Mini-BEHAVIOR-Gran: Revealing U-Shaped Effects of Instruction Granularity on Language-Guided Embodied Agents

Anonymous ACL submission

Abstract

Instruction granularity is an important yet poorly controlled variable in language-guided embodied AI. Existing benchmarks typically pair each task with a single static instruction, making it difficult to study how agent behavior changes when the same task is described at different levels of detail. We introduce MINI-BEHAVIOR-GRAN, a new benchmark for controlled studies of instruction granularity that extends MINI-BEHAVIOR with multiple instruction variants per task, ranging from high-level goal descriptions to step-by-step guidance. Using this benchmark, we compare four candidate metrics for cross-task granularity quantification: token count, entity count, action-verb count, and *planning-width*, and find that width correlates most consistently with agent performance. Using width to organize training and evaluation further reveals a non-monotonic U-shaped relationship between instruction granularity and performance, with peaks at both fine and coarse extremes. Further analysis suggests that the coarse-granularity performance rebound is associated with shallow grounding, where agents learn vision-dominant policies[1].

1 Introduction

Language-guided embodied agents learn a policy $\pi(a|v, l)$ that maps visual observations v and language instructions l to executable actions a. Recent progress has largely focused on model architectures, most notably the Vision-Language-Action (VLA) paradigm (Kim et al., 2024; Black et al., 2025; Shukor et al., 2025) and its advanced variants that adapt *world models* for future frame prediction ($v_t \mapsto v_{t+1}$) (Wu et al., 2026; Zhao et al., 2025; Bi et al., 2025) or *Chain-of-Thought* (CoT) reasoning for intermediate planning (Ye et al., 2025; Lu et al., 2025; Yin et al., 2025). Yet the impact of *instruction granularity* remains underexplored.

[1]partial code and data are provided via the the Openreview platform, full version will be released upon publication.

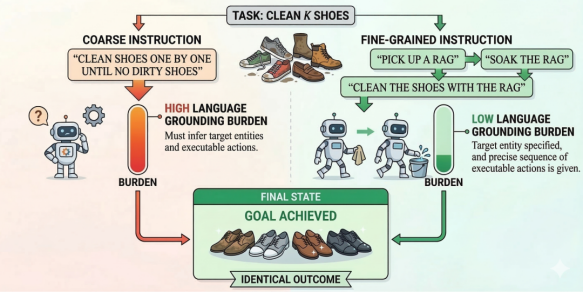

Figure 1: For the same task (cleaning k shoes), a coarse instruction requires the agent to infer target entities and executable actions. In contrast, a fine-grained instruction provides more detailed steps, significantly reducing the language grounding effort to reach the goal.

Instruction granularity refers to the level of detail provided in l to guide agent decision-making (Arumugam et al., 2017), and it is typically an uncontrolled variable in current datasets and evaluations. It matters because varying granularity fundamentally alters the cross-modal alignment difficulty and the latent planning burden imposed on the agent. Understanding this impact is crucial not only for optimizing instruction design during training, but also for ensuring model robustness across diverse human instruction styles encountered at test time.

A systematic investigation of instruction granularity has been hindered by two key bottlenecks. First, existing embodied AI benchmarks lack controlled variation in instruction granularity. Typically, each task instance is paired with a single, static instruction, making it difficult to study how an agent's performance scales as instructions shift from fine-grained (e.g., detailed step-by-step) to coarse (e.g., high-level goal descriptions). For example, popular benchmarks such as LIBERO and BEHAVIOR-1K provide only fixed or high-level instructions. Even recent suites like LIBERO-PLUS (Fei et al., 2025), which aim to increase diversity, primarily vary visual conditions (e.g., lighting, sensor noise) while keeping the language input

fixed.

Second, and perhaps more fundamentally, the field lacks a well-accepted and formalized metric for quantifying *instruction granularity*. Since granularity represents the density of guidance provided for decision-making, an ideal metric should directly reflect the complexity of grounding language into executable actions. With the advent of Large Language Models (LLMs), a seemingly natural proxy is to use the hierarchical depth of an instruction, i.e., its specific level within a decomposed plan, as a measure of granularity (Gao et al., 2025). However, such a metric fails to support meaningful cross-task comparisons because it conflates instructional detail with the task's inherent complexity. Different tasks exhibit fundamentally different internal structures: some involve interleaved subgoals that resist clean hierarchical decomposition (e.g., the Sussman anomaly; Sussman, 1975), whereas others can be arbitrarily elongated into superficial substeps. Consequently, hierarchical levels cannot establish a consistent granularity scale across diverse tasks, thereby undermining their utility for investigating the general impacts of granularity on language grounding.

To enable systematic study of how instruction granularity shapes language-guided embodied agents, we introduce MINI-BEHAVIOR-GRAN, an extension of MINI-BEHAVIOR (Jin et al., 2023) that provides multiple instruction variants per task, spanning from high-level goal description to step-by-step instructions. Leveraging Minigrid's engine (Chevalier-Boisvert et al., 2023), we construct 803 unique problem instances across 20 domains, yielding 10,253 trajectory episodes, each paired with dynamic instructions that evolve as the agent progresses through the task. Within this controlled setting, we compare four metrics for quantifying granularity: token count, entity count, action-verb count, and *width*. Unlike *decomposition depth*, *width* measures the max state features jointly tracked per instruction, thereby enabling cross-task comparability of planning complexity (Bonet et al., 2019; Drexler et al., 2024).

Our study shows that *width* correlates more consistently with agent performance across task horizons and VLA variants than other metrics, serving as a better quantitative proxy for instruction granularity. Organizing training by width reveals a U-shaped pattern. Probing this effect via instruction predictability ($P(l|v)$) suggests that the rebound at coarse end is associated with shallow grounding, where agents prioritize single-modality cues and shift toward vision-dominant policies.

Contribution. We make the following contributions: **(1)** Introduce MINI-BEHAVIOR-GRAN, a benchmark for controlled study of instruction granularity. **(2)** Demonstrate that *width* is the most consistent cross-task proxy among the metrics we evaluate. **(3)** Unveil a non-monotonic U-shaped relationship between granularity and agent performance; and **(4)** Provide evidence that the coarse-granularity performance rebound is linked to shallow grounding.

2 MINI-BEHAVIOR-GRAN: A Controlled Benchmark for Instruction Granularity

Following prior work on varying granularities and abstraction in robot instructions (Arumugam et al., 2019), we define *instruction granularity as the level of details and abstraction with which a language instruction specifies a task. In embodied settings, instruction granularity determines the latent planning complexity that an agent must resolve to map instructions into executable actions.*

Reiterating the example in Figure 1, we note that a coarse instruction (e.g., "clean the shoes") leaves the agent with a large language grounding gap, as it must infer the target entities (e.g., which shoes to clean) and the executable actions (e.g., how to clean). In contrast, a fine-grained instruction provides detailed steps that significantly reduce the language grounding effort to fulfill the same task. In this section, we describe how we construct MINI-BEHAVIOR-GRAN to systematically vary instruction granularity for the same task.

2.1 Shared Feature Space and Symbolic Layer

Many embodied AI simulators expose symbolic representations alongside raw pixel observations for task specification, progress monitoring, or success evaluation (Shridhar et al., 2020; Puig et al., 2020; Li et al., 2022a,b). We leverage this symbolic backbone to define a shared feature space Φ common across all tasks, representing each environment state s as a feature vector $\phi(s) = (\phi_1(s), \ldots, \phi_n(s))$. Features in Φ include both Boolean variables (e.g., H_r: holding the rag; S_r: the rag is soaked) and numerical variables (e.g., p_r: distance to the rag; N_f: number of uncleaned shoes). This shared feature space provides the underlying vocabulary for representing instructions,

which in turn allows us to quantify instruction granularity across tasks.

2.2 Rule-Based Instruction Representation

Building on the shared feature space Φ, we construct each instruction as a set of *"if-then" rules* that specify desired state progression. This choice is motivated by the fact that sequential decision-making tasks in automated planning are commonly modeled in terms of state variables and transitions with preconditions and effects, following the STRIPS formalism (Haslum et al., 2019). Thus, rule-based instructions explicitly encode the latent planning complexity an instruction imposes, as agents must determine how to transition from states satisfying a rule's precondition to those satisfying its effect. Concretely, an instruction in this study is a set of rules $\mathcal{R} = \{r_1, \ldots, r_m\}$, where each rule r_i is of the form $C_{r_i} \mapsto E_{r_i}$:

- C_{r_i} (the *condition*) is a conjunction of feature constraints. For a Boolean feature p, a constraint is p or $\neg p$; for a numerical feature n, it is $n = 0$ or $n > 0$.
- E_{r_i} (the *effect*) is a conjunction of feature changes. For a Boolean feature p, an effect is p, $\neg p$, or $p?$ (allowing any change); for a numerical feature n, it is $n \downarrow$ (decrease), $n \uparrow$ (increase), or $n?$ (any change).

For each task, we begin by manually constructing a finest-grained rule set $\mathcal{R}^{(0)}$ that decomposes the task into atomic steps, i.e., each rule typically can be fulfilled by a single action. We use the task of *cleaning k shoes* as a running example. Table 1 lists the relevant features from the shared space Φ used in this task.

$\mathcal{R}^{(0)}$ contains over a dozen atomic rules. For brevity, we show representative rules from each phase (full set in § G).

Representative rules from $\mathcal{R}^{(0)}$ (Cleaning Shoes).

$$\{\neg H_r,\ p_r > 0\} \mapsto \{p_r \downarrow\} \quad \text{(approach rag)}$$
$$\{H_r,\ \neg T_s,\ p_s = 0\} \mapsto \{T_s\} \quad \text{(turn on tap)}$$
$$\{H_r,\ T_s,\ p_s = 0\} \mapsto \{S_r\} \quad \text{(soak rag)}$$
$$\{S_r,\ N_f > 0,\ p_f > 0\} \mapsto \{p_f \downarrow\} \quad \text{(approach shoe)}$$
$$\{S_r,\ N_f > 0,\ p_f = 0\} \mapsto \{N_f \downarrow,\ \neg H_r\} \quad \text{(clean and drop)}$$
$$\{N_f = 0,\ \neg O_r,\ H_r\} \mapsto \{\neg H_r,\ O_r\} \quad \text{(put away rag)}$$

A coarse rule is not created by mechanically concatenating the preconditions and effects of its constituent atomic rules. Instead, we treat each merge as an abstraction of a contiguous subplan segment into a macro-rule. The macro-rule exposes only the *net input/output interface* of that segment: its condition specifies when the segment applies, and its effect captures the overall progress achieved. Features that can be consumed entirely within the segment are thus omitted from the expression. As shown in Figure 2, the $\mathcal{R}^{(2)}$ rules for picking up the rag (r_{1a}), soaking it (r_{1b}), and cleaning one shoe (r_{1c}) form a coherent subplan r_1 that abstracts into a single coarse rule in $\mathcal{R}^{(3)}$, which captures the net progress of cleaning one shoe while hiding the intermediate features H_r and S_r.

After each merge, we verify the semantic validity of the resulting coarse rules through expert review and use a planning solver to ensure every $\mathcal{R}^{(k)}$ is well-formed: sequentially reachable and goal-terminating (Bonet and Geffner, 2021).

Feature	Description
H_r	holding the rag (Boolean)
p_r	distance to the nearest rag (numerical)
p_s	distance to the sink (numerical)
p_f	distance to the nearest uncleaned shoe (numerical)
N_f	number of uncleaned shoes (numerical)
T_s	tap (sink) is turned on (Boolean)
S_r	rag is soaked (Boolean)
C_i	shoe i is cleaned (Boolean)
O_r	rag is on the floor (Boolean)

Table 1: Features used in the Cleaning Shoes task.

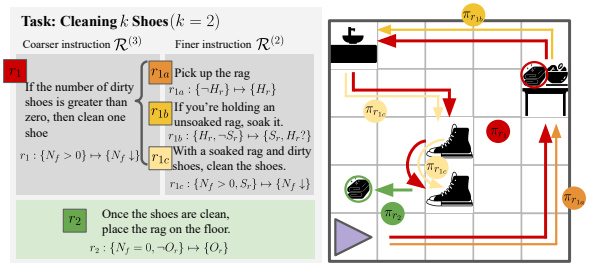

Figure 2: A running example of constructing coarse-grained instructions by merging atomic rules into macro-rules. Features H_r and S_r are omitted as they can be consumed entirely within the subplan chunk.

2.3 Dynamic Instruction Generation

To communicate these rules to the agents, we translate each rule into a natural-language sentence via a template-based generator. The templates verbalize the feature conditions and desired effects in each rule. For example, $\{H_r, \neg S_r\} \mapsto \{S_r\}$ is realized as "If you are holding the rag but the rag is not yet soaked, soak the rag."

