FineVLA: Fine-Grained Instruction Alignment for Steerable Vision-Language-Action Policies

Xintong Hu[*1] Xuhong Huang[*1] JINYU ZHANG[1] Yutong Yao[1]
Yuchong Sun[2] Qiuyue Wang[2] Mingsheng Li[2] Yitao Liu[1]
Yixuan Chen[1] Yingming Zheng[1] Shuai Bai[2] Tao Yu[1†]
[1]XLANG Lab, The University of Hong Kong [2]Qwen Team, Alibaba Inc.

Project Page: https://finevla.xlang.ai/

Abstract

Vision-Language-Action (VLA) models are increasingly expected to not only complete robot tasks, but also follow human instructions about *how* those tasks should be executed. However, existing robot datasets usually pair trajectories with coarse goal-level language, leaving execution-critical details such as active arm, approach direction, and contact region unspecified. This limits steerable policy learning and robotic video understanding. We introduce **FineVLA**, an open framework for action-aligned fine-grained VLA supervision. The framework includes: (1) a data construction tool that unifies 972,247 trajectories across 85K tasks from 10 open-source robot datasets and builds **FineVLA-Data**, a human-verified dataset of 47,159 fine-grained trajectories; (2) a held-out benchmark with 500 videos, 11,631 atomic facts, and 1,030 VQA questions; (3) a robotics-specialized VLM annotator for scalable fine-grained annotation; and (4) a steerable **VLA policy** trained with controlled mixtures of fine-grained and raw goal-level instructions. Our policy experiments yield three findings. First, fine-grained supervision does not sacrifice goal-level task success: fine-grained-only (FG-only) improves over raw-instruction-only (Raw-only) by +1.4 to +8.1 success-rate points across architecture and data-scale settings. Second, fine-grained and raw instructions are complementary: performance follows a consistent inverted-U trend, peaking around FG : Raw = 1 : 2 to 1 : 1. The strongest mixed setting reaches **86.8%/82.5%** in RoboTwin simulation and **62.7**/100 in real-world dual-arm manipulation, compared with 49.9 for Raw-only. Third, fine-grained supervision directly improves steerable control by increasing compliance with language-specified execution factors: in real-world evaluation, the largest gains over Raw-only appear on pose (+23), color (+18), and approach direction (+18)—factors where goal-level instructions provide no guidance. Overall, fine-grained language should augment goal-level instructions: specifying *how* to execute alongside *what* to achieve.

1 Introduction

Vision-Language-Action (VLA) models are moving from task-level robot control toward policies that can be *steered* by human instructions. In this work, we use *steerability* to mean the ability to execute the same high-level goal in different ways according to user-specified execution constraints, such as which arm to use, which target object to manipulate, how to approach it, where to make contact, which motion or rotation direction to follow, and what final configuration to achieve. Recent robot

[*] Equal contribution. [†] Corresponding authors.

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

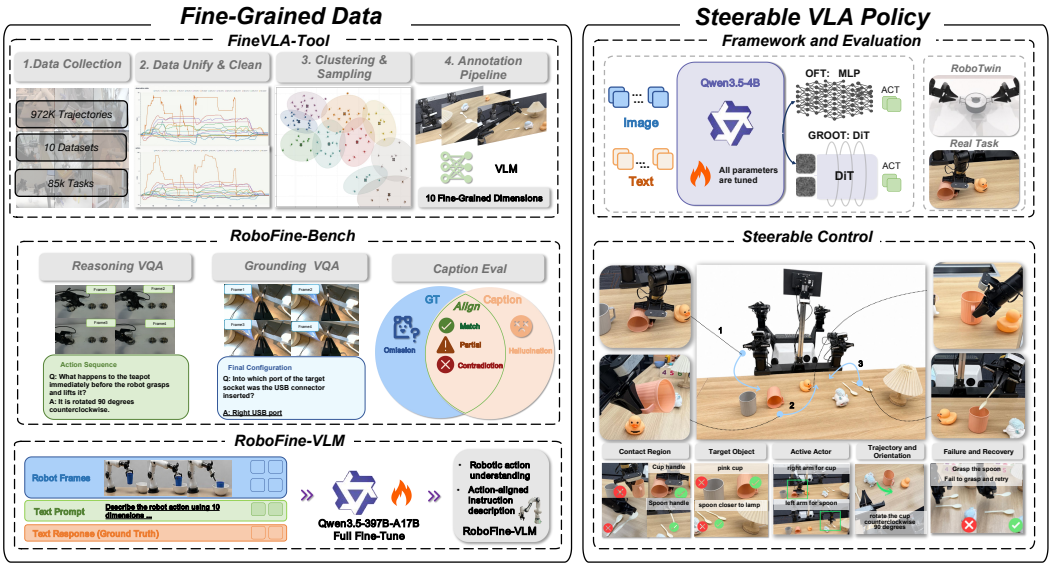


Figure 1: **Overview of FineVLA.** FineVLA-Tool constructs human-verified, action-aligned instructions from heterogeneous robot data. RoboFine-Bench evaluates execution-level understanding, and RoboFine-VLM supports scalable annotation. FineVLA-Policy tests fine-grained/raw instruction mixtures under two action decoders in simulation and real-world manipulation, including actor, contact, trajectory, and recovery factors.

foundation models such as $\pi_{0.7}$, LingBot-VLA, GR00T N1.7, and GEN-1 (1, 2, 3, 4) suggest that future robot policies should not only infer *what* task to complete, but also follow instructions about *how* the task should be performed.

