, non-invasive language-side adaptation.

2.3 VLA Language Robustness

Recent work shows that language remains an underdeveloped modality in embodied AI. Dataset audits find highly repetitive and template-like instructions in robot corpora (Wanna et al., 2026). Stable Language Guidance frames related failures as modality imbalance, where strong visual priors overpower sparse linguistic signals (Zhan et al., 2026), adversarial paraphrases and irrelevant context can significantly degrade VLA behavior even when task intent is preserved (Moskalenko et al., 2026; Zinkovich et al., 2026), and mechanistic analyses suggest that language-conditioned behavior is present internally but not always reliably exposed by the input instruction (Häon et al., 2025). We build on this diagnosis but reframe it as an opti-

mization problem over command space. Instead of treating language brittleness as only a robustness issue, we treat language conditioning as an actionable input space that can be optimized for downstream execution success.

3 Problem Formulation

3.1 Partially Observable Control with Language Instructions

We consider an episodic partially observable Markov decision process (POMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P, Z, r, \rho_0, \gamma)$, where $\mathcal{S}$ is the latent state space, $\mathcal{A}$ the primitive action space, $\mathcal{O}$ the observation space, P the transition kernel, Z the observation kernel, r the reward function, ρ_0 the initial-state distribution, and $\gamma \in [0, 1)$ the discount factor. At time t , the environment occupies state s_t , emits $o_t \sim Z(\cdot | s_t)$, receives action a_t , produces reward $r(s_t, a_t)$, and transitions to $s_{t+1} \sim P(\cdot | s_t, a_t)$. Each episode is paired with a fixed human instruction $u \in \mathcal{U}_H$. We write the discounted return of a trajectory τ as $R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t$.

3.2 Frozen Vision-Language-Action Policies

A pretrained Vision-Language-Action policy maps an observation and a language command to an action distribution, $\pi_{\text{VLA}} : \mathcal{O} \times \mathcal{U} \rightarrow \Delta(\mathcal{A})$, where $\mathcal{U}$ is the space of commands consumed by the action policy. In standard deployment, the raw instruction is passed directly to the policy, so $a_t \sim \pi_{\text{VLA}}(\cdot | o_t, u)$. More generally, each command $\tilde{u} \in \mathcal{U}$ induces a frozen low-level policy $\pi_{\text{VLA}}^{\tilde{u}}(a | o) := \pi_{\text{VLA}}(a | o, \tilde{u})$. Distinct commands that preserve the same task intent may still induce different grounded behavior.

Let $\mathcal{U}(u) \subseteq \mathcal{U}$ denote the command space associated with instruction u , i.e., the set of commands that are semantically consistent with u but may differ in wording, specificity, or grounding cues. The goal is not to learn a new low-level action policy, but to exploit this command space to obtain better behavior from a frozen one.

3.3 Problem Setting

We are given a pretrained frozen VLA policy π_{VLA} and seek to maximize return in $\mathcal{M}$ by selecting a command $\tilde{u} \in \mathcal{U}(u)$ rather than by changing action-model weights. Equivalently, the learned component chooses which member of the frozen family $\{\pi_{\text{VLA}}^{\tilde{u}} : \tilde{u} \in \mathcal{U}(u)\}$ will generate actions during the rollout. This formulation is useful only when

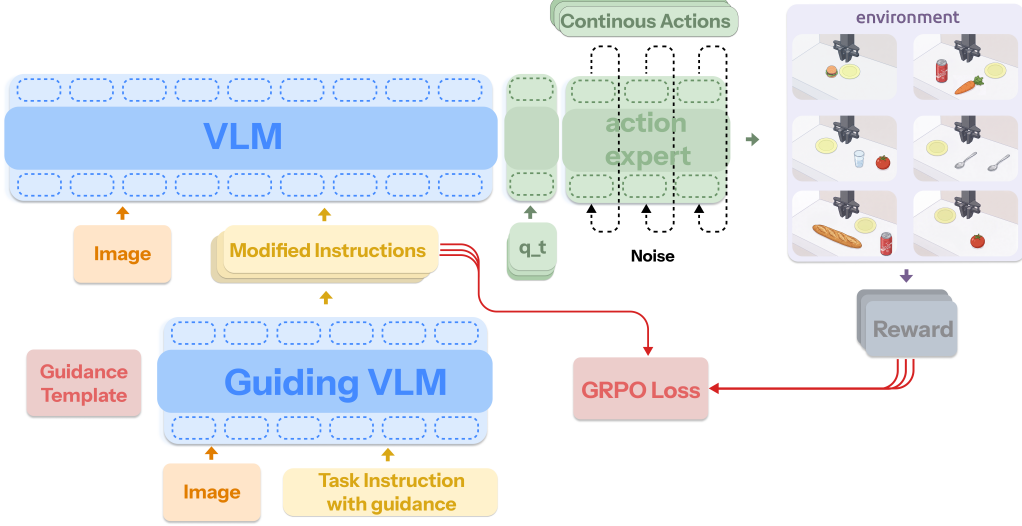


Figure 2: Architecture and training. A language-conditioning space policy maps the task instruction, scene image, and a failure-derived command-space prior to a VLA-grounded command, which is executed by a frozen VLA. Only this policy is trained with RL from sparse task-completion rewards; VLA weights remain frozen.

the policy is sensitive to language-space variation, meaning that semantically valid commands can induce meaningfully different behavior. We assume only black-box access: for any observation o and command $\tilde{u}$, we may sample from $\pi_{\text{VLA}}(\cdot | o, \tilde{u})$, but we do not use weights, gradients, or internal activations.