The benchmark supports *dynamic* instruction generation that updates with task progress. As illustrated in Figure 3, during both training and

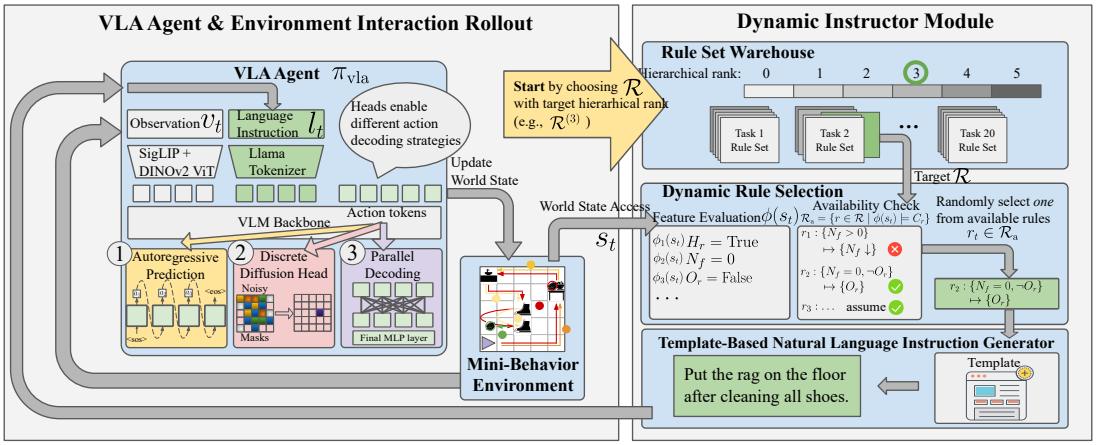

Figure 3: MINI-BEHAVIOR-GRAN Experimental Pipeline. We evaluate the impact of instruction granularity across three VLA action-decoding strategies. During inference time, a dynamic instructor evaluates the current world state to decide available rule set $\mathcal{R}_a$ and active rule r_t. A rule-based natural language instruction l_t is generated accordingly. The process iterates until completion or max steps.

Rank	N	$\overline{\text{Len}}$	U_{tok}	$\overline{\text{Plan}}$
0	51,474	40.58 ± 10.59	145	83.43 ± 53.66
1	27,837	38.19 ± 11.53	139	85.81 ± 56.91
2	14,997	33.51 ± 11.17	253	93.68 ± 57.24
3	6,816	34.09 ± 13.99	171	120.26 ± 51.39
4	1,691	41.78 ± 22.67	112	169.32 ± 37.71
5	923	33.31 ± 7.68	85	146.70 ± 23.56

Table 2: Distribution of instruction variants across granularity ranks. **N**: number of instruction instances; $\overline{\text{Len}}$: mean instruction length in words; U_{tok}: number of unique tokens; $\overline{\text{Plan}}$: mean plan length in steps.

evaluation, an instructor module monitors the simulator state at each timestep, computes the set of applicable rules $\mathcal{R}_a$, and determines which rule r_t to activate. This setup ensures that the agent always receives state-relevant guidance, in contrast to existing benchmarks (Li et al., 2022b; Fei et al., 2025) that provide only a single static instruction.

2.4 Benchmark Analysis

Table 2 summarizes the scale and coverage of MINI-BEHAVIOR-GRAN. Across 20 domains, we define a shared feature space Φ with 166 grounded features. Task-specific feature usage ranges from only 2 features in simple tasks such as *Opening Packages* to 29 in long-horizon tasks such as *Cleaning Up the Kitchen*. Despite their structured origin, the resulting instructions are of moderate linguistic complexity, averaging 40.2 words in length (see § F.3 for details).

3 Quantifying Instruction Granularity

While hierarchical rank remains a useful within-task measure, it cannot be used as a consistent cross-task metric to study how instruction granularity impacts agent performance across a diverse task set. The reason is that the number of attainable levels depends on the internal structure of each task: some tasks admit long chains of cleanly mergeable subgoals, whereas others involve interleaved dependencies that prevent further merging. Consequently, the same hierarchical rank across two different tasks may entail vastly different language grounding complexities. We therefore seek a separate cross-task metric for quantifying granularity.

3.1 Candidate Metrics

To this end, we evaluate four candidate metrics, each corresponding to a different hypothesis about which properties of an instruction plausibly reflect its granularity and associated grounding burden. *Token Count* measures *verbal elaboration*: finer-grained instructions often require more words to spell out procedural details. *Entity Count* measures *referential grounding load*, since mentioning more objects or locations requires the agent to ground a larger set of task-relevant referents. *Action-Verb Count* measures *procedural complexity*, as finer-grained instructions tend to specify more individual actions. While these three metrics capture surface-level properties, our fourth metric, *Width*, captures a more structural notion of planning burden.

3.2 Instantiating Width as a Metric

Among the candidate metrics we consider, *width* is the least surface-oriented. Originating from width-based planning (Lipovetzky and Geffner, 2012, 2017), for a rule r_i, its width $w(r_i)$ is the maximum

State Progression	Concurrent Features Tracked	Width
Initial State	$[\langle\rangle]$	0
Navigate to rag	$[\langle p_r = X\rangle, \ldots, \langle p_r = 0\rangle]$	1
Pick up rag	$[\langle p_r = 0\rangle, \langle H_r\rangle]$	1
Navigate to sink	$[\langle H_r, p_s = X\rangle, \ldots, \langle H_r, p_s = 0\rangle]$	2
Turn on tap (sink)	$[\langle H_r, p_s = 0\rangle, \langle T_s\rangle]$	2
Soak rag	$[\langle H_r, T_s, p_s = 0\rangle, \langle S_r\rangle]$	**3**
Navigate to Shoe 1	$[\langle H_r, S_r, p_{f1} \downarrow\rangle, \ldots,]$	**3**
Clean Shoe 1	$[\langle H_r, S_r, p_{f1} = 0\rangle, \langle C_1\rangle]$	3
Navigate to Shoe 2	$[\langle H_r, S_r, C_1, p_{f2} \downarrow\rangle, \ldots]$	**4**
Clean Shoe 2	$[\langle H_r, S_r, C_1, p_{f2} = 0\rangle, \langle C_1, C_2\rangle]$	4
Put rag on floor	$[\langle C_1, C_2, H_r\rangle, \ldots, \langle C_1, C_2, O_r\rangle]$	3

Table 3: State progression and width for the "Cleaning k Shoes" ($k = 2$). Width scales as $w = k + 2$.

number of features that must be simultaneously tracked along an optimal transition from a state satisfying C_{r_i} to one satisfying E_{r_i}. This differs from simply counting the number of features mentioned in the rule's condition or effect, since these features need not all be coordinated concurrently. We instantiate instruction-level width by aggregating over its constituent rules $w(\mathcal{R}) = \max_{r_i \in \mathcal{R}} w(r_i)$. A special case of $w = 0$ occurs for situations where the desired state progression can be achieved in zero or one step (Bonet and Geffner, 2024); this will appear in our experimental analysis.

Table 3 illustrates width computation for a two-shoe cleaning task. Approaching the rag ($p_r \downarrow\rightarrow p_r = 0$) requires tracking only p_r ($w = 1$). Obtaining the rag at $p_r = 0$ ($\rightarrow H_r$) also needs only p_r ($w = 1$). Then, maintaining H_r while progressing raises width to 2. Soaking the rag requires simultaneously maintaining three features: at sink ($p_s = 0$), holding rag (H_r), and tap on (T_s), yielding $w = 3$. Cleaning the first shoe (C_1) requires tracking H_r, S_r, and p_f ($w = 3$). Cleaning the second shoe adds maintaining C_1, increasing width to 4. We acknowledge that w is not a direct measure of internal grounding difficulty, since learning-based embodied agents operate as black-box decision-makers rather than explicit width-based search. However, w provides a cross-task proxy for the latent planning demands. We thus view it as a planning-inspired measure of granularity.

4 Experiments

4.1 Experimental Setup

Data. We use MINI-BEHAVIOR-GRAN, which contains 20 long-horizon household tasks defined over a shared feature space Φ. For each task, the benchmark provides 50 training and 10 evaluation instances, each paired with reference trajectories generated by the BFWS planning solver (Lipovetzky and Geffner, 2017).

Action-Decoding Archetypes. The primary architectural difference among recent VLA models lies in their action-decoding strategy (Zhong et al., 2025). We therefore adopt a unified VLM backbone and instantiate three representative action-decoding paradigms (see the left panel of Figure 3): (1) **Autoregressive** (AR), exemplified by RT-2 (Zitkovich et al., 2023), OpenVLA (Kim et al., 2024), and related models (Reed et al., 2022; Huang et al., 2024; O'Neill et al., 2024), which predict actions sequentially via causal attention; (2) **Discrete Diffusion**, which predicts action chunks through a diffusion process (Black et al., 2025; Bjorck et al., 2025; Shukor et al., 2025; Liang et al., 2025); and (3) **Parallel Decoding**, which uses bidirectional attention for single-pass action prediction (Kim et al., 2025b; Song et al., 2025; Yin et al., 2025; Wang et al., 2025; Xiao et al., 2025). Implementation details are given in § H.

Two-Stage Experimental Protocol. We conduct experiments in two stages. In **Stage I** (§ 4.2), we keep the training distribution fixed and train all models under a mixed-granularity regime that samples from all available instruction variants. This stage is used only to compare candidate granularity metrics by how consistently they correlates with agent Success Rate (SR) across tasks, task horizons, and action-decoding architectures. In **Stage II** (§ 4.3–§ 4.4), after identifying which is the most consistent metric, we use it to partition instruction variants into *Fine*, *Medium*, and *Coarse* groups and study how different granularity mixtures affect learning and generalization. Specifically, we consider seven training settings: **Pure F/M/C** (1 : 0 : 0), (2) **Centroid** (1 : 1 : 1): uniformly sample from all three classes; (3) **Axial F/M/C** (3 : 1 : 1): emphasizes one class with some exposure to the others.

4.2 RQ1: Which metric best captures instruction granularity?

We first ask which candidate metric most consistently captures instruction granularity in a way that it reflects embodied language grounding difficulty across tasks. We train all models under a balanced granularity regime and evaluate the Pearson correlation between each candidate metric and agent SR. A useful cross-task metric should exhibit a stable relationship with performance across tasks of different sequential burdens.

Why partition by instruction horizon? A critical confound in evaluating instruction granularity

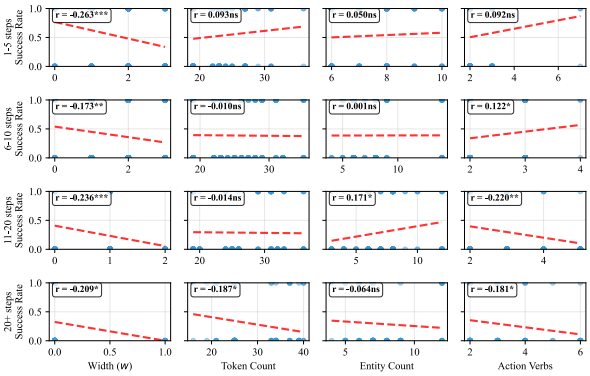

Figure 4: Metric-success correlation by task-horizon, with *width* showing consistent negative correlations.

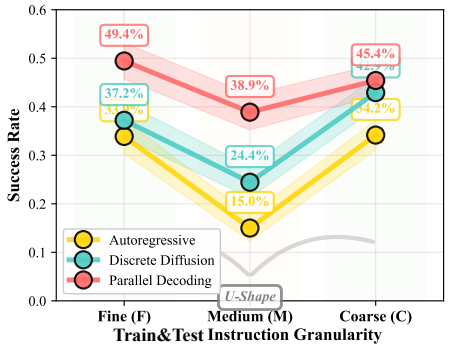

Figure 5: In-Distribution (matched train/test) SR across granularities (i.e., train Fine → test Fine, etc). A non-monotonic U-shaped trend appears.

is sequential burden. A natural choice would be to stratify examples by the total number of primitive actions in the expert trajectory. However, this is not the most relevant notion of difficulty in our setting. In MINI-BEHAVIOR-GRAN, agents do not ground a single static instruction over the entire trajectory; instead, the instructor issues a sequence of rule-conditioned instruction updates as execution unfolds. The more relevant notion of sequential burden is therefore the *instruction horizon*, defined as the number of rule calls issued during execution. We thus partition metric–SR associations evaluation instances by instruction horizon.

Width vs. surface-level metrics. We compare between token count, entity count, action-verb count, and width. Figure 4 show that width is the only metric that exhibits a statistically significant and consistently negative correlation with SR across all instruction-horizon bins (e.g, -0.263 in the 1-5 steps bin). In other words, as width increases, agent success decreases in a stable manner regardless of sequential burden. By contrast, the other three metrics are notably less consistent. Token count is insignificant in the first three bins and becomes significantly negative only in the longest-horizon bin. Entity count is mostly insignificant and even becomes weakly positive in the 11–20 bin. Action-verb count reaches significance in several bins, but its direction flips across horizons, indicating that it does not provide a stable notion of granularity across tasks. We thus use width to organize training in the remainder of the paper.