However, building open, steerable VLA systems remains challenging for three reasons. **(i) Heterogeneous data and missing fine-grained annotation infrastructure.** Existing open-source robot datasets use diverse action and state representations that cannot be directly unified (5, 6). Within the same task, demonstrations are heavily redundant, and most trajectories carry only a single goal-level description (e.g., "pick up the cup") while the execution process—actor choice, approach direction, contact region, motion path, and state transitions—remains unspecified. The problem is not only that labels are coarse, but that there is still no open infrastructure for producing action-aligned fine-grained supervision at scale. **(ii) Lack of benchmarks and scalable annotators for fine-grained robotic video understanding.** Although general video-language models and dense captioning methods have advanced video description, their captions often focus on scene appearance rather than action-relevant execution details such as contact regions, approach directions, and motion paths. Existing embodied benchmarks (7, 8, 9) mainly evaluate spatial reasoning or hand-object dynamics, but do not systematically measure whether VLMs capture process-level manipulation details. There is also a lack of open, robotics-specialized annotators for action-aligned fine-grained captions. **(iii) Unknown effectiveness and training recipe.** Even if fine-grained data were available, the community lacks systematic evidence on whether action-aligned instructions improve policy learning, and what mixture of fine-grained and goal-level supervision yields the best steerable control.

To address these challenges, we introduce **FineVLA**, a fully open-source framework for scaling fine-grained VLA data, robotic video understanding, and steerable VLA policies (Figure 1). The framework operationalizes this principle through four components, each targeting one of the gaps above. **(1) FineVLA-Tool + FineVLA-Data (Gap i).** FineVLA-Tool unifies 972,247 trajectories across 85K tasks from 10 open-source datasets, selects representative samples via dynamic time warping (DTW)-based clustering, and annotates them with process-level descriptions across ten fine-grained dimensions (Table 9). This produces **FineVLA-Data**, a human-verified corpus of 47,159 trajectories whose average instruction length increases 10.4× (from 9.3 to 96.8 words). **(2) RoboFine-Bench (Gap ii).** We curate RoboFine-Bench—500 videos with 11,631 human-reviewed atomic facts and 1,030 VQA questions spanning all ten fine-grained dimensions, with complementary VQA and caption tracks. All benchmark trajectories are held out from both VLM fine-tuning and policy training, ensuring an independent evaluation. **(3) RoboFine-VLM (Gap ii).** We fine-tune Qwen3.5-

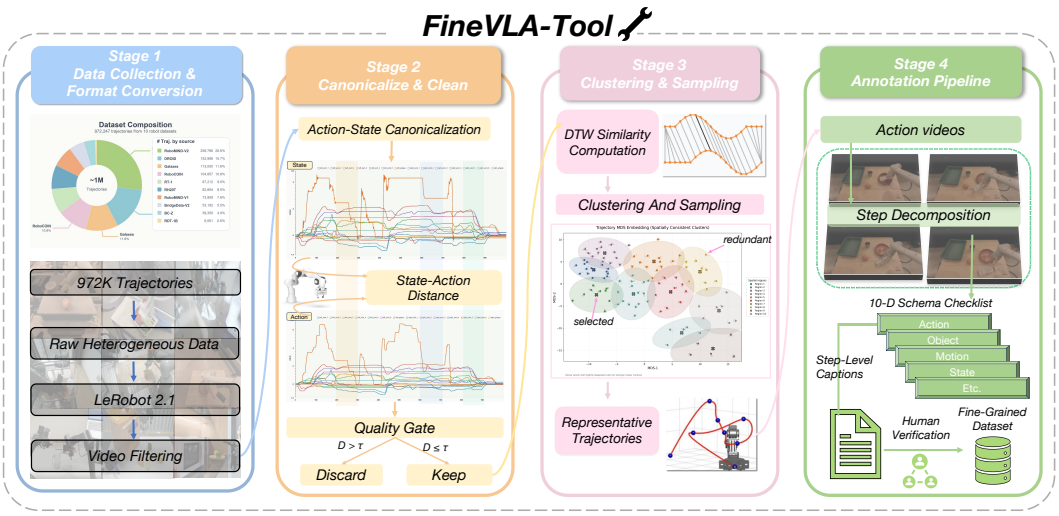


Figure 2: **Pipeline of FineVLA-Tool.** Four stages convert heterogeneous robot data into fine-grained supervision: format conversion and filtering; action/state canonicalization and quality control; DTW-based representative sampling; and ten-dimensional step annotation with human verification. The resulting data train the annotator and downstream policies.

397B-A17B (10) on FineVLA-Data to obtain RoboFine-VLM, a VLM specialized for robotic action understanding that serves as a scalable annotator for new trajectories. **(4) FineVLA-Policy + training recipe (Gap iii).** We train FineVLA-Policy under two action-decoding architectures (StarVLA-OFT and StarVLA-GR00T) and systematically vary the ratio between fine-grained (FG) and raw goal-level (Raw) instructions— sampling from fixed FG and Raw data pools while preserving the stored actions and visual observations—to study action-aligned supervision.