4 Method

4.1 RL in Language-Conditioning space

In standard deployment, a VLA acts under the raw instruction u . We instead treat the command itself as the high-level decision variable: at observation o , we first select a command $\tilde{u} \in \mathcal{U}(u)$ and then sample the primitive action from $\pi_{\text{VLA}}(\cdot | o, \tilde{u})$. This induces an action-space transformation from primitive actions to language commands.

$$\begin{aligned}
 P^{\text{lang}}(s' | s, \tilde{u}) &:= \mathbb{E}_{\substack{o \sim Z(\cdot | s) \\ a \sim \pi_{\text{VLA}}^{\tilde{u}}(\cdot | o)}} [P(s' | s, a)], \\
 r^{\text{lang}}(s, \tilde{u}) &:= \mathbb{E}_{\substack{o \sim Z(\cdot | s) \\ a \sim \pi_{\text{VLA}}^{\tilde{u}}(\cdot | o)}} [r(s, a)].
 \end{aligned} \tag{1}$$

These induced kernels define a transformed control problem, denoted $\mathcal{M}_u^{\text{lang}}$, whose action space is $\mathcal{U}(u)$ and whose dynamics and rewards are given by P^{lang} and r^{lang} . This is the sense in which the adaptation problem can be cast as reinforcement learning in language space. We focus on

rollout-level optimization: a policy $q_\phi(\tilde{u} | o_0, u)$ is invoked once at the beginning of the episode, outputs a single command, and keeps it fixed for the full rollout. Sampling $\tilde{u} \sim q_\phi(\cdot | o_0, u)$ therefore selects one member of the frozen family $\{\pi_{\text{VLA}}^{\tilde{u}} : \tilde{u} \in \mathcal{U}(u)\}$.

$$\begin{aligned}
 J(\phi) &= \mathbb{E}_{(o_0, u) \sim \mathcal{D}} \mathbb{E}_{\tilde{u} \sim q_\phi(\cdot | o_0, u)} \\
 &\quad \mathbb{E}_{\tau \sim p(\cdot | o_0, \pi_{\text{VLA}}^{\tilde{u}})} [R(\tau)].
 \end{aligned} \tag{2}$$

where $\mathcal{D}$ is the distribution of initial image-instruction pairs. The raw frozen-VLA baseline is recovered by the degenerate policy that always returns the original instruction.

4.2 Language Aliasing and Structured Exploration

Language-Conditioning space has additional structure that naive RL does not exploit. In particular, there may exist distinct commands $\tilde{u} \neq \tilde{u}'$ such that $\pi_{\text{VLA}}(\cdot | o, \tilde{u}) \approx \pi_{\text{VLA}}(\cdot | o, \tilde{u}')$ for many observations o ; we refer to this as *language aliasing*. It is common for shallow rewrites, where grammatical edits or near-synonymous substitutions are lexically different but behaviorally redundant. Our method therefore biases exploration toward short executable commands that vary along grounding-relevant dimensions such as source-object description, target cues, color, shape, and spatial location.

4.3 Optimizing the Guiding VLM Policy

We instantiate q_ϕ with a Guiding VLM and train it with GRPO (Shao et al., 2024) using only scalar rollout reward. Figure 2 summarizes the language-conditioning space optimization loop: the policy receives the initial image and instruction, proposes candidate commands, and the frozen VLA policy executes a rollout under each one. For each training pair (o_i, u_i) , we sample a group of candidate commands $\{\tilde{u}_i^{(k)}\}_{k=1}^G$ from $q_\phi(\cdot \mid o_i, u_i)$, execute the frozen VLA under each command, and compare the resulting returns $\{R_i^{(k)}\}_{k=1}^G$ within the group. We use the normalized advantage

$$\hat{A}_i^{(k)} = \frac{R_i^{(k)} - \bar{R}_i}{\sigma_i + \varepsilon}, \quad \bar{R}_i = \frac{1}{G} \sum_{k=1}^G R_i^{(k)}, \quad (3)$$

where σ_i is the empirical standard deviation of rewards in the group. Commands with larger relative advantage are up-weighted by the next update, while weaker ones are down-weighted. Appendix E summarizes the procedure.

4.4 VLA Black-Box Deployment

At inference time, the system receives a new image and instruction, samples one command $\hat{u}$ from the trained language-conditioning space policy, and conditions the frozen VLA policy on that command for the full rollout. There is no reward query, no RL update, and no action-model modification at test time. VLA Grounder changes only the language-conditioning input and requires only black-box forward access to the downstream action policy. In this paper, we evaluate this setup on frozen π_0 and OpenVLA backbones.

4.5 Why Language Exploration Matters

Group-based RL algorithms, such as GRPO, derive a signal by comparing multiple samples for the same task instance. If the sampled set differs only in grammar, then the reward cannot meaningfully separate better and worse conditioning signals. By contrast, if samples vary in robot-relevant grounding cues, for example, color, left-right position, object description, or target relation—the optimization process can assign credit to language transformations that actually change downstream behavior.

5 Experiments

We evaluate on two benchmark families. VL-Think (Kachaev et al., 2025) isolates visual-

language reasoning under fixed manipulation: the robot places the same source object on a target board matching an abstract concept such as an arrow, sign, shape or weather icon. RL4VLA (Liu et al., 2026) tests broader semantic generalization; we use *MultiPlate*, *MultiCarrot*, and fixed-object *Pepper*, which require distractor handling and object-target grounding. We also evaluate frozen MolmoAct2 (Fang et al., 2026) on a physical SO-101. All reported VLA backbones are frozen, and we report success rate as mean $\pm$ standard deviation.

The primary baseline is the frozen VLA conditioned on the raw human instruction. We also compare against the structured language-conditioning space policy before RL (“w/o GRPO”), generic command rewriting without structured reasoning (“no reasoning”), GRPO without structured guidance (“GRPO w/o guidance”), and structured-guidance variants with different language-conditioning space policy sizes. To evaluate cross-backbone transfer, we additionally swap trained Grounder checkpoints between π_0 and OpenVLA and evaluate them with the other frozen VLA without retraining. For Pepper and VL-Think, we include TextGrad (Yuksekgonul et al., 2024) and GEPA (Agrawal et al., 2026), two general-purpose prompt-optimization baselines that rewrite or mutate commands in language space. We organize the empirical section around main research questions:

RQ1: Does language-conditioning space optimization improve frozen VLA policies across backbones? Our experiments on VL-Think and RL4VLA (Table 1, Table 2, Figure 4) reveal a consistent effect: changing only the language input can substantially improve frozen action policies. The effect appears for both π_0 and OpenVLA, which suggests that the method is not exploiting a single backbone-specific quirk. The gains are most visible on VL-Think, where success depends heavily on converting abstract or symbolic targets into visually grounded commands. RL4VLA shows a smaller but still consistent trend, which is expected because these tasks add multi-object distractors and execution errors that cannot be removed by language alone.