4.3 RQ2: How does agent performance vary across instruction granularity?

To enable a fair cross-task comparison of instruction granularity effects, we construct relative granularity classes via per-task quantile binning. This approach addresses two key limitations of fixed-width grouping: (i) the inherent variation in task complexity results in varying widths (e.g., *opening_packages* maxes at $w = 1$, while *cleaning_the_kitchen* reaches $w = 5$). Therefore, for each task, we collect all available instruction widths, sort them, and partition them into F/M/C classes using terciles (or an equivalent quantile split that balances the number of instances per level). When the number of distinct widths is insufficient for a clean three-way split (e.g., only two widths exist), we apply a priority-fill heuristic. This normalization ensures that the F/M/C classes represent relative granularity levels within each task, enabling meaningful cross-task comparisons of how performance varies with instruction granularity.

Figure 5 reveals a consistent non-monotonic U-shaped relationship between instruction granularity and agent performance across all three action-decoding archetypes. At the fine extreme, where instructions require minimal latent reasoning, models achieve high success rates. Performance then drops at medium granularity, before rebounding under coarse instructions. This is notable because width tracks latent coordination burden: if granularity affected performance only through increasing planning difficulty, one would expect a monotonic decline rather than a recovery at the coarse end.

4.4 RQ3: What drives the U-shaped effect?

The coarse-granularity rebound is initially counter-intuitive: if width reflects latent coordination burden, why does performance improve again when instructions become coarser? We hypothesize that this rebound stems from increasing instruction predictability $P(l|v)$, which makes coarse instructions

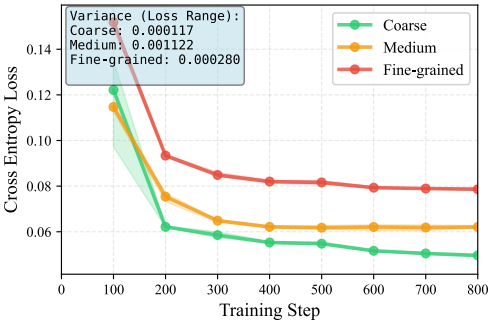

Figure 6: VLM instruction-prediction Cross Entropy (CE) loss across granularities and 3 seeds. Larger CE loss indicates lower $P(l \mid v)$.

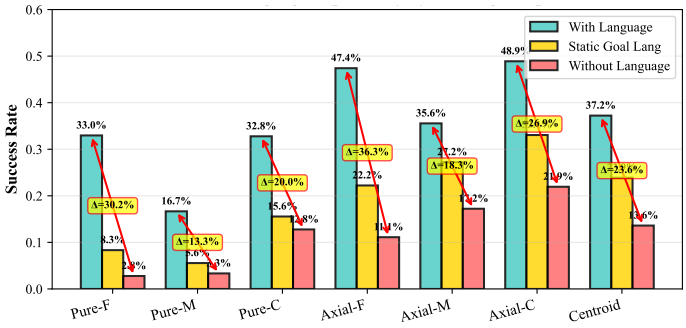

Figure 7: Smaller SR gaps indicate a shift towards shallow grounding. Under semantic perturbation, the Pure-C setting exhibits tiny variance (2.8%) between *w/ goal lang* and *w/o lang*.

Model	T1	T2	T3
AR	78.1% (25/32)**	12.5% (4/32)	65.6% (21/32)
Parallel	15.6% (5/32)	28.1% (9/32)	15.6% (5/32)
Diffusion	53.1% (17/32)	71.9% (23/32)*	75.0% (24/32)**

Table 4: Layer-wise ($N = 32$) attention trends from action tokens. **T1**: lang $\uparrow (F \to M)$; **T2**: lang $\downarrow (M \to C)$; **T3**: vis $\uparrow (M \to C)$. $^{*}p < 0.05$, $^{**}p < 0.01$ (Binomial test). Parallel models lack consistent trends, likely due to modality entanglement inherent in bidirectional attention (Kim et al., 2021). T2/T3 are not zero-sum as the it includes self-attention. Full table and stats in § I.2.

Training Setup	Width 0 (20 tasks)	Width 1 (20 tasks)	Width 2 (19 tasks)	Width 3 (10 tasks)	Width ≥ 4 (3 tasks)
Pure-F	34.9%	16.9%	10.8%	1.0%	0.0%
Pure-M	12.2%	22.0%	11.7%	0.0%	0.0%
Pure-C	26.3%	33.3%	32.9%	22.9%	0.0%
Axial-F	50.2%	53.3%	48.8%	34.3%	0.0%
Axial-M	44.7%	43.5%	45.8%	29.5%	0.0%
Axial-C	55.7%	52.2%	50.8%	37.1%	0.0%
Centroid (Mix)	42.7%	38.4%	40.8%	28.6%	0.0%

Table 5: Success rate (%) across training setups and test granularities. Numbers in parentheses indicate task coverage at each width. For Pure models, cells where test width was unseen during training measure zero-shot transfer. Transfer is asymmetric: coarse $\to$ fine succeeds.

easier to infer from visual context alone. This high predictability incentivizes the policy to exploit single-modality cues, leading to a dominant reliance on visual features rather than complex cross-modal coordination.

This increased predictability has a formal basis: coarse instructions subsume multiple finer-grained variants. Formally, consider a surjective mapping f from fine-grained labels l_i to coarser ones c_j. The conditional probability of a coarse label aggregates its fine-grained constituents: $P(c_j|v) = \sum_{l_i \in f^{-1}(c_j)} P(l_i|v)$. This aggregation makes coarse instructions statistically easier to predict from vision alone.

We confirm this empirically by training a VLM-based instructor (Qwen3-VL-4B with LoRA) to predict instructions from visual observations. As Figure 6 shows, cross-entropy loss decreases monotonically with coarseness (Fine: 0.078, Medium: 0.063, Coarse: 0.047), a relative drop of 40% from fine to coarse, confirming that $P(l|v)$ indeed increases with coarseness.

We next ask whether this increased predictability changes how much models actually rely on language. Following prior work (Fei et al., 2025), we use ΔSR when language is weakened or removed as a probe of grounding reliance. When full instructions are removed entirely, fine-trained policies suffer a severe 36.3% drop, whereas coarse-trained policies show a much smaller 20.0% gap (Figure 7). Notably, under reduced informativeness (*w/ goal lang*), the Pure-C policy exhibits only a negligible 2.8% advantage over having no language at all (15.6% vs. 12.8%). This indicates that policies trained on coarse instructions develop shallow grounding: language is not deeply integrated into decision-making.

Table 4 provides a *supplementary diagnostic*, for the shallow-grounding account. Using a layer-wise Binomial test ($H_0 = 0.5$), we find that Discrete Diffusion models exhibit a shift consistent with greater visual reliance at the coarse end (T2, $p = 0.01$; T3, $p < 0.01$). AR models show a similar but weaker pattern: a stronger language grounding phase from F to M granularity (T1: $25/32$, $p \approx 0.001$), followed by a directional shift toward visual reliance at the coarse end (T3: $21/32$). We therefore interpret these results as converging evidence rather than definitive mechanistic explanation. Note that T2 and T3 are not redundant: action tokens can also attend to previous action tokens, so a decrease in language attention does not imply a one-to-one

increase in visual attention.

Together, these results reveal a three-regime account of the U-shaped performance pattern. At **fine granularity**, Low $P(l|v)$ (high CE loss) means language provides unique information. Combined with low-width grounding burden, this drives genuine multimodal alignment for generating actions, yielding high SR. **Medium granularity** introduces a moderate increase in grounding burden. The model attempts to ground language but struggles with the increased complexity, leading to the lowest performance. At **coarse granularity**, instructions are too complex to ground effectively, and high $P(l|v)$ incentivizes agents to exploit visual cues. This visual-dominant strategy recovers SR but sacrifices language grounding.

Grounding via Mixed-Granularity Training. We hypothesize that increasing the diversity of the training language space should artificially lower average $P(l|v)$, forcing agents to re-engage with text. Our mixed-granularity training confirms this: all mixed-granularity models exhibit substantially larger ΔSR gaps than their Pure-X counterparts. Among these, the fine-weighted mixture *Axial-F* emerges as the optimal compromise, achieving strong overall performance while maintaining larger ΔSR gaps indicative of language grounding.

Asymmetric Generalization. We observe an asymmetric transfer pattern (Table 5): coarse-trained models successfully zero-shot transfer to fine-grained instructions, but fine-trained models fail on coarse-grained ones. This asymmetry aligns with our ΔSR analysis: coarse training induces a visual-dominant policy, hence refining the instruction yields minimal behavioral change. Fine training induces tight coupling with language, making the policy brittle when detailed guidance drops. Also see § I for failure mode analysis, showing low width encounters regression failures more frequently, while high width leads to stagnation.

5 Related Work

Instruction Abstraction and Granularity. Here, *instruction granularity* refers to the level of detail that affects the latent planning burden of an instruction. This differs from *linguistic abstraction*, defined as representational compression (Wing, 2010; Lachmy et al., 2022; Zheng et al., 2024). Granularity varies independently: "tidy the room" is coarse yet non-abstract (underspecified actions, no compression); "repeat wiping until clean" is specific yet abstract (prescribes action via looped condition).

While not framed as the term "granularity", we acknowledge that a rich body of work has focused on augmenting language specificity and its impact, for example through coarse-to-fine semantic parsing (Dong and Lapata, 2018; Li et al., 2020). Other work generates compositional instructions via constraint dropout and sub-task recombination (Huang et al., 2025b), or construct counterfactual instructions to augment diversity (Glossop et al., 2025).

Varying instruction detail is often handled via a simple two-part decomposition: high-level plans are converted into sequences of low-level steps before being grounded into actions (Arumugam et al., 2017; Yang et al., 2024; Shi et al., 2025). Most recent Vision Language Action (VLA) works, however, operate with static language instructions (Ma et al., 2025; Li et al., 2025). Some models like InstructVLA and ThinkAct explore having chain-of-thought reasoning or planning steps before action decoding (Huang et al., 2022; Yang et al., 2025; Huang et al., 2025a). These hierarchical definitions remain task-specific and qualitative, lacking a unified, cross-task metric to establish shared granularity levels across tasks. Consequently, quantitatively controlled studies on the formal dynamics of instruction granularity remain scarce.

Language Grounding vs. Shortcuts. Prior work highlights a dichotomy between fragile language sensitivity (Akula et al., 2022) and visuo-motor shortcuts (Xing et al., 2025; Zhang et al., 2026). We reconcile this via width (w), providing complementary insights into the trade-offs between language grounding and visual reliance based on varying instruction granularity in training distributions.

6 Conclusion

We introduce MINI-BEHAVIOR-GRAN, the first testbed for studying the impact of instruction granularity. Comparing four metrics, we find *width* most consistently correlates with grounding difficulty. Using width to organize training reveals a U-shaped granularity-performance pattern. We provide a granularity-centric diagnosis of the underlying mechanisms, showing that coarse instructions are more predictable from vision, leading to visual-dominant policies that exploit visual cues at the expense of language grounding. The results also suggest that training with a fine-weighted mixtures of granularities helps mitigate it.

7 Limitations

Discrete Action Spaces. We evaluate our framework in environments with discrete action spaces, a common abstraction in language-conditioned embodied AI (e.g., ALFWorld-based agent (Xiong et al., 2025; Kim et al., 2025a)) and LLM-based agent systems (Qin et al., 2024; Patil et al., 2024), where high-level reasoning is decoupled from low-level execution via skill primitives or API calls. This choice is intentional: it isolates symbolic planning from continuous control noise, allowing us to cleanly attribute observed effects to instruction granularity. However, extending this granularity-aware framework to hybrid discrete-continuous or continuous action spaces remains essential for real-world robotics applications, where low-level motor commands interact with high-level linguistic abstractions.

Scope of Granularity Definition. We view subgoal decomposition as the primary axis of instruction granularity, as it directly determines the latent planning burden—the central construct we aim to measure. This focus aligns with the classical planning view that hierarchical task decomposition is the fundamental mechanism for reducing search complexity. Other forms of granularity (e.g., vagueness, abstraction via loops or branching) are important but introduce additional complexities that would confound our initial investigation. By first isolating and understanding the effect of subgoal decomposition in a controlled setting, we establish a foundation for future work to incorporate these other dimensions. Our benchmark and findings provide a baseline against which such extensions can be compared.