Our experiments establish three findings: FG-only supervision improves rather than sacrifices goal-level success; mixing FG and Raw produces an inverted-U trend with a favorable 1:2–1:1 region; and fine-grained language improves real-world execution-factor control. The best AlohaMix-OFT policy reaches **86.8%/82.5%** on RoboTwin Easy/Hard and **62.7**/100 on real-world tasks (49.9 for Raw-only), with the largest gains on pose, color, and approach direction. RoboFine-VLM reaches **68.2%** VQA accuracy and **82.2%** hard-setting caption score. We release the data pipeline, annotations, benchmark, model checkpoints, and training code to support open research on steerable VLA policies.

2 FineVLA Data: Construction, Benchmark, and Scalable Annotation

This section describes the data and annotation substrate of FineVLA. We first unify heterogeneous robot demonstrations and construct human-verified fine-grained action-aligned instructions. We then build a held-out benchmark to evaluate process-level robotic video understanding, and finally instantiate RoboFine-VLM as a scalable annotator for extending the same annotation schema to new trajectories.

FineVLA-Tool converts large-scale heterogeneous robot datasets into fine-grained, action-aligned instruction supervision (Figure 2). Its design addresses three practical bottlenecks in open robot data: (1) inconsistent action/state formats across datasets, (2) heavy redundancy among demonstrations of the same task, and (3) sparse, task-level instruction annotations. Starting from 972,247 trajectories across 10 source datasets, FineVLA-Tool produces FineVLA-Data, a human-verified corpus of 47,159 representative trajectories with fine-grained process-level supervision.

2.1 FineVLA-Tool: Canonicalization, Clustering, and Annotation

Data collection and format conversion (Figure 2, Stage 1). We aggregate 972,247 trajectories from 10 open-source datasets (11, 12, 13, 14, 15, 16, 17, 18, 6, 19), convert them to the LeRobot 2.1 format, and filter out invalid or degenerate recordings. The full per-dataset breakdown is in Appendix A.1.1.

Table 1: **FineVLA-Data statistics.** Fine-grained annotations dramatically increase instruction information density compared to original coarse instructions across all data sources.

Source	Trajectories	Steps	Avg. Words (Coarse)	Avg. Words (FG)	Density ↑
BridgeData-V2	4,958	21,554	10.1	61.7	6.1×
BC-Z	1,513	5,313	5.2	51.2	9.8×
RT-1	5,232	22,023	6.8	61.4	9.1×
Galaxea	2,834	18,484	4.7	219.9	47.1×
RoboMIND-V1	4,605	20,341	8.6	72.8	8.5×
RoboMIND-V2	7,119	39,166	6.6	98.8	14.9×
RoboCOIN	8,513	43,926	16.1	122.6	7.6×
RH20T	1,387	5,560	7.9	92.1	11.7×
RDT	1,275	8,437	16.9	114.0	6.7×
DROID	9,723	35,802	8.0	90.9	11.3×
Total	**47,159**	**220,606**	9.3	96.8	10.4×

Action-state canonicalization and cleaning(Figure 2, Stage 2). Across datasets, action and state values differ in temporal reference (absolute, relative, or delta) and kinematic representation (joint space vs. end-effector space with varied rotation encodings). We canonicalize all trajectories to absolute coordinates with normalized quaternion rotations, then remove trajectories whose action-state DTW distance exceeds a dataset-specific threshold, filtering out corrupted logs or inconsistent control conventions. Details and conversion examples are in Appendix A.1.2.

Trajectory clustering and representative sampling(Figure 2, Stage 3). Open robot datasets contain many near-duplicate demonstrations within the same task, often differing only in execution speed or minor spatial offsets. To maximize annotation diversity under a fixed budget, we cluster trajectories within each task using DTW over canonicalized action sequences, followed by hierarchical clustering on the resulting distance matrix. We then select high-quality representatives from each cluster according to cluster size and trajectory quality. This reduces 972,247 raw trajectories to 47,159 representative samples while preserving diversity in manipulation strategies, object interactions, and motion patterns. Details of the DTW formulation, action-space normalization, frame costs, and clustering procedure are provided in Appendix A.1.4.

Fine-grained multi-aspect annotation(Figure 2, Stage 4). Each selected trajectory is annotated with a ten-dimensional schema capturing the control-relevant factors that goal-level instructions omit: *action sequence*, *active actor*, *target object*, *initial configuration*, *final configuration*, *contact and approach*, *trajectory and orientation*, *object interaction*, *failure and recovery*, and *body motion*. Detailed definitions and examples are provided in Appendix A.1.5, Table 9. Annotation proceeds in two phases: we first input sampled video frames from each trajectory into Qwen3.5-Plus (10), which decomposes the manipulation process into temporally ordered steps and fills structured slots for actor, target, contact region, motion path, and state change; human annotators then review the model-generated descriptions against the original video, correcting factual errors and verifying temporal alignment. The result is FineVLA-Data, a human-verified fine-grained instruction dataset for training RoboFine-VLM and downstream controllable VLA policies.

2.2 FineVLA-Data Statistics

Table 1 summarizes the statistics of FineVLA-Data. Fine-grained annotations dramatically increase instruction information density compared to original coarse instructions: the average word count per trajectory rises from 9.3 to 96.8, an approximately 10.4× increase, while covering 47 unique action verbs across all sources. Detailed source dataset statistics are reported in Appendix A.1.1, Table 6.