RQ2: Does GRPO improve over the non-RL language-conditioning space policy and generic prompt-optimization baselines? The non-RL language-conditioning space policy already pro-

Task	π_0		OpenVLA			
	orig	Qwen3.5-9B	orig	Qwen3.5-9B	TextGrad	GEPA
Arrow	6.2 $\pm$ 1.3	36.5 $\pm$ 3.2	13.5 $\pm$ 3.2	62.5 $\pm$ 1.3	44.0 $\pm$ 2.5	51.3 $\pm$ 2.8
Color	45.3 $\pm$ 7.1	44.8 $\pm$ 7.0	55.7 $\pm$ 8.7	82.8 $\pm$ 2.6	61.2 $\pm$ 3.6	73.2 $\pm$ 1.2
Laundry	16.1 $\pm$ 0.7	26.6 $\pm$ 2.6	15.6 $\pm$ 2.6	37.5 $\pm$ 3.8	19.1 $\pm$ 3.6	47.0 $\pm$ 2.1
Public Info	14.6 $\pm$ 1.5	43.2 $\pm$ 2.9	21.4 $\pm$ 3.7	65.6 $\pm$ 3.4	25.6 $\pm$ 1.6	48.5 $\pm$ 3.1
Shape	20.3 $\pm$ 5.8	49.5 $\pm$ 6.0	27.6 $\pm$ 1.5	74.5 $\pm$ 6.4	54.2 $\pm$ 3.3	59.5 $\pm$ 4.1
Traffic	6.2 $\pm$ 1.3	37.0 $\pm$ 10.2	16.7 $\pm$ 2.7	63.0 $\pm$ 4.8	35.8 $\pm$ 3.4	49.4 $\pm$ 5.1
Weather	19.3 $\pm$ 2.7	42.2 $\pm$ 3.4	20.3 $\pm$ 4.6	61.5 $\pm$ 3.9	30.4 $\pm$ 4.1	57.2 $\pm$ 3.4
Avg.	18.3	40.0	24.4	63.9	38.6	55.2

Table 1: Main results on VL-Think. All VLA backbones are frozen. The language-conditioning space policy uses the strongest structured reasoning configuration, Qwen3.5-9B.

vides a useful command prior, but it does not reliably identify which command aliases the frozen action policy will execute. GRPO adds exactly this missing selection pressure: candidate commands are compared by downstream task reward rather than by linguistic plausibility. This distinction is clearest on VL-Think, where the trained policy separates sharply from its non-RL version. On RL4VLA the improvement is more modest, but remains positive despite higher action-side noise.

RQ3: How does reward-trained language optimization compare with traditional prompt optimization? We also tested generic prompt-optimization methods, including TextGrad and GEPA, using the same frozen-VLA setting. Our results in Table 1 show that these methods can improve over the raw instruction baseline, but not nearly as much as GRPO-based RL adaptation. The reason is qualitative as well as quantitative: generic prompt optimizers often modify punctuation, wording, or writing style in ways that are hard to interpret as better robot grounding. VLA Grounder instead updates the language-conditioning space policy from rollout reward, so it favors command changes that actually alter execution success rather than commands that only look cleaner as text.

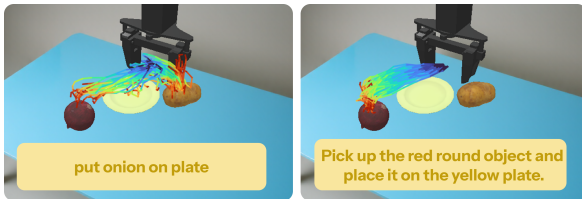


Figure 3: Trajectories comparison on RL4VLA MultiObject with frozen π_0 . The default command underspecifies the object and causes unstable motion, while the rewritten command adds visible grounding cues and yields a more direct successful trajectory.

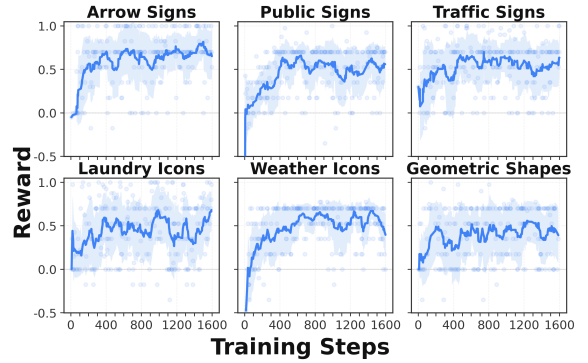


Figure 4: GRPO training for VL-Think tasks. Each panel reports rollout success rate over training steps for a symbolic grounding category. The upward trends indicate that reward updates to the language-conditioning space policy improve command selection while the downstream VLA action policy remains frozen.

RQ4: Do better commands lead to better VLA representations? We further probe whether improved commands merely change the final text input or also change the internal representations used by the VLA. Using VL-Think Public Info task, we compare OpenVLA representations under the original task instruction and under the rewritten command. The probing results Table 4, Figure 3 indicate that rewritten commands produce more informative prompt-conditioned representations: task-relevant distinctions become easier to recover from the VLA hidden states, and this aligns with the downstream success improvements. This suggests that language-conditioning space optimization changes the conditioning representation seen by the action model, rather than only changing a superficial instruction string.

RQ5: What command transformations does the learned policy discover? Based on prompt evolution analysis during training (Table 3) we found

Task	π_0		OpenVLA	
	orig	Qwen3.5-9B	orig	Qwen3.5-9B
MultiPlate	16.7 ± 7.0	20.3 ± 2.6	54.7 ± 2.2	64.6 ± 5.2
MultiCarrot	24.5 ± 0.7	33.3 ± 4.8	59.4 ± 5.9	63.5 ± 5.3
Avg.	20.6	26.8	57.0	64.0

Table 2: Results on the RL4VLA semantic generalization tasks. The same structured language-conditioning space policy improves both frozen π_0 and frozen OpenVLA in multi-object and distractor-heavy settings.