Practicality of width computation. Computing width currently requires access to symbolic state features and optimal plans, which limits its direct applicability to natural language instructions in unstructured environments. We emphasize, however, that width is designed as an analytic metric for controlled studies, not as a deployable measure for real-world settings. Within our benchmark, width enables precise quantification of latent planning burden, allowing us to isolate the effect of instruction granularity and uncover fundamental phenomena such as the U-shaped performance curve and asymmetric transfer. Extending width estimation to realistic settings remains future work and does not affect the validity of our findings.

References

Arjun R. Akula, Spandana Gella, Aishwarya Padmakumar, Mahdi Namazifar, Mohit Bansal, Jesse Thomason, and Dilek Hakkani-Tur. 2022. ALFRED-L: investigating the role of language for action learning in interactive visual environments. In *EMNLP*, pages 9369–9378. Association for Computational Linguistics.

Dilip Arumugam, Siddharth Karamcheti, Nakul Gopalan, Edward C. Williams, Mina Rhee, Lawson L. S. Wong, and Stefanie Tellex. 2019. Grounding natural language instructions to semantic goal representations for abstraction and generalization. *Auton. Robots*, 43(2):449–468.

Dilip Arumugam, Siddharth Karamcheti, Nakul Gopalan, Lawson LS Wong, and Stefanie Tellex. 2017. Accurately and efficiently interpreting human-robot instructions of varying granularities. *arXiv preprint arXiv:1704.06616*.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. 2025. Motus: A unified latent action world model. *Preprint*, arXiv:2512.13030.

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, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, and 16 others. 2025. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR.

Blai Bonet, Guillem Francès, and Hector Geffner. 2019. Learning features and abstract actions for computing generalized plans. In *AAAI*, pages 2703–2710. AAAI Press.

Blai Bonet and Hector Geffner. 2021. General policies, representations, and planning width. In *AAAI*, pages 11764–11773. AAAI Press.

Blai Bonet and Hector Geffner. 2024. General policies, subgoal structure, and planning width. *Journal of Artificial Intelligence Research*, 80:475–516.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo De Lazcano Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and J K Terry. 2023. Minigrid & miniworld: Modular &

customizable reinforcement learning environments for goal-oriented tasks. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Dominik Drexler, Jendrik Seipp, and Hector Geffner. 2024. Expressing and exploiting subgoal structure in classical planning using sketches. *J. Artif. Intell. Res.*, 80.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. 2025. Libero-plus: In-depth robustness analysis of vision-language-action models. *Preprint*, arXiv:2510.13626.

Chongkai Gao, Zixuan Liu, Zhenghao Chi, Junshan Huang, Xin Fei, Yiwen Hou, Yuxuan Zhang, Yudi Lin, Zhirui Fang, and Lin Shao. 2025. VLA-OS: Structuring and dissecting planning representations and paradigms in vision-language-action models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Catherine Glossop, William Chen, Arjun Bhorkar, Dhruv Shah, and Sergey Levine. 2025. Cast: Counterfactual labels improve instruction following in vision-language-action models. *Preprint*, arXiv:2508.13446.

Danijar Hafner. 2021. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*.

Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, Christian Muise, Ronald Brachman, Francesca Rossi, and Peter Stone. 2019. *An introduction to the planning domain definition language*, volume 13. Springer.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. 2025a. Thinkact: Vision-language-action reasoning via reinforced visual latent planning. *Preprint*, arXiv:2507.16815.

Hui Huang, Jiaheng Liu, Yancheng He, Shilong Li, Bing Xu, Conghui Zhu, Muyun Yang, and Tiejun Zhao. 2025b. Musc: Improving complex instruction following with multi-granularity self-contrastive training. In *ACL (1)*, pages 10667–10686. Association for Computational Linguistics.

Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. 2024. An embodied generalist agent in 3d world. In *ICML*.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 1769–1782. PMLR.

Emily Jin, Jiaheng Hu, Zhuoyi Huang, Ruohan Zhang, Jiajun Wu, Li Fei-Fei, and Roberto Martín-Martín. 2023. Mini-BEHAVIOR: A procedurally generated benchmark for long-horizon decision-making in embodied AI. In *NeurIPS 2023 Workshop on Generalization in Planning*.

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. 2024. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*.

Jeonghye Kim, Sojeong Rhee, Minbeom Kim, Dohyung Kim, Sangmook Lee, Youngchul Sung, and Kyomin Jung. 2025a. ReflAct: World-grounded decision making in LLM agents via goal-state reflection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33433–33465, Suzhou, China. Association for Computational Linguistics.

Moo Jin Kim, Chelsea Finn, and Percy Liang. 2025b. 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 Paul Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2024. Openvla: An open-source vision-language-action model. In *CoRL*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR.

Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5583–5594. PMLR.

Royi Lachmy, Valentina Pyatkin, Avshalom Manevich, and Reut Tsarfaty. 2022. Draw me a flower: Processing and grounding abstraction in natural language. *Trans. Assoc. Comput. Linguistics*, 10:1341–1356.

Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish

Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. 2022a. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 455–465. PMLR.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R. Matthews, and 9 others. 2022b. BEHAVIOR-1K: A benchmark for embodied AI with 1, 000 everyday activities and realistic simulation. In *CoRL*, volume 205 of *Proceedings of Machine Learning Research*, pages 80–93. PMLR.

Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. 2025. Cogvla: Cognition-aligned vision-language-action model via instruction-driven routing & sparsification. *Advances in neural information processing systems*.

Zechang Li, Yuxuan Lai, Yansong Feng, and Dongyan Zhao. 2020. Domain adaptation for semantic parsing. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3723–3729. ijcai.org.

Zhixuan Liang, Yizhuo Li, Tianshuo Yang, Chengyue Wu, Sitong Mao, Liuao Pei, Xiaokang Yang, Jiangmiao Pang, Yao Mu, and Ping Luo. 2025. Discrete diffusion vla: Bringing discrete diffusion to action decoding in vision-language-action policies. *arXiv preprint arXiv:2508.20072*.

Nir Lipovetzky and Hector Geffner. 2012. Width and serialization of classical planning problems. In *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 540–545. IOS Press.

Nir Lipovetzky and Hector Geffner. 2017. Best-first width search: Exploration and exploitation in classical planning. In *AAAI*, pages 3590–3596. AAAI Press.

Guanxing Lu, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. 2025. Thinkbot: Embodied instruction following with thought chain reasoning. In *The Thirteenth International Conference on Learning Representations*.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. 2025. A survey on vision-language-action models for embodied ai. *Preprint*, arXiv:2405.14093.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alexander Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew E. Wang, Anikait Singh, and 260 others. 2024. Open x-embodiment: Robotic learning datasets and RT-X models : Open x-embodiment collaboration. In *ICRA*, pages 6892–6903. IEEE.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, and 7 others. 2024. Dinov2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.*, 2024.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In *NeurIPS*.

Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Joshua B. Tenenbaum, Sanja Fidler, and Antonio Torralba. 2020. Watch-and-help: A challenge for social perception and human-ai collaboration. *Preprint*, arXiv:2010.09890.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *ICLR*. OpenReview.net.

Scott E. Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. 2022. A generalist agent. *Trans. Mach. Learn. Res.*, 2022.

Lucy Xiaoyang Shi, Brian Ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. 2025. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. In *ICML*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, and 1 others. 2025. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*.

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Han Zhao, Wei Zhao, Zhide Zhong, Zongyuan Ge, Zhijun Li, Donglin Wang, Lujia Wang, Jun Ma, and Haoang Li. 2025. PD-VLA: accelerating vision-language-action model integrated with action chunking via parallel decoding. In *IROS*, pages 13162–13169. IEEE.

Gerald Jay Sussman. 1975. *A computer model of skill acquisition*. Elsevier Science Inc.

Hongyu Wang, Chuyan Xiong, Ruiping Wang, and Xilin Chen. 2025. Bitvla: 1-bit vision-language-action models for robotics manipulation. *arXiv preprint arXiv:2506.07530*.

Jeannette M. Wing. 2010. Computational thinking: What and why? *The Link*.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Shuai Zhou, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Zechen Wang, and 6 others. 2026. A pragmatic vla foundation model. *Preprint*, arXiv:2601.18692.

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

Youguang Xing, Xu Luo, Junlin Xie, Lianli Gao, Heng Tao Shen, and Jingkuan Song. 2025. Shortcut learning in generalist robot policies: The role of dataset diversity and fragmentation. In *9th Annual Conference on Robot Learning*.

Weimin Xiong, Yifan Song, Qingxiu Dong, Bingchan Zhao, Feifan Song, XWang, and Sujian Li. 2025. MPO: Boosting LLM agents with meta plan optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 3914–3935, Suzhou, China. Association for Computational Linguistics.

Ruochu Yang, Fumin Zhang, and Mengxue Hou. 2024. Oceanplan: Hierarchical planning and replanning for natural language AUV piloting in large-scale unexplored ocean environments. In *WUWNet*, pages 11:1–11:5. ACM.

Shuai Yang, Hao Li, Yilun Chen, Bin Wang, Yang Tian, Tai Wang, Hanqing Wang, Feng Zhao, Yiyi Liao, and Jiangmiao Pang. 2025. Instructvla: Vision-language-action instruction tuning from understanding to manipulation. *Preprint*, arXiv:2507.17520.

Angen Ye, Zeyu Zhang, Boyuan Wang, Xiaofeng Wang, Dapeng Zhang, and Zheng Zhu. 2025. Vla-r1: Enhancing reasoning in vision-language-action models. *Preprint*, arXiv:2510.01623.

Cheng Yin, Yankai Lin, Wang Xu, Sikyuen Tam, Xiangrui Zeng, Zhiyuan Liu, and Zhouping Yin. 2025. Deepthinkvla: Enhancing reasoning capability of vision-language-action models. *arXiv preprint arXiv:2511.15669*.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *ICCV*, pages 11941–11952. IEEE.

Ninghao Zhang, Bin Zhu, Shijie Zhou, and Jingjing Chen. 2026. Restoring linguistic grounding in vla models via train-free attention recalibration. *Preprint*, arXiv:2603.06001.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Tsung-Yi Lin, Gordon Wetzstein, Ming-Yu Liu, and Donglai Xiang. 2025. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *CVPR*, pages 1702–1713. Computer Vision Foundation / IEEE.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models. In *ICLR*.

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*.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong T. Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, and 35 others. 2023. RT-2: vision-language-action models transfer web knowledge to robotic control. In *CoRL*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR.

A Use of LLM Statement

An LLM was used only for polishing language; the scientific content is entirely original.

B Technical Appendix

Appendix Table of Contents

1. **Anticipatory defense on the choice of test environments and metrics** (§ C)
   Justification for using MINI-BEHAVIOR over alternatives (e.g., iGibson, CALVIN) based on controllability, symbolic abstraction, and isolation of language-grounding challenges.

2. **Symbolic Representation of the Embodied Task** (§ D) Formal definition of states, actions, and planning problems using classical STRIPS/PDDL semantics.

3. **Details of the MINI-BEHAVIOR Environment** (§ E) Overview of scene layout, object types, action space, and perception model.

4. **Details of the MINI-BEHAVIOR-GRAN Extension** (§ F) Procedure for generating multi-granularity instructions, details on the feature set, dynamic instructor, and dataset construction.

5. **Rule Set and Natural Language Prompt Examples** (§ G) Details of the rule set bank of 20 tasks and sample generated instructions at different granularities.

6. **Implementation Details** (§ H) Model architecture (Prismatic-7B backbone, SigLIP+DINOv2 visual encoders), training hyperparameters, optimizer settings. VLM Predictor architecture and training details.

7. **Additional Experimental Results** (§ I) Extended analysis and ablations.

C Anticipatory defense on the choice of test environments and metrics

In this work, we selected Mini-BEHAVIOR as our testbed. This choice is motivated by several key considerations aligned with our research objectives. Our research objective is not to solve the full stack of robotic embodiment (which includes perception, continuous control, and physics), but specifically to isolate and analyze the impact of instruction granularity on language grounding and long-horizon planning.

Therefore, high-fidelity simulators (e.g., iGibson (Li et al., 2022a), BEHAVIOR-1K (Li et al., 2022b)) introduce significant noise through complex texture rendering and continuous motor control. While realistic, these factors act as confounders. When an agent fails in such environments, it is often difficult to attribute the failure to a lack of linguistic understanding (granularity issues) or a failure in low-level execution (e.g., grasping physics or occlusion). Mini-BEHAVIOR retains the high-level decision complexity of household tasks—long horizons, multi-object interactions, and state dependencies—while abstracting away low-level sensorimotor noise.

In the perspective of fast prototyping and iterative experimentation, we have to also consider the rendering and physics simulation costs. Mini-BEHAVIOR's grid-based discrete control and simplified visual representation allow for rapid data generation and model training.

Another consideration is the diversity of the task structures so that we can systematically vary instruction granularity across a wide range of scenarios. As such, Crafter (Hafner, 2021), which focuses on survival in a single domain, lacks the diverse task structures needed to obtain robust insights about instruction granularity.