2.3 RoboFine-Bench: Fine-Grained Robotic Video Understanding Benchmark

We introduce RoboFine-Bench to evaluate whether VLMs capture execution-level details of robot manipulation. The benchmark contains 500 videos from 10 robot datasets, covering 32 embodiments, diverse camera views, and a wide range of manipulation tasks. Each trajectory is paired with human-reviewed step-level annotations decomposed into 11,631 atomic facts across ten action-relevant dimensions (Table 9), with an average of 4.3 steps and 23.3 facts per sample. All 500 benchmark trajectories are strictly disjoint from both the RoboFine-VLM SFT training set and all policy-training

splits—no trajectory appears in both the 47,159 training samples and the benchmark. Appendix A.9, Figure 6, illustrates the benchmark statistics and structure.

RoboFine-Bench contains two complementary tracks. The **VQA track** (Figure 6, right bottom) evaluates discriminative understanding through 1,030 questions distributed across the same ten fine-grained dimensions used in annotation, which are aggregated into three reporting axes: **Entity and Scene Grounding**, **Action and Motion Understanding**, and **Interaction and State Reasoning** (Table 16 in Appendix A.3.2). Each model receives video frames and all questions for one sample in a single prompt, and answers are scored by deterministic matching against ground-truth labels. The **Caption track**(Figure 6, right top) evaluates generative understanding by asking models to produce action-aligned step-level fine-grained descriptions of the manipulation process. Generated captions are then judged by an LLM against pre-extracted ground-truth atomic facts, yielding per-fact alignment labels (match, partial, contradiction, omission, hallucination) that are aggregated into Consistency, Coverage, and Anti-Hallucination metrics. Two settings are evaluated: *easy*, where the original task instruction is provided, and *hard*, where the model must infer the process from visual observations alone. Full prompt templates and the evaluation protocol are provided in Appendix A.3.4.

2.4 RoboFine-VLM: Scalable Fine-Grained Annotator

While FineVLA-Data is human verified, scaling the same annotation schema to future robot trajectories still requires a robotics-specialized annotator. General-purpose VLMs often miss execution-level details such as contact regions, approach directions, grasp types, motion paths, and object state transitions, leading to substantial human correction cost.

We therefore fine-tune Qwen3.5-397B-A17B (10) on the human-verified FineVLA-Data to obtain RoboFine-VLM. Given a robot manipulation video, RoboFine-VLM generates temporally action-aligned step-level fine-grained descriptions covering the ten fine-grained dimensions (Table 9). The model serves as a scalable annotator for future data expansion; all policy experiments in this paper use FineVLA-Tool-generated and human-verified annotations, rather than RoboFine-VLM-generated labels. Full training details are provided in Appendix A.2.2; its annotation quality is evaluated on the held-out RoboFine-Bench in Section 4.2.

3 Training Fine-Grained VLA Policies

With human-verified fine-grained instructions fixed, we now study how they should be used to train steerable VLA policies. Our goal is not to propose a new policy architecture, but to isolate the effect of action-aligned language supervision. We therefore vary the pretraining sampling ratio between a fine-grained annotated subset and the full raw-instruction dataset, without changing the stored actions or visual observations of their shared trajectories.

3.1 FineVLA-Policy Architecture

We instantiate FineVLA-Policy under multiple action-decoding frameworks to isolate the effect of instruction supervision from architectural choices. Rather than proposing a new architecture, we adopt two existing frameworks implemented in the StarVLA codebase (20), both built on a shared Qwen3.5-4B vision-language backbone.

StarVLA-OFT attaches a lightweight MLP regression head that reads the hidden states of predefined action tokens and predicts continuous action chunks in parallel with an L1 objective, following OpenVLA-OFT. **StarVLA-GR00T** adopts a dual-system design where the VL backbone serves as System 2 for slow reasoning and a DiT-based flow-matching module serves as System 1 for continuous action generation, consistent with GR00T N1.5. Both variants produce multi-step action chunks and share the same visual observations and language inputs; only the action decoding differs. Using two architectures lets us verify that the benefits of fine-grained supervision are architecture-independent.

3.2 Training Data Mixtures

To isolate the effect of language supervision, we construct two parallel training datasets from the same source trajectories. The **FG dataset** contains the representative trajectories selected by FineVLA-

Tool, each paired with its fine-grained process-level instruction (1,275 trajectories for RDT; 5,872 for AlohaMix). The **Raw dataset** contains *all* source trajectories, each paired with its original goal-level instruction (6,061 for RDT; 86,662 for AlohaMix). AlohaMix is an ALOHA-compatible dual-arm mixture aggregated from RDT, RoboCOIN, RoboMIND-V1.0, and RoboMIND-V2.0, containing 86,662 episodes across 598 tasks (Table 21 in Appendix A.4.2). We restrict the mixture to a single embodiment class to avoid cross-embodiment confounds. Trajectories that appear in both datasets share identical action labels and visual observations; only the paired language instruction differs.