Failure type	Raw instruction	Useful rewrite dimension	Language policy rewrite
Abstract target	“sunrise icon”	visible symbol/color	“white card with yellow sun”
Rare object name	“champagne glass”	shape/material	“tall white glass”
Multi-object ambiguity	“plate”	spatial relation	“left yellow plate”
Weak object grounding	“bread”	color/shape	“brown round object”

Table 3: Common instruction-grounding failures and useful command rewrite dimensions. The learned policy improves frozen VLA behavior by replacing abstract target names, rare object categories, ambiguous references, or weak object names with visually grounded descriptions that expose color, shape, material, symbol, or spatial cues.

Prompt	Accuracy
Default	66.6
VLA Grounder	89.3

Table 4: Linear probing accuracy on VL-Think Public Info tasks using OpenVLA hidden states under the default instruction and the rewritten command.

Audit statistic	Value
Same source object and placement target	98.89% (6,345/6,416)
Mentions color	84.96%
Mentions shape	18.87%
Mentions both color and shape	17.44%
Preserves the original object name	62.98%

Task	π_0 orig	π_0 + TextGrad	π_0 + GEPA	π_0 + Qwen3.5
Pepper	39.1 ± 6.6	44.3 ± 5.5	61.7 ± 4.1	89.6 ± 3.2

Table 5: Fixed-object RL4VLA result. On Pepper, the structured language-conditioning space policy yields a large gain with the π_0 action policy frozen.

that the learned policy tends to produce shorter and more executable commands rather than more verbose descriptions. On VL-Think, it often converts symbolic target labels into compact spatial or visual aliases, such as replacing an icon name with a reference to a white card at a particular position. On RL4VLA and Pepper, it usually keeps familiar receptacle names but rewrites brittle source-object names into visible color and shape descriptions. This pattern indicates that the policy is learning how to reduce the semantic burden placed on the frozen action model. The appendix provides Figure 6 prompt-format examples.

RQ6: Do rewrites preserve task semantics?

Structured guidance defines the intended command space but is not a hard semantic constraint. We therefore audited all 6,416 saved MultiCarrot and

Table 6: Automatic scene-aware semantic audit of generated commands.

MultiPlate commands across the train and test splits and both VLA backbones. For each command, we re-rendered the corresponding scene and used the base Qwen3.5-9B model to determine whether the rewrite retained the same source object and placement target.

As shown in Table 6, 98.89% of the commands (6,345/6,416) preserve both the source object and placement target. Color is mentioned in 84.96% of the rewrites, shape in 18.87%, and both color and shape in 17.44%. The original object name is preserved in 62.98% of the commands. Thus, only 1.11% of the rewrites are judged to change the intended source or target, while most add visible grounding cues without replacing the task.

The lowest semantic-faithfulness result is 97.76% on the MultiPlate test split. The observed failures include incorrect visual descriptions, confusion between watering cans and cups, and references to incorrect non-plate targets. Because the command generator and the automatic evaluator both belong to the Qwen3.5-9B model family,

we report this result specifically as a large-scale scene-aware automatic audit rather than as human annotation.

RQ7: Does a simulation-trained Grounder transfer to a different VLA on a physical robot?

Without additional adaptation, a Grounder trained with frozen π_0 in MultiCarrot simulation improves frozen MolmoAct2 on a physical SO-101 from 26/50 to 38/50 target-reaching successes. This transfer across simulator, backbone, and embodiment supports reusable grounding cues. [Appendix C](#) gives the complete protocol and per-task results.

6 Discussion

6.1 Command-Space Design, Reward Optimization, and Transfer

The ablations suggest that the command space and the language model play different roles. Detailed per-task results for all ablations are reported in ([Appendix B](#)). Generic rewriting can already help, which confirms that the raw instruction is often a weak conditioning signal. Structured guidance improves the search space by pushing candidates toward object identity, visual attributes, and spatial grounding rather than shallow paraphrases. As shown in ([Table 8](#)), the matched ablations further separate the effects of structured guidance and reward optimization. The “w/o GRPO” configuration measures the benefit of the guidance prior alone, whereas “GRPO w/o guidance” tests reward-based optimization over a generic rewriting space. GRPO provides the larger individual improvement, while the full configuration achieves the strongest average performance, showing that structured exploration and reward-based selection are complementary. The model-scale results in ([Table 7](#)) further show that scaling the language model becomes more useful when the model has a richer space of possible aliases from which to choose. The benefit is not perfectly monotonic across every category, however, so scale alone does not explain the improvement; the strongest setting combines structured command variation with reward-based selection. The checkpoint-swap experiment ([Table 9](#)) additionally shows that the learned Grounders are not restricted to the VLA backbone used during training: transferring checkpoints between π_0 and OpenVLA improves both frozen policies without retraining. This result indicates that the optimized commands capture reusable visual-language ground-

ing cues in addition to backbone-specific linguistic preferences.

6.2 Limits of Language-Conditioning Space Optimization

The gains are largest when the user instruction is semantically valid but visually under-specified for the action policy. Symbolic VL-Think tasks such as arrows, public-information signs, weather icons, shapes, and traffic signs benefit because the learned policy can replace the symbolic target with a simpler visual or spatial description. Direct perceptual categories such as Color leave less room for improvement because the raw instruction already exposes a usable cue. The method is also limited on tasks where language is not the only bottleneck: clutter, distractors, wrong object grounding, severe visual misperception, missing low-level manipulation skill, or long-horizon reasoning demands can still dominate the rollout outcome.

6.3 Broader Implication

The broader implication of this work is that language can function as an optimizable adaptation layer for robot foundation models. If this view is correct, then future post-training of VLAs need not operate only in weight space or action space; it can also operate in the space of semantic conditioning signals that mediate between human intent and embodied action.