Table 6 provides a detailed comparison of Mini-BEHAVIOR against other prominent benchmarks, highlighting why they were less suitable for this specific investigation.

D Symbolic Representation of the Embodied Task

We formalize the Mini-BEHAVIOR environment as a deterministic grid-based symbolic world. This allows us to represent the challenge as a classical planning problem $\mathcal{P} = \langle \mathcal{F}, \mathcal{A}, s_0, G \rangle$, where:

- $\mathcal{F}$ is a finite set of propositional fluents representing the state variables of the environment (e.g., object locations, states like *cooked* or *open*).
- $\mathcal{A}$ is a finite set of actions available to the agent.
- $s_0 \subseteq \mathcal{F}$ is the initial state configuration.
- $G \subseteq \mathcal{F}$ is the goal condition that must be satisfied.

A state $s \in \mathcal{S}$ is defined as a truth valuation over $\mathcal{F}$, with the state space $\mathcal{S} \subseteq 2^{\mathcal{F}}$ encompassing all valid combinations of fluent truth values.

E Details of MINI-BEHAVIOR Environment

To operationalize our formalism, we developed a **Universal PDDL Domain Model** (available at `minibehavior_gran_domain_model.pddl` in the codebase) capable of supporting all 20 household tasks without modification. This unified model comprises over 1,000 lines of PDDL 2.1 definitions and is fully compatible with the LAMA planner. We highly recommend checking the codebase for the complete domain model.

Table 6: Comparison of Embodied AI Environments regarding their suitability for studying Instruction Granularity.

Environment	Control	Horizon	Visuals	Symbolic Abstraction	Limitation
Mini-BEHAVIOR(*)	Discrete	Long	Grid	**Support**	**Ideal balance:** Supports symbolic state abstraction; diverse task suite; removes control noise while retaining logic complexity.
Crafter	Discrete	Long	Cartoon	Partial	Single task suite (survival); partially observable (memory heavy); few object types.
Meta-World	Continuous	Short	3D	No	Continuous control; no symbolic abstraction; focus on multi-task RL/manipulation, not long-horizon planning.
LIBERO	Continuous	Short	3D	No	Robotic arm focus; continuous control; no symbolic abstraction; VLA success rate already saturated (tasks relatively easy).
RoboTwin 2.0	Continuous	Short	3D	No	Continuous control; no symbolic abstraction; dual-arm manipulation focus; hard to segment expert trajectories and construct granular instructions.
CALVIN	Continuous	Medium	3D	No	Hard to design instruction granularity due to continuous action space (though provides diverse language); confounds planning with control difficulties.
ALFRED	Discrete	Long	3D Photo	Support	3D partially observable; poor infrastructure support for VLA training; high rendering cost.
iGibson / BEHAVIOR-1K	Continuous	Long	3D Photo	Partial	Good for long-horizon tasks, but 3D realistic vision context becomes a confounder.

Table 7: Planning complexity in typical tasks.

Metric	Typical Range
Objects	15–40
Grid Locations	20–60
Ground Actions	10^3–10^5
Plan Length	50–200 primitive actions
Planning Time (LAMA)	0.5–300+ seconds

Table 8: Boolean State Features

Feature	Applicable Objects	Semantics
`Holding`	All Items	Is the agent holding the specified object?
`isOpened`	Cabinets, Drawers, Doors	Is the container/door open?
`isToggled`	Sinks, Stoves, Switches	Is the appliance turned on?
`isSoaked`	Rags, Sponges	Is the cleaning tool wet?
`isCleaned`	Shoes, Dishes, Floor	Is the object free of stains?
`isDustFree`	Furniture, Surfaces	Is the object free of dust?

Search Space Characteristics

F Details of MINI-BEHAVIOR-GRAN Extension

MINI-BEHAVIOR provides the foundational task skeletons, which we extend in MINI-BEHAVIOR-GRAN to support multi-granularity instruction generation. A core component of this extension is the definition of a symbolic feature set used to generate rule-based policy sketches.

F.1 Feature Categories

The environment defines two primary categories of features: Boolean State Features and Quantitative Features.

F.1.1 Boolean State Features

These features evaluate whether a specific object instance satisfies a predicate. They require grounding (binding to specific object instances, e.g., `rag-01`).

F.1.2 Quantitative and Spatial Features

These features capture distances, counts, and spatial relationships. Crucially, they support temporal comparison allowing conditions such as "distance decreased" or "count dropped to zero."

- **`DistanceToNearest(type, condition)`**: Measures the Manhattan distance from the agent to the nearest object of a specific type.
  - *Conditions:* `=0` (adjacent), `>0`, `increase`, `decrease`, `change`.

Table 9: Available Count Functions for Rule Definitions

Category	Function Description
State-Based	`count_not_cleaned`: Objects that are currently stained. `count_not_soaked`: Cleaning tools that are dry. `count_not_toggled`: Appliances that are turned off. `count_closed`: Containers/doors that are closed. `count_not_sliced`: Food items that require slicing.
Spatial	`count_not_onfloor`: Objects not placed on the floor layer. `count_isolated`: Objects with no neighbors of the same type. `count_not_inside_target`: Objects not contained within a specific furniture. `count_not_ontop`: Objects not placed on top of a specific surface. `count_not_near_target`: Objects not within 1 step of a target type.
Task-Specific	`count_incomplete_salad`: Plates lacking necessary ingredients. `count_not_complete_tables`: Tables lacking specific setups (e.g., candles).

- **`Count(type, condition, function)`**: Counts the number of objects of a specific type that satisfy a filter function.
  - *Conditions:* `=0`, `>0`, `increase`, `decrease`.

F.2 Count Functions

A diverse set of counting functions is implemented to support complex logical conditions (e.g., "all shoes are clean" is equivalent to *count_not_cleaned == 0*).

F.3 Reference Plan Generation and Task Complexity Analysis

While the original MINI-BEHAVIOR framework provides the high-level skeletons for household activities, it does not provide reference plans by default. Thus, we utilize the BFWS (Best-First Width Search) classical planner to generate reference plans for all 20 tasks in MINI-BEHAVIOR-GRAN. These reference plans serve as the ground truth for our agents and allow us to quantitatively categorize the complexity of each task.

Detailed statistics regarding the specific rule sets used for instruction generation and the resulting natural language statistics are discussed in the next section.

G Rule Set and Natural Language Prompt Examples For Different Tasks

We present the rank 0 rule set for 10 over 20 household tasks in the Mini-BEHAVIOR-Gran environment. For other levels, please refer to the codebase at `mini-behavior-gran/mini_behavior/utils/policy_sketch/`.

G.1 Task 1: Laying Wood Floors

Features:

- H: holding an isolated wood (plywood)
- p: distance to the nearest isolated wood
- t: distance to an arbitrary (non-held) wood we'll place next to
- n: number of isolated woods

$$\{\neg H, p > 0, n > 0\} \mapsto \{p\downarrow, t?\} \quad \text{; move close to isolated wood}$$
$$\{\neg H, p = 0, n > 0\} \mapsto \{H\} \quad \text{; pick up isolated wood when reachable}$$
$$\{H, t > 0, n > 0\} \mapsto \{t\downarrow\} \quad \text{; move to some wood location}$$
$$\{H, n > 0, t = 0\} \mapsto \{H?, n\downarrow, p?\} \quad \text{; drop the isolated wood next to other wood}$$

G.2 Task 2: Preparing Salad

Features:

- H_n: holding a not-sliceable food
- H_s: holding a sliceable food
- H_k: holding a slicer
- U: number of sliceable food that is not sliced yet
- d_p: distance to the selected incomplete plate p
- d_n, d_s: distances to nearest not-sliceable / sliceable
- k: distance to a slicer (knife)
- B: selected plate p complete bottom salad
- n: number of incomplete plates
- nn: number of plates that do not have bottom salad yet

Acquire slicer:
$$\{U > 0, \neg H_k, k > 0\} \mapsto \{k\downarrow\} \quad \text{; move to slicer}$$
$$\{U > 0, \neg H_k, k = 0\} \mapsto \{H_k\} \quad \text{; hold a slicer}$$
$$\{U > 0, H_k, d_s > 0\} \mapsto \{d_s\downarrow\} \quad \text{; move to sliceable food}$$
$$\{U > 0, H_k, d_s = 0\} \mapsto \{U\downarrow, \neg H_k\} \quad \text{; cut sliceable food}$$
$$\{U = 0, H_k\} \mapsto \{\neg H_k\} \quad \text{; put down slicer}$$

Place bottom salad:
$$\{n > 0, U = 0, nn > 0, \neg B, \neg H_n, d_n > 0\} \mapsto \{d_n\downarrow\}$$
$$\{n > 0, U = 0, nn > 0, \neg B, \neg H_n, d_n = 0\} \mapsto \{H_n\}$$
$$\{n > 0, U = 0, nn > 0, \neg B, H_n, d_p > 0\} \mapsto \{d_p\downarrow\}$$
$$\{n > 0, U = 0, nn > 0, \neg B, H_n, d_p = 0\} \mapsto \{B, nn\downarrow, \neg H_n\}$$

Handle sliceable salad:
$$\{nn = 0, n > 0, U = 0, \neg H_s, d_s > 0\} \mapsto \{d_s\downarrow\} \quad \text{; move to sliceable food}$$
$$\{nn = 0, n > 0, U = 0, \neg H_s, d_s = 0\} \mapsto \{H_s\} \quad \text{; pick up sliceable food}$$
$$\{nn = 0, n > 0, U = 0, H_s, d_p > 0\} \mapsto \{d_p\downarrow\} \quad \text{; move to the plate}$$
$$\{nn = 0, n > 0, U = 0, H_s, d_p = 0\} \mapsto \{n\downarrow, \neg H_s\} \quad \text{; put down sliced food}$$

G.3 Task 3: Cleaning Up the Kitchen Only

Features:

- *Holding flags:* H_b (blender), H_f (food), H_s (soap), H_r (rag), H_p (plate), H_o (vegetable oil)
- *Distances:* d_{ct} (countertop), d_{fr} (fridge), d_{si} (sink), d_{c1} (cabinet for plate), d_{c2} (cabinet for oil), d_x (to object x)
- *State booleans:* Open(c), Tog(s), Soaked, DustFree(c), Clean(p), OilIn(c_2), Diff
- *Counters:* N_b (blenders not on countertop), N_a (apples not in fridge), N_{cas} (casseroles not in fridge), N_s (soaps not next to sink), N_{cab} (cabinets not dusted), N_r (rags not next to sink), N_{p_clean} (plates not clean), N_{oil} (oil not in cabinet), N_{p_store} (plates not stored), N_{close} (closed furniture)

Open Cabinet and Fridge:
$$\{N_{close} > 0, d_c > 0\} \mapsto \{d_c\downarrow\} \quad \text{; move toward closed furniture}$$
$$\{N_{close} > 0, d_c = 0\} \mapsto \{N_{close}\downarrow\}$$

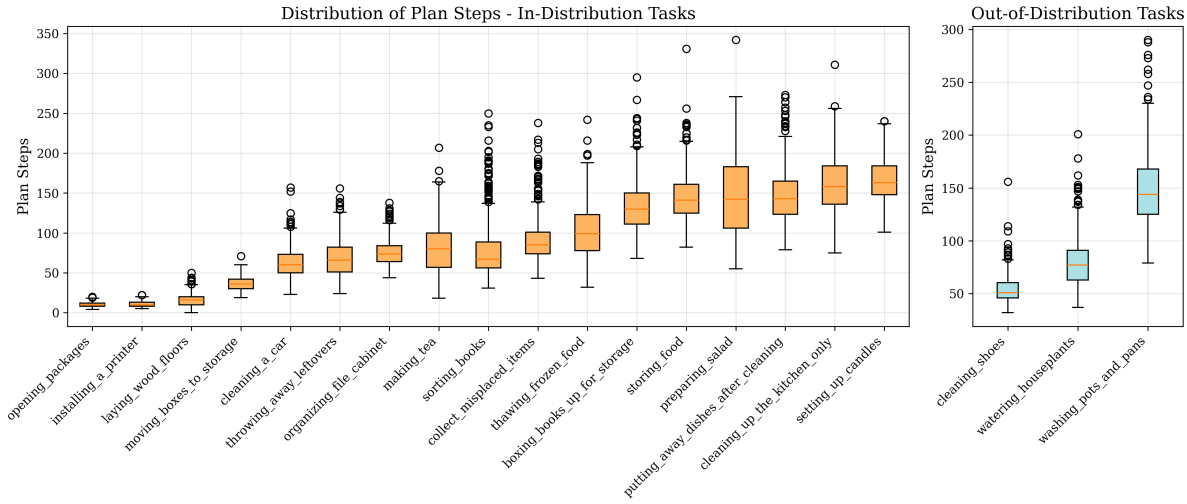

Figure 8: Distribution of reference plan steps required for tasks in MINI-BEHAVIOR-GRAN obtained by BWFS classical planner.