The FG:Raw ratio controls the probability of drawing from each dataset at every pretraining step, and therefore determines the relative number of training iterations that use fine-grained versus raw instructions. For example, FG:Raw = 2:1 means the FG dataset is sampled with twice the weight of the Raw dataset, so approximately two-thirds of training steps use a fine-grained instruction and one-third use a raw instruction. Under Raw-only, training draws exclusively from the Raw dataset; under FG-only, exclusively from the FG dataset.

We compare seven configurations: Raw-only, FG:Raw = 1:4, 1:2, 1:1, 2:1, 4:1, and FG-only. We study three (dataset, framework) combinations—RDT-OFT, RDT-GR00T, and AlohaMix-OFT—to control for architecture and data-scale effects. Within each setting, the FG and Raw pools and total pretraining budget are fixed. Because FG is a subset of Raw, changing the ratio also changes the sampling frequency of trajectories; it is not a per-trajectory instruction swap. Real-world SFT uses the same data and optimization settings for every resulting pretrained checkpoint.

4 Experiments

4.1 Experimental Setup

Evaluation benchmark. **(1) RoboFine-Bench.** RoboFine-Bench (Section 2.3) evaluates whether RoboFine-VLM captures fine-grained robotic action details. We compare RoboFine-VLM with five strong general-purpose VLMs on both VQA and captioning tracks. The VQA track reports overall and dimension-wise accuracy across the ten FineVLA dimensions, while the captioning track scores ordered step-level action descriptions using Consistency, Coverage, and Anti-Hallucination metrics.

(2) RoboTwin Simulation Evaluation. RoboTwin (21) evaluates simulated bimanual manipulation. We test the seven FG:Raw instruction ratios defined in Section 3.2 across three controlled policy settings: RDT-OFT, RDT-GR00T, and AlohaMix-OFT. Policies are evaluated on the official Easy and Hard splits, with 20 episodes per task.

(3) Real-world Steerability Evaluation. We design this self-contained real-world benchmark on a Cobot Magic dual-arm platform to measure language-conditioned controllability. Unlike broad robustness benchmarks that vary scenes, objects, or lighting, our suite isolates instruction following: for each instruction-sensitive task family, paired variants use the same object set and nearly identical initial scene layout while changing only one language-specified control factor. The suite includes two general manipulation tasks, five in-distribution instruction-sensitive task families (each comprising a paired variant) covering object color, object pose, approach direction, rotation direction, and active arm, and one out-of-distribution active-arm–target binding probe. Each task is evaluated over 10 trials in total, yielding 80 trials per policy setting across the eight tasks. Scores use a partial-completion rubric normalized to a 0–100 scale. Additional hardware and inference details are reported in Appendix A.6.1.

Training setup. We compare RDT-OFT and RDT-GR00T to study decoder effects, and RDT-OFT and AlohaMix-OFT to study pretraining-data effects. Each is pretrained for 100k steps with the specified FG:Raw sampling ratio, then evaluated through downstream fine-tuning on RoboTwin Clean and Random data (27,500 trajectories) and real-world demonstrations. Real-world SFT starts directly from the AlohaMix-OFT pretrained checkpoint for each ratio and uses the same 600 demonstrations (12 tasks, 50 each), instruction annotations, and optimization settings across variants. Thus, real-world comparisons vary the pretraining mixture, not the SFT data or its instruction ratio. Complete resource budgets, optimization settings, and sampling details appear in Appendix A.4, including Tables 20 and 22.

Table 2: **VQA benchmark results on RoboFine-Bench (%).** We report overall VQA accuracy together with all ten fine-grained capability dimensions. **AA**: Active Actor; **TO**: Target Object; **IC**: Initial Configuration; **AS**: Action Sequence; **C&A**: Contact & Approach; **T&O**: Trajectory & Orientation; **BM**: Body Motion; **OI**: Object Interaction; **FC**: Final Configuration; **F&R**: Failure & Recovery. Best value per column is **bold**.

Model	Overall↑	Gnd.↑			Act.↑				State↑		
		AA	TO	IC	AS	C&A	T&O	BM	OI	FC	F&R
Qwen3-VL-Plus	47.7	57.7	47.1	44.2	56.0	45.2	46.9	60.0	46.2	39.6	42.9
Qwen3.5-Plus	55.9	73.1	60.0	58.4	56.6	49.4	53.8	80.0	38.5	57.1	42.9
Doubao-Seed-2.0-Pro	58.5	63.5	55.3	53.2	62.4	49.7	58.8	70.0	53.8	64.3	50.0
Gemini-3.1-Pro	59.6	**84.6**	60.0	53.2	65.1	58.7	51.7	80.0	50.0	58.8	57.1
GPT-5.4	60.2	**84.6**	60.0	49.4	64.7	60.7	53.1	80.0	**61.5**	59.9	50.0
RoboFine-VLM (Ours)	**68.2**	82.7	**65.9**	**68.8**	**70.6**	**69.0**	**63.0**	**100.0**	**61.5**	**65.4**	**78.6**

Table 3: **Caption benchmark results on RoboFine-Bench (%).** We report caption quality under two settings: **easy**, where the original task instruction is provided, and **hard**, where the model must infer the manipulation process from vision alone. **Cons.**: Consistency; **Cov.**: Coverage; **A-Hal.**: Anti-Hallucination. Best value per column is **bold**.