7 Conclusion

We study adaptation of frozen VLA policies through the language-conditioning input rather than through action-weight updates. The resulting perspective treats language as an optimizable conditioning variable and casts improvement as reinforcement learning in language space. VLA Grounder combines a language-conditioning space policy, a failure-derived command-space prior, and a sparse downstream task reward to discover VLA-grounded commands that better expose latent capability in frozen action policies. The central takeaway is simple but consequential: for large robot foundation models, policy improvement can be lifted from action space into language space.

Limitations

Our results show that language-conditioning space optimization can improve frozen VLA policies, but the scope of the evidence is still limited. The method also cannot repair all failures of a frozen action policy. Language-space optimization should be viewed as an adaptation layer over existing VLA capabilities rather than as a replacement for improving perception, robustness, control, or long-horizon reasoning.

Ethical Considerations

This work studies language-conditioned robot control in benchmark settings and does not involve human-subject data.

Artifact Licenses

We use publicly available benchmark and model artifacts, including VL-Think, RL4VLA, OpenVLA, INT-ACT, π_0 , and Qwen, under their respective licenses and terms of use. Any released code or checkpoints will be distributed with an explicit license. Our use of these artifacts is limited to research evaluation.

References

- Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziemis, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, and 1 others. 2026. Gpa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pages 8479–8565.
- Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, and 1 others. 2024. π_0 : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*.
- Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, and 1 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*.
- Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, and 1 others. 2025. π_{RL} : Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*.
- Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, and 1 others. 2026. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*.
- Irving Fang, Juexiao Zhang, Shengbang Tong, and Chen Feng. 2025. From intention to execution: Probing the generalization boundaries of vision-language-action models. *arXiv preprint arXiv:2506.09930*.
- Bear Häon, Kaylene Stocking, Ian Chuang, and Claire Tomlin. 2025. Mechanistic interpretability for steering vision-language-action models. *arXiv preprint arXiv:2509.00328*.
- Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, and 1 others. 2025a. $\pi_{0.6}^*$: A vla that learns from experience. *arXiv preprint arXiv:2511.14759*.
- Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, and 1 others. 2025b. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*.
- Nikita Kachaev, Mikhail Kolosov, Daniil Zelezetsky, Alexey K Kovalev, and Aleksandr I Panov. 2025. Don’t blind your vla: Aligning visual representations for ood generalization. *arXiv preprint arXiv:2510.25616*.
- Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sankeeti, and 1 others. 2024. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.
- Weize Kong, Spurthi Hombaiah, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Prewrite: Prompt rewriting with reinforcement learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 594–601.
- Minchan Kwon, Gaeun Kim, Jongsuk Kim, Haeil Lee, and Junmo Kim. 2024. Stableprompt: Automatic prompt tuning using reinforcement learning for large language model. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 9868–9884.
- Dongjun Lee, Seokwon Song, Jihee Suh, Joonmyeong Choi, Sanghyeok Lee, and Hyunwoo J Kim. 2023. Read-only prompt optimization for vision-language few-shot learning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1401–1411. IEEE.
- Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. 2026. What can rl bring to vla generalization? an empirical study. *Advances in Neural Information Processing Systems*, 38:97121–97151.

- Andrey Moskalenko, Daria Pugacheva, Denis Shepelev, Andrey Kuznetsov, Vlad Shakhuro, and Elena Tutubalina. 2026. Bring the apple, not the sofa: Impact of irrelevant context in embodied ai commands on vla models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 840–860.
- Abby O’Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandekar, 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.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Aleksei Staroverov, Tatiana Zemskova, Dmitry Yudin, and Aleksandr Panov. 2025. Semantic object navigation with segmenting decision transformer. *IEEE Access*.
- Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, and 1 others. 2023. Bridgedata v2: A dataset for robot learning at scale. In *Conference on robot learning*, pages 1723–1736. PMLR.
- Selma Liliane Wanna, Agnes Luhtaru, Jonathan Salfity, Ryan Barron, Juston Moore, Cynthia Matuszek, and Mitch Pryor. 2026. Limited linguistic diversity in embodied ai datasets. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11058–11086.
- Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*.
- Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. 2024. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*.
- Zhihao Zhan, Yuhao Chen, Jiaying Zhou, Qinhan Lyu, Hao Liu, Keze Wang, Liang Lin, and Guangrun Wang. 2026. Stable language guidance for vision-language-action models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4137–4159.
- Y. Zhang, Y. Qi, and X. Zheng. 2025. Experiences from benchmarking vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2511.11298*.
- Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. 2023. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*.
- Yifan Zhong, Fengshuo Bai, Shaofei Cai, Xuchuan Huang, Zhang Chen, Xiaowei Zhang, Yuanfei Wang, Shaoyang Guo, Tianrui Guan, Ka Nam Lui, and 1 others. 2025. A survey on vision-language-action models: An action tokenization perspective. *arXiv preprint arXiv:2507.01925*.
- Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. 2024. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*.
- Viktoriia Zinkovich, Anton Antonov, Andrei Spiridonov, Denis Shepelev, Andrey Moskalenko, Daria Pugacheva, Elena Tutubalina, Andrey Kuznetsov, and Vlad Shakhuro. 2026. Sparta: Evaluating reasoning segmentation robustness through black-box adversarial paraphrasing in text autoencoder latent space. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2754–2775.

A Appendix Overview

This appendix provides additional experimental results and implementation details supporting the main paper. [Appendix B](#) reports the model-scale, structured-guidance, GRPO, and cross-backbone-transfer ablations. [Appendix C](#) presents the real-robot evaluation with a frozen MolmoAct2 policy on an SO-101 arm. [Appendix D](#) illustrates the prompt format, output contract, and model-matched prompt-optimization comparisons. [Appendix E](#) summarizes the GRPO training procedure and black-box optimization loop. Finally, [Appendix F](#) reports the language-policy configuration, optimization settings, rollout configuration, and computational cost of the reported training runs.