Oil placement:

$\{N_{close} = 0, \neg H_o, d_{oil} > 0\} \mapsto \{d_{oil} \downarrow\}$; approach oil

$\{N_{close} = 0, \neg H_o, d_{oil} = 0\} \mapsto \{H_o\}$; pick up oil

$\{N_{close} = 0, H_o, N_{oil} > 0, d_{c2} > 0\} \mapsto \{d_{c2} \downarrow\}$; go to cabinet c_2

$\{N_{close} = 0, H_o, N_{oil} > 0, d_{c2} = 0\} \mapsto \{N_{oil} \downarrow, \neg H_o\}$; place oil in c_2

Plate cleaning:

$\{\neg \text{Tog}(s), d_{si} > 0\} \mapsto \{d_{si} \downarrow\}$; toggle sink on

$\{\neg \text{Tog}(s), d_{si} = 0\} \mapsto \{\text{Tog}(s)\}$

$\{\neg H_r, d_{rag} > 0\} \mapsto \{d_{rag} \downarrow\}$; get rag

$\{\neg H_r, d_{rag} = 0\} \mapsto \{H_r\}$

$\{H_r, \neg \text{Soaked}, d_{si} > 0\} \mapsto \{d_{si} \downarrow\}$; soak rag

$\{H_r, \neg \text{Soaked}, d_{si} = 0\} \mapsto \{\text{Soaked}\}$

$\{H_r, \text{Soaked}, N_{p_clean} > 0, d_{plate} > 0\} \mapsto \{d_{plate} \downarrow\}$

$\{H_r, \text{Soaked}, N_{p_clean} > 0, d_{plate} = 0\} \mapsto \{N_{p_clean} \downarrow, d_{plate}?\}$

Plate placement distinct from oil:

$\{N_{close} = 0, N_{oil} = 0, N_{p_clean} = 0, \neg H_p, N_{p_store} > 0, d_{plate} > 0\} \mapsto \{d_{plate} \downarrow\}$

$\{N_{close} = 0, N_{oil} = 0, N_{p_clean} = 0, \neg H_p, N_{p_store} > 0, d_{plate} = 0\} \mapsto \{H_p\}$

$\{N_{close} = 0, N_{oil} = 0, N_{p_clean} = 0, H_p, N_{p_store} > 0, d_{c1} > 0\} \mapsto \{d_{c1} \downarrow\}$

$\{N_{close} = 0, N_{oil} = 0, N_{p_clean} = 0, H_p, N_{p_store} > 0, d_{c1} = 0\} \mapsto \{N_{p_store} \downarrow, \neg H_p\}$

Blender to countertop:

$\{N_b > 0, \neg H_b, d_{blender} > 0\} \mapsto \{d_{blender} \downarrow\}$

$\{N_b > 0, \neg H_b, d_{blender} = 0\} \mapsto \{H_b\}$

$\{H_b, N_b > 0, d_{ct} > 0\} \mapsto \{d_{ct} \downarrow\}$

$\{H_b, N_b > 0, d_{ct} = 0\} \mapsto \{N_b \downarrow, \neg H_b\}$

Apple to fridge:

$\{N_{close} = 0, \neg H_f, d_{apple} > 0\} \mapsto \{d_{apple} \downarrow\}$

$\{N_{close} = 0, \neg H_f, d_{apple} = 0\} \mapsto \{H_f\}$

$\{N_{close} = 0, H_f, N_a > 0, d_{fr} > 0\} \mapsto \{d_{fr} \downarrow\}$

$\{N_{close} = 0, H_f, N_a > 0, d_{fr} = 0\} \mapsto \{N_a \downarrow, \neg H_f\}$

Casserole to fridge:

$\{N_{close} = 0, \neg H_f, d_{casserole} > 0\} \mapsto \{d_{casserole} \downarrow\}$

$\{N_{close} = 0, \neg H_f, d_{casserole} = 0\} \mapsto \{H_f\}$

$\{N_{close} = 0, H_f, N_{cas} > 0, d_{fr} > 0\} \mapsto \{d_{fr} \downarrow\}$

$\{N_{close} = 0, H_f, N_{cas} > 0, d_{fr} = 0\} \mapsto \{N_{cas} \downarrow, \neg H_f\}$

Cabinet dusting:

$\{\neg H_r, d_{rag} > 0\} \mapsto \{d_{rag} \downarrow\}$; get rag

$\{\neg H_r, d_{rag} = 0\} \mapsto \{H_r\}$

$\{H_r, N_{cab} > 0, d_c > 0\} \mapsto \{d_c \downarrow\}$; clean cabinet

$\{H_r, N_{cab} > 0, d_c = 0\} \mapsto \{N_{cab} \downarrow, \neg H_r\}$

Rag near/inside sink:

$\{N_{cab} = 0, N_{p_clean} = 0, \neg H_r, d_{rag} > 0\} \mapsto \{d_{rag} \downarrow\}$

$\{N_{cab} = 0, N_{p_clean} = 0, \neg H_r, d_{rag} = 0\} \mapsto \{H_r\}$

$\{N_{cab} = 0, N_{p_clean} = 0, H_r, N_r > 0, d_{si} > 0\} \mapsto \{d_{si} \downarrow\}$

$\{N_{cab} = 0, N_{p_clean} = 0, H_r, N_r > 0, d_{si} = 0\} \mapsto \{N_r \downarrow, \neg H_r\}$

Soap next to sink:

$\{N_{cab} = 0, N_{p_clean} = 0, \neg H_s, d_{soap} > 0\} \mapsto \{d_{soap} \downarrow\}$

$\{N_{cab} = 0, N_{p_clean} = 0, \neg H_s, d_{soap} = 0\} \mapsto \{H_s\}$

$\{N_{cab} = 0, N_{p_clean} = 0, H_s, N_s > 0, d_{si} > 0\} \mapsto \{d_{si} \downarrow\}$

$\{N_{cab} = 0, N_{p_clean} = 0, H_s, N_s > 0, d_{si} = 0\} \mapsto \{N_s \downarrow, \neg H_s\}$

G.4 Task 4: Organizing File Cabinet

Features:

- H_m: holding a marker
- H_f: holding a folder
- H_d: holding a document
- p_m: distance to the nearest marker
- p_f: distance to the nearest folder
- p_d: distance to the nearest document
- p_t: distance to a table
- p_c: distance to a valid cabinet target
- N_m: number of markers not on a table
- N_f: number of folders not in a cabinet
- N_d: number of documents not in a cabinet
- N_o: number of closed cabinets

Opening Cabinet:

$\{N_o > 0, p_c > 0\} \mapsto \{p_c \downarrow\}$; move toward a closed cabinet

$\{N_o > 0, p_c = 0\} \mapsto \{N_o \downarrow\}$; open the cabinet when reachable

Marker task:

$\{N_m > 0, \neg H_m, p_m > 0\} \mapsto \{p_m \downarrow, p_t?\}$; move toward the nearest marker

$\{N_m > 0, \neg H_m, p_m = 0\} \mapsto \{H_m, p_t?\}$; pick up the marker when reachable

$\{N_m > 0, H_m, p_t > 0\} \mapsto \{p_t \downarrow\}$; move toward a table

$\{N_m > 0, H_m, p_t = 0\} \mapsto \{N_m \downarrow, \neg H_m, p_m?\}$; put marker on table

Folder task:

$\{N_f > 0, N_o = 0, \neg H_f, p_f > 0\} \mapsto \{p_f \downarrow, p_c?\}$

$\{N_f > 0, N_o = 0, \neg H_f, p_f = 0\} \mapsto \{H_f, p_c?\}$

$\{N_f > 0, N_o = 0, H_f, p_c > 0\} \mapsto \{p_c \downarrow\}$

$\{N_f > 0, N_o = 0, H_f, p_c = 0\} \mapsto \{N_f \downarrow, \neg H_f, p_f?\}$

Document task:

$\{N_d > 0, N_o = 0, \neg H_d, p_d > 0\} \mapsto \{p_d \downarrow, p_c?\}$

$\{N_d > 0, N_o = 0, \neg H_d, p_d = 0\} \mapsto \{H_d, p_c?\}$

$\{N_d > 0, N_o = 0, H_d, p_c > 0\} \mapsto \{p_c \downarrow\}$

$\{N_d > 0, N_o = 0, H_d, p_c = 0\} \mapsto \{N_d \downarrow, \neg H_d, p_d?\}$

G.5 Task 5: Thawing Frozen Food

Features:

- H_{fish}: holding a fish
- H_{olive}: holding an olive
- H_{date}: holding a date
- p_{fish}: distance to the nearest fish
- p_{olive}: distance to the nearest olive
- p_{date}: distance to the nearest date
- p_{sink}: distance to a sink
- N_{fish}: number of fish not next to a sink
- N_{olive}: number of olives not next to a sink

Task	Split	Class	Goal Description	Mean Steps
opening_packages	ID	**short**	Open every package.	10.2
installing_a_printer	ID	**short**	Place a printer on a table and turn it on.	10.6
laying_wood_floors	ID	**short**	Lay every plywood sheet on the floor, each touching at least one other sheet.	15.9
moving_boxes_to_storage	ID	**short**	Place every carton inside a shelf.	36.3
cleaning_a_car	ID	**short**	Make at least one car dust-free. Place a rag and soap together inside a bucket.	62.6
throwing_away_leftovers	ID	**middle**	Throw away every hamburger by placing it in a trash can.	68.3
organizing_file_cabinet	ID	**middle**	Put a marker on a table and file every folder and document in a cabinet.	75.5
making_tea	ID	**middle**	Slice a lemon. Put a teapot on an activated stove. Keep a soaked teabag with the teapot.	78.7
sorting_books	ID	**middle**	Place every book and every hardback on a shelf.	78.8
collect_misplaced_items	ID	**middle**	Place all gym shoes, necklaces, notebooks, and socks on a table.	90.7
thawing_frozen_food	ID	**middle**	Place a date label next to a fish. Put every fish next to a sink. Set an olive next to a sink.	101.7
boxing_books_up_for_storage	ID	**long**	Place every book inside a box.	134.4
storing_food	ID	**long**	Store every cookable item inside a cabinet.	144.6
preparing_salad	ID	**long**	Slice all apples and tomatoes; place them on plates. Put every radish and lettuce on plates. Ensure each plate holds at least one cookable item.	146.7
putting_away_dishes_after_cleaning	ID	**long**	Put every plate inside a cabinet.	147.7
cleaning_up_the_kitchen_only	ID	**long**	Put every blender on a countertop. Refrigerate all apples and casseroles. Keep soap next to a sink. Make plates unstained and cabinets dust-free. Place each rag either beside or inside the sink. Store plates and vegetable oil in different cabinets.	161.0
setting_up_candles	ID	**long**	On every table, place three distinct candles on top.	165.2
cleaning_shoes	OOD	**short**	Leave a towel on the floor and make sure every shoe is unstained and dust-free.	54.2
watering_houseplants	OOD	**middle**	Water every potted plant until it is soaked.	79.4
washing_pots_and_pans	OOD	**long**	Clean every teapot, kettle, and pan so they're unstained, then store each inside a cabinet.	149.5

Table 10: Classification of tasks in MINI-BEHAVIOR-GRAN based on the median number of plan steps, as determined by the BWFS planner. Tasks are categorized into short, middle, and long classes to reflect varying planning complexity based on plan length.