Model	Easy				Hard			
	Overall↑	Cons.↑	Cov.↑	A-Hal.↑	Overall↑	Cons.↑	Cov.↑	A-Hal.↑
Qwen3-VL-Plus	75.4	75.2	58.2	92.8	64.4	67.4	54.3	71.6
Qwen3.5-Plus	76.6	75.3	59.1	95.5	72.4	71.0	55.1	91.2
Doubao-Seed-2.0-Pro	80.2	78.5	68.2	93.8	73.4	72.4	63.7	84.0
Gemini-3.1-Pro	80.1	79.9	62.7	**97.7**	75.9	75.7	58.5	93.4
GPT-5.4	81.4	79.5	72.1	92.5	78.0	73.8	66.8	93.4
RoboFine-VLM (Ours)	**83.2**	**82.1**	**72.7**	94.8	**82.2**	**80.4**	**71.5**	**94.8**

4.2 RoboFine-Bench Results

Tables 2 and 3 compare RoboFine-VLM with five strong general-purpose VLMs on RoboFine-Bench. A more detailed analysis of the benchmark results is provided in Appendix A.3.8.

VQA results. RoboFine-VLM achieves **68.2%** overall accuracy, outperforming the strongest general-purpose baseline, GPT-5.4, by **8.0** absolute points. The largest gain appears on Action and Motion Understanding (**75.7%** vs. **64.6%**), indicating that fine-grained supervision improves execution-level reasoning beyond scene recognition. Compared with its base model Qwen3.5-Plus (i.e., Qwen3.5-397B-A17B), SFT on FineVLA-Data improves overall accuracy from **55.9%** to **68.2%**, with consistent gains across grounding, action, and state reasoning.

Caption results. RoboFine-VLM also leads the caption track. In the easy setting, where the task instruction is provided, it obtains the best Overall, Consistency, and Coverage scores. In the hard setting, where the model must infer the manipulation process from video alone, RoboFine-VLM ranks first on all four metrics and improves Overall from the strongest baseline score of **78.0%** to **82.2%**. This setting is especially important because it measures whether the model captures the execution process rather than relying on task-level language priors. Token and latency statistics are reported in Appendix A.3.7.

Benchmark validity. The caption ranking is robust to the choice of alignment judge: replacing GPT-5.4-Pro with Gemini-3.1-Pro preserves the same strongest model in both easy and hard settings, with only small changes in absolute scores (Appendix A.3.5). The automatic scores also align strongly with human preference. As shown in Appendix A.9, Figure 7, the correlation between benchmark Overall scores and the rankings from 10 human raters is high in both settings (easy: Pearson **0.937**, Spearman ρ **0.943**; hard: Pearson **0.922**, Spearman ρ **0.943**).

These results provide evidence that RoboFine-VLM can produce dense, action-aligned robotic descriptions, and that the proposed annotation schema captures execution-level manipulation details. Importantly, the fine-grained supervision used for policy training is produced by FineVLA-Tool with human verification, rather than by RoboFine-VLM. Thus, RoboFine-VLM is evaluated here as a scalable annotator for future data expansion.

Table 4: **RoboTwin simulation success rate (%).** We compare three training settings (RDT-OFT, RDT-GR00T, and AlohaMix-OFT) under seven FG:Raw instruction ratios. Easy/Hard follow the official RoboTwin splits. Best value per column is **bold**.

FG:Raw	RDT-OFT		RDT-GR00T		AlohaMix-OFT	
	Easy↑	Hard↑	Easy↑	Hard↑	Easy↑	Hard↑
Raw-only	61.5	60.0	55.1	53.4	71.8	71.4
FG : Raw = 1 : 4	68.2	66.5	58.2	55.7	75.3	74.3
FG : Raw = 1 : 2	**74.1**	72.1	61.7	60.9	82.8	78.6
FG : Raw = 1 : 1	73.9	**72.4**	**69.4**	**68.2**	**86.8**	**82.5**
FG : Raw = 2 : 1	70.4	68.3	65.9	63.1	80.9	79.3
FG : Raw = 4 : 1	68.6	67.5	64.9	63.2	79.5	78.5
FG-only	62.9	62.0	62.1	61.5	78.3	76.1

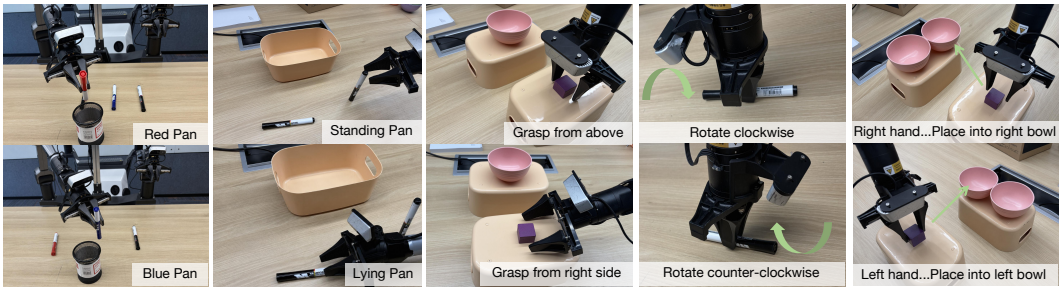


Figure 3: **Paired real-world evaluation.** Each column shows one control factor under the same visual scene with two language variants. From left to right: Color (red/blue), Pose (standing/lying), Approach (above/side), Rotation (clockwise/counterclockwise), Arm (right/left).