B Ablations

Task	π_0	+Qwen3-4B no reasoning	+Qwen3-4B reasoning	+Qwen3.5-4B reasoning	+Qwen3.5-9B reasoning
Arrow	6.2 $\pm$ 1.3	11.5 $\pm$ 7.1	18.7 $\pm$ 2.9	17.3 $\pm$ 3.0	36.5 $\pm$ 3.2
Color	45.3 $\pm$ 7.1	28.8 $\pm$ 6.8	33.5 $\pm$ 6.5	35.0 $\pm$ 6.0	44.8 $\pm$ 7.0
Laundry	16.1 $\pm$ 0.7	21.4 $\pm$ 2.5	24.1 $\pm$ 3.8	25.7 $\pm$ 2.3	26.6 $\pm$ 2.6
Public Info	14.6 $\pm$ 1.5	22.3 $\pm$ 2.8	20.9 $\pm$ 3.1	26.1 $\pm$ 3.5	43.2 $\pm$ 2.9
Shape	20.3 $\pm$ 5.8	24.7 $\pm$ 4.6	33.4 $\pm$ 5.8	36.5 $\pm$ 5.0	49.5 $\pm$ 6.0
Traffic	6.2 $\pm$ 1.3	23.5 $\pm$ 3.2	18.0 $\pm$ 4.4	21.1 $\pm$ 3.8	37.0 $\pm$ 10.2
Weather	19.3 $\pm$ 2.7	25.4 $\pm$ 3.9	21.2 $\pm$ 3.8	23.9 $\pm$ 4.1	42.2 $\pm$ 3.4
Avg.	18.3	22.5	24.3	26.5	40.0

Table 7: Ablation on frozen π_0 over VL-Think benchmark. “No reasoning” denotes a generic command-rewriting policy; “reasoning” denotes the structured guidance used by our method.

Task	π_0				OpenVLA			
	orig	Guidance w/o GRPO	GRPO w/o Guidance	GRPO + Guidance	orig	Guidance w/o GRPO	GRPO w/o Guidance	GRPO + Guidance
<i>VL-Think</i>								
Arrow	6.2 $\pm$ 1.3	7.3 $\pm$ 2.7	37.5 $\pm$ 6.4	36.5 $\pm$ 3.2	13.5 $\pm$ 3.2	21.4 $\pm$ 2.0	34.9 $\pm$ 6.0	62.5 $\pm$ 1.3
Color	45.3 $\pm$ 7.1	35.9 $\pm$ 2.2	49.5 $\pm$ 4.1	44.8 $\pm$ 7.0	55.7 $\pm$ 8.7	63.5 $\pm$ 4.5	75.5 $\pm$ 3.2	82.8 $\pm$ 2.6
Laundry	16.1 $\pm$ 0.7	12.5 $\pm$ 1.3	31.2 $\pm$ 2.2	26.6 $\pm$ 2.6	15.6 $\pm$ 2.6	21.9 $\pm$ 5.6	28.7 $\pm$ 4.8	37.5 $\pm$ 3.8
Public Info	14.6 $\pm$ 1.5	20.3 $\pm$ 2.2	34.4 $\pm$ 2.2	43.2 $\pm$ 2.9	21.4 $\pm$ 3.7	27.1 $\pm$ 3.2	40.6 $\pm$ 3.4	65.6 $\pm$ 3.4
Shape	20.3 $\pm$ 5.8	34.9 $\pm$ 4.5	37.0 $\pm$ 6.4	49.5 $\pm$ 6.0	27.6 $\pm$ 1.5	49.0 $\pm$ 7.7	68.2 $\pm$ 3.2	74.5 $\pm$ 6.4
Traffic	6.2 $\pm$ 1.3	14.1 $\pm$ 4.6	35.9 $\pm$ 8.9	37.0 $\pm$ 10.2	16.7 $\pm$ 2.7	28.1 $\pm$ 4.6	42.2 $\pm$ 7.8	63.0 $\pm$ 4.8
Weather	19.3 $\pm$ 2.7	20.8 $\pm$ 3.7	35.4 $\pm$ 5.9	42.2 $\pm$ 3.4	20.3 $\pm$ 4.6	31.3 $\pm$ 3.4	41.2 $\pm$ 5.3	61.5 $\pm$ 3.9
Avg.	18.3	20.8	37.3	40.0	24.4	34.6	47.3	63.9
<i>RL4VLA</i>								
MultiPlate	16.7 $\pm$ 7.0	14.6 $\pm$ 6.0	19.3 $\pm$ 6.0	20.3 $\pm$ 2.6	54.7 $\pm$ 2.2	56.3 $\pm$ 4.6	62.5 $\pm$ 2.1	64.6 $\pm$ 5.2
MultiCarrot	24.5 $\pm$ 0.7	30.2 $\pm$ 0.7	29.2 $\pm$ 2.7	33.3 $\pm$ 4.8	59.4 $\pm$ 5.9	60.9 $\pm$ 2.6	61.5 $\pm$ 4.8	63.5 $\pm$ 5.3
Avg.	20.6	22.4	24.2	26.8	57.0	58.6	62.0	64.0

Table 8: Matched ablation of structured guidance and GRPO. All experiments use Qwen3.5-9B as the language-conditioning policy, while the downstream VLA remains frozen.

Task	π_0			OpenVLA		
	orig	transferred	matched	orig	transferred	matched
MultiPlate	16.7 $\pm$ 7.0	21.9 $\pm$ 3.8	20.3 $\pm$ 2.6	54.7 $\pm$ 2.2	60.4 $\pm$ 2.7	64.6 $\pm$ 5.2
MultiCarrot	24.5 $\pm$ 0.7	37.5 $\pm$ 4.6	33.3 $\pm$ 4.8	59.4 $\pm$ 5.9	63.5 $\pm$ 4.1	63.5 $\pm$ 5.3
Avg.	20.6	29.7	26.8	57.0	62.0	64.0

Table 9: Cross-backbone transfer on RL4VLA. “Transferred” evaluates a Grounder trained using rewards from the other VLA backbone, without retraining. “Matched” evaluates a Grounder trained using rewards from the same VLA backbone used for evaluation. All downstream VLA models remain frozen.