- N_{date}: number of dates not next to a fish

Fish Task:
$$\{\neg H_{fish}, p_{fish} > 0\} \mapsto \{p_{fish} \downarrow\}$$
$$\{\neg H_{fish}, p_{fish} = 0\} \mapsto \{H_{fish}\}$$
$$\{H_{fish}, p_{sink} > 0\} \mapsto \{p_{sink} \downarrow\}$$
$$\{H_{fish}, p_{sink} = 0\} \mapsto \{N_{fish} \downarrow, \neg H_{fish}, p_{fish}?\}$$

Olive Task:
$$\{\neg H_{olive}, p_{olive} > 0\} \mapsto \{p_{olive} \downarrow\}$$
$$\{\neg H_{olive}, p_{olive} = 0\} \mapsto \{H_{olive}\}$$
$$\{H_{olive}, p_{sink} > 0\} \mapsto \{p_{sink} \downarrow\}$$
$$\{H_{olive}, p_{sink} = 0\} \mapsto \{N_{olive} \downarrow, \neg H_{olive}, p_{olive}?\}$$

Date Task:
$$\{\neg H_{date}, N_{\text{fish olive}} = 0, p_{date} > 0\} \mapsto \{p_{date} \downarrow\}$$
$$\{\neg H_{date}, N_{\text{fish olive}} = 0, p_{date} = 0\} \mapsto \{H_{date}\}$$
$$\{H_{date}, N_{\text{fish olive}} = 0, p_{fish} > 0\} \mapsto \{p_{fish} \downarrow\}$$
$$\{H_{date}, N_{\text{fish olive}} = 0, p_{fish} = 0\} \mapsto \{N_{date} \downarrow, \neg H_{date}, p_{date}?\}$$

G.6 Task 6: Making Tea

Features:

- H_l: holding a lemon
- H_{knife}: holding a knife
- H_t: holding a teapot
- H_{tb}: holding a tea bag
- p_l: distance to the nearest lemon
- p_{knife}: distance to the knife
- p_t: distance to the nearest teapot
- p_{tb}: distance to the nearest tea bag
- p_s: distance to the stove
- p_{sink}: distance to the sink
- is_sliced(lemon): whether the lemon is sliced
- is_toggled(stove): whether the stove is toggled on
- is_toggled(sink): whether the sink is toggled on
- is_soaked(tea_bag): whether the tea bag is soaked
- atsamelocation(tea_bag, teapot): tea bag at same location as teapot
- onTop(teapot, stove): teapot on stove

Toggle stove:
$$\{\neg \text{is_toggled(stove)}, p_s > 0\} \mapsto \{p_s \downarrow\}$$
$$\{\neg \text{is_toggled(stove)}, p_s = 0\} \mapsto \{\text{is_toggled(stove)}\}$$

Slice lemon:
$$\{\neg H_{knife}, p_{knife} > 0, \neg \text{is_sliced(lemon)}\} \mapsto \{p_{knife} \downarrow\}$$
$$\{\neg H_{knife}, p_{knife} = 0, \neg \text{is_sliced(lemon)}\} \mapsto \{H_{knife}\}$$
$$\{H_{knife}, \neg \text{is_sliced(lemon)}, p_l > 0\} \mapsto \{p_l \downarrow\}$$
$$\{H_{knife}, \neg \text{is_sliced(lemon)}, p_l = 0\} \mapsto \{\text{is_sliced(lemon)}\}$$
$$\{H_{knife}, \text{is_sliced(lemon)}\} \mapsto \{\neg H_{knife}, p_{knife}?\}$$

Handle teapot:
$$\{\neg \text{atsameloc(tb, tp)}, \neg \text{onTop(tp, st)}, \neg H_t, p_t > 0\} \mapsto \{p_t \downarrow\}$$
$$\{\neg \text{atsameloc(tb, tp)}, \neg \text{onTop(tp, st)}, \neg H_t, p_t = 0\} \mapsto \{H_t\}$$
$$\{\neg \text{atsameloc(tb, tp)}, \neg \text{onTop(tp, st)}, H_t, p_s > 0\} \mapsto \{p_s \downarrow\}$$
$$\{\neg \text{atsameloc(tb, tp)}, \neg \text{onTop(tp, st)}, H_t, p_s = 0\} \mapsto \{\text{onTop(tp, st)}, \neg H_t, p_t?\}$$
$$\{\neg \text{atsameloc(tb, tp)}, \text{onTop(tp, st)}, \neg H_{tb}, p_{tb} > 0\} \mapsto \{p_{tb} \downarrow\}$$
$$\{\neg \text{atsameloc(tb, tp)}, \text{onTop(tp, st)}, \neg H_{tb}, p_{tb} = 0\} \mapsto \{H_{tb}\}$$
$$\{\neg \text{atsameloc(tb, tp)}, \text{onTop(tp, st)}, H_{tb}, p_t > 0\} \mapsto \{p_t \downarrow\}$$
$$\{H_{tb}, \neg \text{atsameloc(tb, tp)}, \text{onTop(tp, st)}, p_t = 0\} \mapsto \{\text{atsameloc(tb, tp)}, \neg H_{tb}, p_{tb}?\}$$

G.7 Task 7: Opening Packages

Features:

- p: distance to the nearest package
- N: number of unopened packages

$$\{N > 0, p > 0\} \mapsto \{p \downarrow\} \quad \text{; move toward the nearest unopened package}$$
$$\{N > 0, p = 0\} \mapsto \{N \downarrow\} \quad \text{; open the package when reachable}$$

G.8 Task 8: Boxing Books Up for Storage

Features:

- H: holding an unboxed book
- p: distance to the nearest unboxed book
- b: distance to a valid box target
- N: number of unboxed books

$$\{\neg H, p > 0\} \mapsto \{p \downarrow, b?\} \quad \text{; move toward the nearest unboxed book}$$
$$\{\neg H, p = 0\} \mapsto \{H\} \quad \text{; pick up the book when reachable}$$
$$\{H, N > 0, b > 0\} \mapsto \{b \downarrow\} \quad \text{; move toward a chosen box spot}$$
$$\{H, N > 0, b = 0\} \mapsto \{\neg H, N \downarrow, p?\} \quad \text{; place the book into the box}$$

G.9 Task 9: Collect Misplaced Items

Features:

- H_s: holding a gym shoe
- H_n: holding a necklace
- H_b: holding a notebook
- H_k: holding a sock
- p_s: distance to the nearest gym shoe
- p_n: distance to the nearest necklace
- p_b: distance to the nearest notebook
- p_k: distance to the nearest sock
- p_t: distance to a table
- N_s: number of gym shoes not on a table
- N_n: number of necklaces not on a table
- N_b: number of notebooks not on a table
- N_k: number of socks not on a table

Gym shoe task:
$$\{\neg H_s, N_s > 0, p_s > 0\} \mapsto \{p_s \downarrow, p_t?\}$$
$$\{\neg H_s, N_s > 0, p_s = 0\} \mapsto \{H_s, p_t?\}$$
$$\{H_s, N_s > 0, p_t > 0\} \mapsto \{p_t \downarrow\}$$
$$\{H_s, N_s > 0, p_t = 0\} \mapsto \{N_s \downarrow, \neg H_s, p_s?\}$$

Necklace task:
$$\{\neg H_n, N_n > 0, p_n > 0\} \mapsto \{p_n \downarrow, p_t?\}$$
$$\{\neg H_n, N_n > 0, p_n = 0\} \mapsto \{H_n, p_t?\}$$
$$\{H_n, N_n > 0, p_t > 0\} \mapsto \{p_t \downarrow\}$$
$$\{H_n, N_n > 0, p_t = 0\} \mapsto \{N_n \downarrow, \neg H_n, p_n?\}$$

Notebook task:
$$\{\neg H_b, N_b > 0, p_b > 0\} \mapsto \{p_b \downarrow, p_t?\}$$
$$\{\neg H_b, N_b > 0, p_b = 0\} \mapsto \{H_b, p_t?\}$$
$$\{H_b, N_b > 0, p_t > 0\} \mapsto \{p_t \downarrow\}$$
$$\{H_b, N_b > 0, p_t = 0\} \mapsto \{N_b \downarrow, \neg H_b, p_b?\}$$

Sock task:
$$\{\neg H_k, N_k > 0, p_k > 0\} \mapsto \{p_k \downarrow, p_t?\}$$
$$\{\neg H_k, N_k > 0, p_k = 0\} \mapsto \{H_k, p_t?\}$$
$$\{H_k, N_k > 0, p_t > 0\} \mapsto \{p_t \downarrow\}$$
$$\{H_k, N_k > 0, p_t = 0\} \mapsto \{N_k \downarrow, \neg H_k, p_k?\}$$

G.10 Task 10: Putting Away Dishes After Cleaning

Features:

- H: holding a plate
- p: distance to the nearest plate
- b: distance to a valid cabinet target
- N: number of unboxed plates
- is_opened: whether the cabinet is opened

$$\{\neg \text{is_opened}, b > 0\} \mapsto \{b \downarrow\} \quad \text{; move toward the cabinet}$$
$$\{\neg \text{is_opened}, b = 0\} \mapsto \{\text{is_opened}\} \quad \text{; open the cabinet when reachable}$$
$$\{\neg H, N > 0, p > 0, \text{is_opened}\} \mapsto \{p \downarrow, b?\}$$
$$\{\neg H, N > 0, p = 0, \text{is_opened}\} \mapsto \{H, b?\}$$
$$\{H, N > 0, b > 0, \text{is_opened}\} \mapsto \{b \downarrow\}$$
$$\{H, N > 0, b = 0, \text{is_opened}\} \mapsto \{\neg H, N \downarrow, p?\}$$

G.11 Instruction Generation Statistics

We used a template-based generator to translate the symbolic rule sets into natural language instructions, ensuring alignment with the VLA's training data format. Table 11 presents the summary statistics for the generated instructions across the three granularity levels.

H Implementation Details

H.1 Model Architecture

Our policy is built upon the OpenVLA architecture (Kim et al., 2024). Specifically, we utilize the Prismatic-7B backbone (Karamcheti et al., 2024), which integrates a frozen Llama-2-7B language model with fused visual features. The visual encoder consists of a dual-stream setup combining SigLIP (for semantic understanding) and DINOv2 (for fine-grained spatial geometry) (Zhai et al., 2023; Oquab et al., 2024).

H.2 Training Configuration

We fine-tune the model using Low-Rank Adaptation (LoRA) on the language model backbone, keeping the visual encoders frozen. All training setting variants (Pure, Centroid, Axial) share the same underlying training hyperparameters to ensure fair comparison.

Hardware and Duration: Training was conducted on a single node equipped with 8× NVIDIA A100 GPUs. A standard training run for 100,000 steps took approximately 23 hours to complete.

Table 12 summarizes the core hyperparameters used across our experiments.

Table 11: Statistics of generated natural language instructions across granularity levels.

Granularity	Count	Avg Length (words)	Std Length	Min	Max	Unique Tokens	TTR (Type-Token Ratio)	Avg Sent. Length
Fine (F)	1636	41.12	11.36	19	87	140	0.0021	41.12
Medium (M)	859	38.62	12.76	22	81	108	0.0033	38.62
Coarse (C)	587	33.60	13.39	22	95	94	0.0048	33.60

Table 12: Hyperparameters for OpenVLA fine-tuning on MINI-BEHAVIOR-GRAN.

Hyperparameter	Value
Base Model	Prismatic-7B (Llama-2)
Visual Encoders	SigLIP + DINOv2
Parameter Efficient Tuning	LoRA (Rank $r = 32$)
Compute Hardware	8× NVIDIA A100
Training Duration	∼23 hours
Optimization Steps	100,000
Steps Before Decay	80,000
Learning Rate	2.5×10^{-4}
Batch Size (per GPU)	$7 - 8$

We also train a lightweight VLM predictor to quantify instruction predictability via perplexity. The model is fine-tuned using Low-Rank Adaptation (LoRA) on the Qwen3-VL-4B-Instruct backbone, with vision encoders frozen to preserve visual grounding capabilities established during pretraining. All models are trained on MINI-BEHAVIOR-GRAN instruction and image training data pairs, with training and evaluation splits maintained consistently across experiments. Hardware and Duration: Training was conducted on a single node equipped with 4× NVIDIA RTX 5090 GPUs. Each training run for 2 epochs. Table 13 summarizes the core hyperparameters used for VLM predictor training.

Table 13: Hyperparameters for Qwen3-VL-4B-Instruct fine-tuning on MINI-BEHAVIOR-GRAN (High-domain).

Hyperparameter	Value
Base Model	Qwen3-VL-4B-Instruct
Visual Encoders	Frozen (Qwen3-VL native)
Parameter Efficient Tuning	LoRA (Rank $r = 64$, $\alpha = 128$)
LoRA Target Modules	All linear layers
Compute Hardware	4× NVIDIA RTX 5090
Training Epochs	2
Learning Rate	1.0×10^{-5}
Learning Rate Scheduler	Cosine with 10% warmup
Batch Size (per GPU)	32
Seed	[42,50,60]
Gradient Accumulation Steps	1
DeepSpeed Strategy	Zero-2

I Additional Experimental Results

This section provides additional detailed per-task performance breakdowns, and failure mode distributions.

I.1 Detailed Task Performance

Figure 9 details the success rates across all 20 tasks in the benchmark. We see some tasks are too difficult to solve and thus low across all instruction granularity. And then we also see the U shape trends in most tasks, showing the same phenomenon we discussed in **(Q1)** that both very fine and very coarse instructions are easier for VLA to follow.

I.2 Detailed Attention Analysis Table

We quantify the internal information-gathering behavior of Vision-Language-Action (VLA) models by analyzing the **attention weight distribution of the action tokens**. In these architectures, the action tokens serve as the terminal query point; their attention allocation reveals which input modalities (vision vs. language) the model prioritizes at different stages of its internal processing hierarchy.

I.2.1 Input Sequence Formalization

The input sequence $\mathcal{S}$ is composed of visual patches, linguistic tokens, and action tokens. Given a single sample, the sequence layout is defined as:

$$\mathcal{S} = [\texttt{BOS}, \underbrace{v_1, \ldots, v_P}_{\text{vision}}, \underbrace{t_1, \ldots, t_T}_{\text{text}}, \underbrace{a_1, \ldots, a_A}_{\text{action}}, \texttt{EOS}] \tag{1}$$

where P is the number of visual patch tokens, T is the number of text tokens (comprising both prompt prefix and task instruction), and A is the number of action tokens (where $A = d_{\text{action}} \times$ chunk size).