4.3 RoboTwin Simulation Results

We evaluate on RoboTwin (21), a simulation benchmark for bimanual manipulation, and report success rate on its official **Easy** and **Hard** splits. Each policy is evaluated over 20 episodes per task and averaged to produce the per-split score. Table 4 reports results across three (dataset, framework) combinations: RDT-OFT, RDT-GR00T, and AlohaMix-OFT. Note that AlohaMix is approximately 13× larger than RDT in episode count, enabling a controlled study of data-scale effects.

Table 4 shows two main results. First, fine-grained supervision does not harm goal-level task success: FG-only improves over Raw-only across all evaluated settings, with gains of +1.4/+2.0 on RDT-OFT (Easy/Hard), +7.0/+8.1 on RDT-GR00T, and +6.5/+4.7 on AlohaMix-OFT. Second, fine-grained and raw instructions are complementary: as the FG proportion increases from 0% to 100%, success rate follows a consistent inverted-U trend across all three settings, peaking around FG : Raw = 1 : 2 to 1 : 1. The best setting, FG : Raw = 1 : 1, reaches **86.8%/82.5%** on AlohaMix-OFT Easy/Hard, a gain of +15.0/+11.1 over the Raw-only baseline (71.8%/71.4%). Both conclusions hold across all three (dataset, framework) combinations, regardless of action-decoding architecture (OFT vs. GR00T) and pretraining data scale (RDT vs. AlohaMix). We analyze this trend and its mechanism in Appendix A.10.2.

4.4 Real-World Steerability Results

We evaluate physical steerable control on our self-designed Real-world Steerability Suite (Figure 3). This benchmark is designed to isolate language-conditioned controllability: paired task variants share nearly the same visual scene but require different behaviors according to the instruction, such as choosing a different object color, object pose, approach direction, rotation direction, or active arm.

Table 5 shows two main results. First, fine-grained supervision improves steerable control—a conclusion that simulation benchmarks alone cannot provide. FG : Raw = 1 : 1 improves every instruction-sensitive factor over Raw-only: Color (22 → **40**), Pose (24 → **47**), Approach (60 → **78**), Rotate (76 → **86**), and Arm (60 → **64**). The largest gains appear on factors invisible to goal-level language—Pose (+23), Color and Approach (+18 each)—precisely the execution choices that raw

Table 5: **Real-world scores on a 100-point scale.** All models use StarVLA-OFT pretrained on AlohaMix; FG:Raw denotes the pretraining mixture. All variants use the same real-world SFT data and settings for 100k steps. Each trial is scored by manually checking ordered subgoals; a completed subgoal receives proportional credit, and the final score is normalized to 100. *Avg (ID)* averages the seven non-OOD tasks (two general + five instruction-sensitive); *Avg (All)* additionally includes the OOD probe (†). **Bold** indicates the best score per column. Task descriptions and control factors are listed in Table 25.

Supervision	In-Distribution Tasks							OOD	Average	
	Clean Table	Stack Block	Color	Pose	Approach	Rotate	Arm	L→R†	(ID)	(All)
Raw-only	72	35	22	24	60	76	60	0	49.9	43.6
FG : Raw = 1 : 4	76	36	28	32	65	79	61	0	53.9	47.1
FG : Raw = 1 : 2	79	39	36	**48**	76	**87**	63	5	61.1	54.1
FG : Raw = 1 : 1	**84**	**40**	**40**	47	**78**	86	**64**	**10**	**62.7**	**56.1**
FG : Raw = 2 : 1	80	38	34	42	72	83	62	5	58.7	52.0
FG : Raw = 4 : 1	74	37	31	43	72	83	62	5	57.4	50.9
FG-only	70	35	25	41	70	80	60	0	54.4	47.6

† $L \rightarrow R$: use left hand to place into right bowl; unseen actor-target combination (OOD compositional probe). Control factors are detailed in Appendix A.6.1.

instructions leave unspecified. Second, consistent with the simulation findings, fine-grained and raw instructions are complementary in the real world. The in-domain score follows a clear inverted-U trend across all seven settings ($49.9 \rightarrow 53.9 \rightarrow 61.1 \rightarrow$ **62.7** $\rightarrow 58.7 \rightarrow 57.4 \rightarrow 54.4$), peaking at FG : Raw = 1 : 1 (**62.7**/100), which outperforms both Raw-only (49.9) and FG-only (54.4). On the two general manipulation tasks, Clean Table ($72 \rightarrow$ **84**) and Stack Block ($35 \rightarrow$ **40**), mixed supervision also matches or exceeds Raw-only, indicating that process-level language does not interfere with routine execution. The OOD actor-target binding remains challenging, but the mixed model improves from 0 to 10/100, suggesting partial factor-level generalization. We analyze factor-level controllability and failure modes in Appendix A.10.4.

5 Limitations

Fine-grained supervision improves the tested control factors, but does not solve compositional generalization: the unseen actor–target probe remains at only 10/100. Failures include incorrect grounding (wrong object, arm, or target) and physical execution errors after correct grounding (unstable contact, incomplete rotation, or inaccurate placement). Human verification is still required, and physical validation is limited to a tabletop dual-arm platform. Caption Anti-Hallucination measures fabricated action steps only: the other nine dimensions are not exhaustively annotated, so unmatched details need not be false. Deployment also requires feasibility and safety checks. Extended discussion appears in Appendix A.10.