C Real-Robot Evaluation

We evaluate VLA Grounder on a physical SO-101 arm controlled by frozen MolmoAct2, using overhead and wrist cameras. At the start of each trial, the gripper faces three objects. We vary the object combination across trials, always include the requested object, and randomize its position. The evaluation pool comprises a toy green cucumber; blue, yellow, red, and green cubes; a white charger; green and red markers; a toy brown baguette; green and brown toy blocks and cylinders; pink, green, and yellow flowers; and black tape. We test five object-selection instructions with ten trials per method and instruction. A trial succeeds when the gripper reaches the designated region above the intended object; accordingly, this experiment measures target-reaching and object-selection success rather than completion of a full manipulation sequence.

The Grounder checkpoint was trained only in MultiCarrot simulation using rewards from frozen π_0 . We apply it to MolmoAct2 without real-world training, MolmoAct2-specific adaptation, manual command editing, or test-time candidate selection. As shown in Table 10, VLA Grounder improves four of the five tasks, preserves performance on tape, and increases aggregate success from 26/50 to 38/50.

D Prompt Format Example

Figure 6 illustrates the prompt format used to transform a human instruction into a VLA-grounded command. The input consists of the current scene image, the original task instruction, and a structured instruction that asks the language model to identify the source object, target, and visible grounding cues. The policy is allowed to use this reasoning to choose the command, but the downstream VLA receives only the final text inside the `<answer>` field. This separation prevents long explanations from entering the action model while preserving the benefit of scene-aware language reasoning.

D.1 Prompt-Optimization Comparisons

To isolate the optimization method from the optimizer-model choice, we reran TextGrad and GEPA with Qwen3.5-9B, the same language model used by generic GRPO and VLA Grounder. Table 11 reports the OpenVLA averages from these controlled reruns together with the original GPT-4o configurations. Model matching does not reverse the overall conclusion: VLA Grounder remains strongest, outperforming model-matched TextGrad, GEPA, and generic GRPO. Generic GRPO is competitive with model-matched GEPA, while adding structured guidance produces the largest gain. Because these algorithms have different optimization and interaction patterns, we describe the comparison as optimizer-model matched rather than fully budget matched.

E Training Algorithm and Optimization Loop

The training procedure treats command generation as the only learned component. For each image-instruction pair, the language-conditioning space policy samples a group of candidate commands. Each candidate is executed by the same frozen VLA policy, and the resulting sparse rollout rewards are compared within the group. GRPO then updates only the language policy, leaving the action model unchanged.

Algorithm 1: VLA Grounder Training

Input: training distribution $\mathcal{D}$ over image-instruction pairs (o, u) , frozen VLA policy π_{VLA} , language-conditioning space policy q_ϕ , GRPO group size G .

For each training instance:

1. Sample a GRPO group of commands $\{\tilde{u}^{(k)}\}_{k=1}^G$ from $q_\phi(\cdot \mid o, u)$.
2. For each $\tilde{u}^{(k)}$, run a rollout with the same frozen VLA policy, keep the command fixed for the rollout, and collect scalar return $R^{(k)}$.
3. Compute normalized within-group advantages from $\{R^{(k)}\}_{k=1}^G$.
4. Apply a GRPO update to q_ϕ using the grouped commands and advantages; keep π_{VLA} unchanged.

Output: trained language-conditioning space policy q_ϕ .

Human instruction	VLA Grounder command	MolmoAct2	+ VLA Grounder
“pick up the cucumber”	“pick up the green cucumber”	6/10	9/10
“pick up the charger”	“pick up the white cylinder”	4/10	8/10
“pick up the toy baguette”	“pick up the brown round object”	5/10	7/10
“pick up the tape”	“pick up the black circle”	8/10	8/10
“pick up the toy building block”	“pick up the brown cylinder”	3/10	6/10
Overall		26/50	38/50

Table 10: Real-robot target-reaching results. Each entry is the number of successful trials out of ten. The Grounder was trained with frozen π_0 in MultiCarrot simulation and transferred to frozen MolmoAct2 on a physical SO-101 without additional training or adaptation.



Figure 5: Physical SO-101 evaluation with frozen MolmoAct2. The figure show the camera setup and some selected samples for tasks.

Full Prompt and Example Output

Exact prompt prior, runtime scene-conditioned input, and concrete model response. The language policy may reason in `<think>`, but only `<answer>` is sent to the frozen VLA policy.

fixed prior

role: system

fixed across tasks

You are a language translation assistant for robot manipulation tasks. You translate human instructions into commands a Vision-Language-Action (VLA) robot can act on.

You will be given:

- A user-provided instruction describing a manipulation goal and an image of the scene.

Your task is to:

1. Understand the meaning of the original instruction.
2. Correctly identify the object to pick and the target location to place it, then reason about how the robot visually perceives each — these may not be the same as your own understanding.
3. Generate a rephrased command that helps the robot locate and act on the target.

Keep in mind: the VLA perceives the scene visually and may not recognize objects the same way you do. Your translation must bridge that perceptual gap.

Feel free to explore.

image + original task

role: user

current image and original instruction

[image: current scene]

Task: {txt}.

Generate a command for the robot to successfully complete the task.

When describing the object or the target, always reason from what the robot likely sees in the image, not from your own understanding of the object. The rules below are all applications of this.

Rules for the command:

1. Always keep the object name (e.g. 'carrot'), unless the object is unrecognizable to the robot — in that case replace the name with a visual description per Rule 7.
2. One short sentence. No line breaks, no numbering.
3. Try different grounding cues to help locate the object or the target — for example: spatial words (left, right, center, far, near), visual properties (color, size, shape), or relative position (closest, next to the cup). Try combining cues (e.g. 'far right', 'small red one on the left'). Feel free to totally change the name of the target if it may resemble more simple, real thing.
4. Use only simple, everyday words.
5. If the object to pick or the placement target is something the robot is unlikely to recognize by name (abstract concept, brand, celebrity, symbol), describe it purely by its visual appearance in the scene — what the robot would actually see it as.