For analysis, we exclude the [BOS] and [EOS] boundary tokens. For each layer $\ell \in \{1, \ldots, L\}$, we extract the trimmed attention tensor:

$$\mathbf{A}^{(\ell)} \in \mathrm{R}^{H \times (P+T+A) \times (P+T+A)} \tag{2}$$

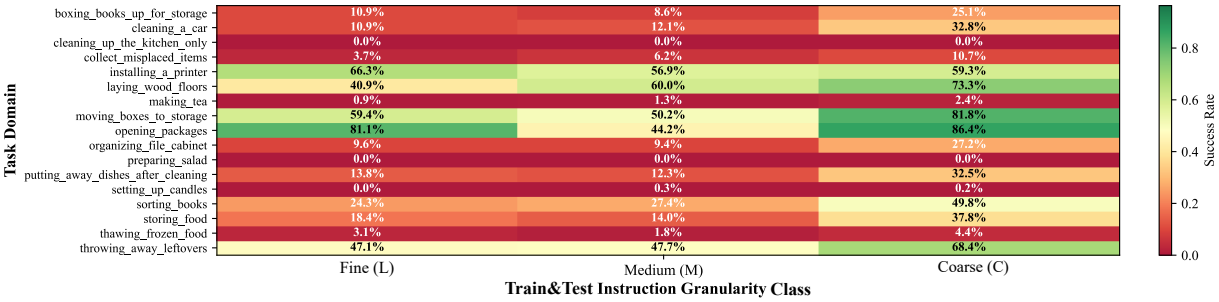

Figure 9: Detailed per-task success rates.

I.2.2 Partitioning of Semantic Regions

To distinguish between general linguistic context and specific task instructions, we partition the key dimension (columns) of $\mathbf{A}^{(\ell)}$ into four non-overlapping semantic regions:

Note: For models utilizing compressed instruction patches, the compressed embeddings are mapped to the Dynamic Text category to ensure functional consistency across architectures.

I.2.3 Quantitative Measurement

For each layer ℓ and head h, we isolate the **action-to-key** sub-matrix $\mathbf{A}^{(\ell)}_{h,\text{action}\to\text{key}} \in \mathrm{R}^{A\times(P+T+A)}$. The mean attention mass allocated to a specific region $R = [r_{\text{start}}, r_{\text{end}})$ is computed by summing weights across the region and averaging over the A action tokens:

$$\text{attn}_h(R) = \frac{1}{A}\sum_{i=1}^{A}\sum_{j=r_{\text{start}}}^{r_{\text{end}}-1} \mathbf{A}^{(\ell)}_{h,\text{action}\to\text{key}}[i,j] \quad (3)$$

I.2.4 Aggregation and Normalization

To derive a layer-wise modality reliance score, we perform the following:

1. **Head Averaging**: $\text{attn}(R) = \frac{1}{H}\sum_{h=1}^{H}\text{attn}_h(R)$.
2. **Sample Mean**: Values are averaged across all samples within a specific granularity-model pair.
3. **Simplex Normalization**: The final regional shares $\hat{a}_R$ are normalized such that $\sum_R \hat{a}_R = 1$.

I.2.5 Statistical Trend Hypothesis Testing

We define three binary indicators to test the **Shallow Grounding Hypothesis** layer-wise:

- **T1 (Lang ↑)**: $\hat{a}_{\text{text}}(\text{Medium}) > \hat{a}_{\text{text}}(\text{Fine})$
- **T2 (Lang ↓)**: $\hat{a}_{\text{text}}(\text{Medium}) > \hat{a}_{\text{text}}(\text{Coarse})$
- **T3 (Vis ↑)**: $\hat{a}_{\text{vis}}(\text{Coarse}) > \hat{a}_{\text{vis}}(\text{Medium})$

We assess the significance of these trends across $N = 32$ layers using a **one-sided Binomial test** ($H_0 : p = 0.5$). Significance is reported as $^*p < 0.05$ and $^{**}p < 0.01$. Confidence intervals for proportions are calculated using the **Clopper-Pearson** method. The full statistical breakdown is presented in Table 14.

I.3 Failure Analysis

Figure 10 presents the distribution of failure types across varying instruction widths. It confirms that low width encounters regression failures more frequently, while high width predominantly leads to stagnation failures. This aligns with our earlier observations in Figure 11 regarding the distinct failure modes associated with different instruction granularities. Agent failure distributions bifurcate strictly with width. Low-width, long-horizon tasks predominantly suffer from *regression*: the agent violates previously satisfied fluents and cycles back to prior states, reflecting subgoal serialization failures. High-width tasks ($w \geq 3$), by contrast, exhibit *stagnation*: the agent freezes, unable to resolve concurrent feature dependencies ($O(|\Phi|^w)$) required to initiate a valid state transition. This dichotomy mirrors the width-based complexity landscape.

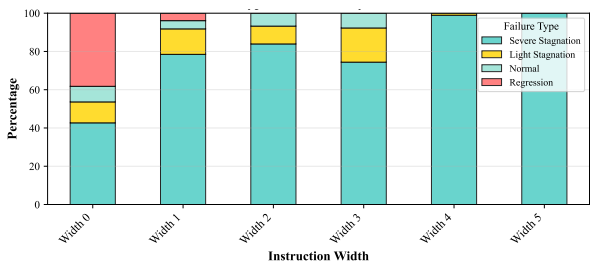

Figure 10: Failure type distribution across instruction widths.

Model	Trend	k/N	Prop.	95% CI	H_0	p-value
Autoregressive	T1	25/32	78.1%	[0.600, 0.907]	50.0%	0.0011
	T2	4/32	12.5%	[0.035, 0.290]	50.0%	1.0000
	T3	21/32	65.6%	[0.468, 0.814]	50.0%	0.1102
Parallel Decoding	T1	5/32	15.6%	[0.053, 0.328]	50.0%	1.0000
	T2	9/32	28.1%	[0.137, 0.467]	50.0%	0.9965
	T3	5/32	15.6%	[0.053, 0.328]	50.0%	1.0000
Discrete Diffusion	T1	17/32	53.1%	[0.347, 0.709]	50.0%	0.4300
	T2	23/32	71.9%	[0.533, 0.863]	50.0%	0.0100
	T3	24/32	75.0%	[0.566, 0.885]	50.0%	0.0035

Table 14: Detailed statistical analysis of layer-wise attention trends across architectures. P-values are calculated using a one-sided Binomial test against a null hypothesis (H_0) of 0.5. Confidence intervals (CI) are calculated at the 95% level.

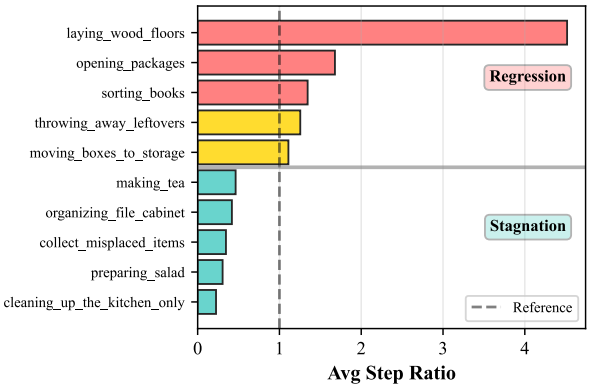

Figure 11: Low-width but long task horizon (e.g., laying woods) tends to cause regression, while high-width stagnates.

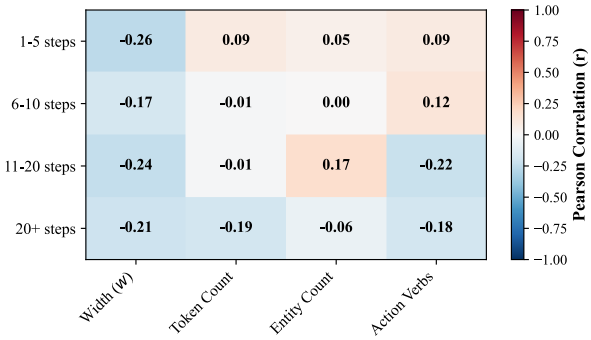

Figure 12: Correlation heatmap between instruction width and other linguistic heuristics.

I.4 More Plots for Width vs. Other Linguistic Heuristics as Metric for Language Grounding Complexity

As shown in the correlation heatmap Figure 12, w is the only metric that maintains a stable negative correlation with Success Rate across all horizons. In contrast, surface metrics like Token Count and Entity Count exhibit near-zero or even positive correlations in shorter horizons (e.g., $r = 0.09$ for 1-5 steps). This confirms the Complexity Paradox: while longer, more detailed instructions (higher token/entity counts) could actually provide helpful guidance that improves SR, increasing the latent planning width (w) invariably degrades performance.

Similarly, the consistency analysis in Figure 13 shows that w possesses the lowest standard deviation ($\sigma = 0.033$), indicating its predictive reliability is invariant to task horizons. Surface metrics, particularly Action Verbs ($\sigma = 0.155$), are highly volatile. It also shows that w provides the highest mean predictive power ($|r| = 0.220$), more than doubling the predictive strength of Token Count (0.076) or Entity Count (0.072).

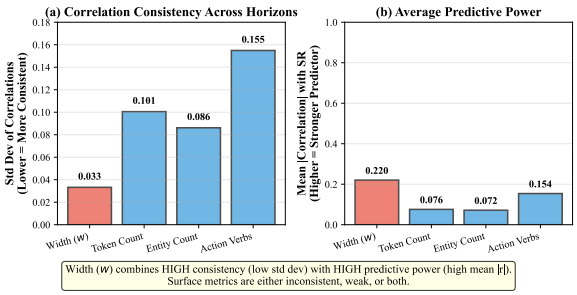

Figure 13: Consistency comparison between instruction width and other linguistic heuristics.

I.5 Sample Generated Instructions by Granularity

To illustrate the qualitative differences between granularity levels, we provide five random samples for each category below. Note how Low (L) granularity explicitly specifies navigation and preconditions (e.g., move toward," not yet at"), while High (H) granularity focuses almost exclusively on the final state change (e.g., "reduce the count of..."), shown in Table 15.

Fine Granularity (F) Samples

1. At step 11: When the count of uncleaned plate is above 0, not yet at sink 0 and sink is off, please try to move toward sink 0.
2. At step 8: When not yet at soap 0, not holding soap 0, the count of unwiped car is 0 and the count of rag not inside bucket is 0, please try to move toward soap 0.
3. At step 27: When the count of chip and oatmeal sugar and vegetable oil not inside cabinet is above 0, you are at oatmeal 1 and not holding oatmeal 1, please pick up oatmeal 1.
4. At step 15: When the count of plate not inside cabinet is above 0, you are at plate 3, the count of closed cabinet is 0 and not holding plate 3, please pick up plate 3.
5. At step 13: When the count of fish and olive not near sink is above 0, not yet at olive 0 and not holding olive 0, please try to move toward olive 0.

Medium Granularity (M) Samples

1. At step 2: When the count of plate not inside cabinet is above 0, the count of closed cabinet is 0 and not holding plate 2, please pick up plate 2.
2. At step 6: When the count of uncleaned plate is above 0, the count of dry rag is above 0 and holding rag 0, please reduce the count of dry rag.
3. At step 9: When the count of plate not inside cabinet 0 is above 0, the count of closed cabinet is 0, the count of vegetable oil not inside cabinet 1 is 0, the count of uncleaned plate is 0 and not holding plate 0, please pick up plate 0.
4. At step 13: When the count of book not inside box is above 0 and not holding book 2, please pick up book 2.
5. At step 9: When the count of chip and oatmeal sugar and vegetable oil not inside cabinet is above 0, holding chip 1 and the count of closed cabinet is 0, please not holding chip 1, then reduce the count of chip and oatmeal sugar and vegetable oil not inside cabinet.

Coarse Granularity (C) Samples

1. At step 2: When the count of hamburger not inside ashcan is above 0, please reduce the count of hamburger not inside ashcan.
2. At step 5: When the count of plate not inside cabinet is above 0 and the count of closed cabinet is 0, please reduce the count of plate not inside cabinet.
3. At step 3: When the count of teapot not on stove is above 0, please reduce the count of teapot not on stove.
4. At step 9: When the count of plate not inside cabinet is above 0 and the count of closed cabinet is 0, please reduce the count of plate not inside cabinet.
5. At step 8: When the count of chip and oatmeal sugar and vegetable oil not inside cabinet is above 0 and the count of closed cabinet is 0, please reduce the count of chip and oatmeal sugar and vegetable oil not inside cabinet.

Table 15: Sample generated instructions across granularity levels.