Think inside `<think>` tags following these steps:

1. Identify the object to pick and the placement target in the scene.
2. Reason about how the robot likely perceives each.
3. Decide how to rephrase the command.

Keep thinking process short, no more than 10 sentences.

Then give your final command inside `<answer>` your_command_here `</answer>` tags.

NOTE: only the command inside `<answer>` will be passed to VLA!

model output example

reasoning plus final command

MODEL OUTPUT: REASONING

`<think>`

The user wants me to translate the instruction "put carrot on sunrise icon" into a command for a robot.

1. Identify the object to pick: There is an orange object with a green top on the table. It looks like a carrot. The robot might see it as an orange carrot-shaped object or just carrot. Given Rule 1, I should keep carrot if recognizable, which it likely is.
2. Identify the placement target: There are three white cards on the table. Left card: grey squiggly lines. Center card: yellow semi-circle with rays behind horizontal lines; this looks like a sun setting or rising. Right card: cloud with blue rain drops. The target is the center card with the sun symbol. The robot sees a white square card with a yellow sun symbol.
3. Reasoning about perception: The robot sees a carrot lying on the table. The robot sees three cards. The one in the middle has a yellow sun drawing. The command needs to direct the robot to move the carrot to that specific card.
4. Drafting the command: "Put the carrot on the sun card" is a bit vague. "Pick up the orange carrot and place it on the white square with the yellow sun" is more descriptive. "Place the carrot on the center card with the sun picture" adds a spatial cue. "Put the carrot on the card with the yellow sun and black rays" is very specific.

The rule asks for grounding cues: spatial words and visual properties. "Sunrise icon" might be abstract for a robot, while "white card with the yellow sun" is concrete. The object name carrot should remain because it is recognizable. Final choice should be one short sentence with simple words.

`</think>`

`<answer>`

Place the carrot on the white card with the yellow sun symbol.

`</answer>`

Figure 6: Prompt-format example for VLA Grounder. The language policy receives the scene image and original task, identifies visual cues for the source object and target, and emits one concise VLA-grounded command. Only the command inside the `<answer>` tags is passed to the frozen VLA policy.

F Training Parameters

This section reports the training configuration used for the released VLA Grounder checkpoints; the language-policy, optimization, and computational-cost settings are summarized in Table 13, Table 14, and Table 12, respectively. All experiments optimize Qwen3.5-9B with LoRA while keeping the downstream VLA policy frozen. The π_0 and OpenVLA experiments use the same language-policy and GRPO hyperparameters but differ in their rollout configurations. Training uses two H100 GPUs: one for the frozen VLA rollout process and one for GRPO optimization.

40831

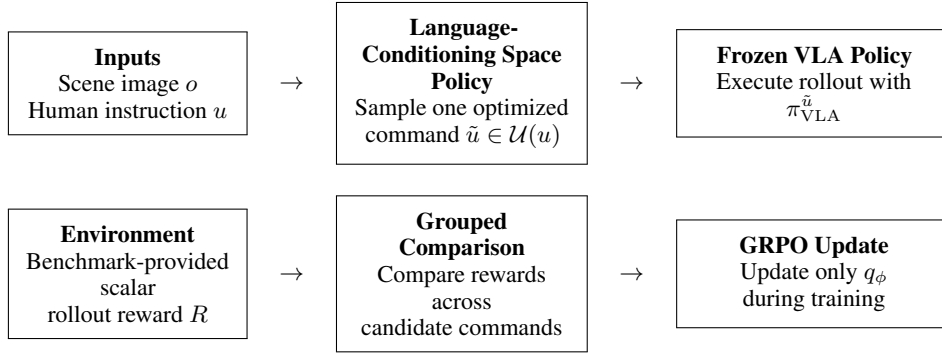


Figure 7: VLA Grounder language-conditioning space optimization loop. The image and user instruction define a context in which the language-conditioning space policy selects a command. The frozen VLA policy then executes a rollout under the command-conditioned action model $\pi_{\text{VLA}}^{\tilde{u}}$. During training, grouped reward comparisons update only the language-conditioning space policy; during inference, the same composition is used without reward queries or parameter updates.

Method	Optimizer model	OpenVLA average
TextGrad	GPT-4o	38.6
TextGrad	Qwen3.5-9B	34.7
GEPA	GPT-4o	55.2
GEPA	Qwen3.5-9B	49.2
Generic GRPO	Qwen3.5-9B	47.3
VLA Grounder	Qwen3.5-9B	63.9

Table 11: Controlled OpenVLA comparison of prompt-optimization methods. TextGrad and GEPA are reported with their original GPT-4o optimizer and with Qwen3.5-9B model-matched reruns. The downstream OpenVLA policy remains frozen.

Frozen VLA	Rollouts	Hardware	Wall time
π_0	76,800	2× H100	83 h
OpenVLA	12,800	2× H100	80 h

Table 12: Training cost of the reported language policies. Inference samples one command and executes one rollout, without candidate selection or reward queries.

Field	Value
Base language model	Qwen/Qwen3.5-9B
PEFT method	LoRA
LoRA rank r	32
LoRA scaling α	64
LoRA dropout	0.05
Train-time precision flag	bf16 = true
Target modules	12 projection modules spanning attention and MLP blocks
Tokenizer context length	model_max_length = 262144
Tokenizer padding / truncation	left padding, left truncation

Table 13: Language-conditioning space policy configuration

Field	π_0 /VL-Think	OpenVLA/VL-Think
Gradient accumulation steps	8	8
Max training steps	1600	1600
Learning rate	5×10^{-6}	5×10^{-6}
Scheduler	cosine	cosine
Stored warmup_steps field	0.05	0.05
Weight decay	0.01	0.01
Adam β_1, β_2	0.9, 0.999	0.9, 0.999
Adam ϵ	10^{-8}	10^{-8}
Max grad norm	1.0	1.0
Per-device eval batch size	8	8
Save strategy / save steps	steps / 500	steps / 500

Table 14: GRPO optimization configuration.