6 Related Work

VLA policy learning and sparse trajectory language. Recent VLA policies such as RT-2 (22), OpenVLA (23), π_0 (24), and Octo (25) leverage pretrained vision-language models and large demonstration datasets such as Open X-Embodiment (5), DROID (19), and BridgeData V2 (11). While these efforts substantially improve generalist policy learning, their paired language supervision remains sparse: each trajectory is annotated with a goal-level task name that specifies the desired outcome but omits execution details such as which arm to use, how to approach the object, or what motion path to follow.

Fine-grained supervision for manipulation. Several works enrich supervision beyond trajectory-level labels. Galaxea (14), RoboCOIN (17), and RoboInter (26) introduce subtask or hierarchical annotations; STEER (27) and PartInstruct (28) study low-level or part-level instruction following. These annotations are typically organized around stages, primitives, or object parts rather than full process-level descriptions. FineVLA instead provides process-level, action-aligned supervision

across a ten-dimensional schema that unifies actor choice, contact patterns, motion trajectories, state transitions, and recovery behavior—and uses this supervision consistently for data construction, VLM training, benchmark evaluation, and policy learning. Human2Robot (29) offers a promising complementary approach by modeling fine-grained human–robot alignment through conditional video generation, learning motion representations that support robot action prediction.

Robotic video understanding and scalable annotation. General video-language models such as Qwen3-VL (30) and Qwen3.5-Omni (31) provide strong foundations for video captioning, while embodied benchmarks such as RoboVQA (7), RoboBench (8), and HanDyVQA (9) evaluate spatial reasoning, affordances, and hand-object dynamics. Dense captioning methods such as Wolf (32), DIAL (33), and RoboAnnotatorX (34) further improve annotation scalability. However, general captions do not necessarily align with robot action. FineVLA closes this gap by connecting robotic video understanding directly to VLA policy learning: RoboFine-VLM generates action-aligned descriptions, RoboFine-Bench evaluates execution-level understanding, and FineVLA-Policy tests whether such supervision improves instruction-sensitive control.

Steerable robot foundation models. Recent robot foundation models increasingly emphasize instruction-steerable behavior, where policies should follow not only task goals but also execution-level constraints (1, 2, 3, 4). However, the data construction and evaluation infrastructure behind many such systems remains limited or closed. FineVLA complements these efforts by providing an open action-aligned annotation pipeline, a held-out benchmark, a scalable annotator, and a controlled policy-training study.

7 Conclusion

FineVLA connects human-verified action-aligned data, robotic video understanding, and steerable policy learning. Across the tested settings, fine-grained supervision improves goal-level success, complements raw instructions most effectively around FG:Raw=1:2–1:1, and improves execution-sensitive real-world control. These findings support augmenting goal-level language with descriptions of how actions should be performed. Our released tools, data, benchmark, and models support further study; robust compositional generalization and broader physical validation remain open challenges.

Acknowledgments and Disclosure of Funding

This work was supported by Alibaba Qwen Lab and XLANG Lab at The University of Hong Kong. The authors declare no competing financial interests beyond the affiliations and support disclosed above.

References

[1] Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities, 2026. URL https://arxiv.org/abs/2604.15483.

[2] 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, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. A pragmatic vla foundation model, 2026. URL https://arxiv.org/abs/2601.18692.

[3] NVIDIA. Nvidia isaac gr00t. https://github.com/NVIDIA/Isaac-GR00T, 2026. GitHub repository, accessed April 13, 2026.

[4] Generalist AI. Gen-1. https://generalistai.com/blog/apr-02-2026-GEN-1, 2026. Blog post, accessed April 2, 2026.

[5] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models, 2023. URL https://arxiv.org/abs/2310.08864.

[6] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation, 2025. URL https://arxiv.org/abs/2410.07864.

[7] Pierre Sermanet, Tianli Ding, Jeffrey Zhao, Fei Xia, Debidatta Dwibedi, Keerthana Gopalakrishnan, Christine Chan, Gabriel Dulac-Arnold, Sharath Maddineni, Nikhil J. Joshi, Pete Florence, Wei Han, Robert Baruch, Yao Lu, Suvir Mirchandani, Peng Xu, Pannag Sanketi, Karol Hausman, Izhak Shafran, Brian Ichter, and Yuan Cao. RoboVQA: Multimodal long-horizon reasoning for robotics, 2023. URL https://arxiv.org/abs/2311.00899.

[8] Yulin Luo, Chun-Kai Fan, Menghang Dong, Jiayu Shi, Mengdi Zhao, Bo-Wen Zhang, Cheng Chi, Jiaming Liu, Gaole Dai, Rongyu Zhang, Ruichuan An, Kun Wu, Zhengping Che, Shaoxuan Xie, Guocai Yao, Zhongxia Zhao, Pengwei Wang, Guang Liu, Zhongyuan Wang, Tiejun Huang, and Shanghang Zhang. Robobench: A comprehensive evaluation benchmark for multimodal large language models as embodied brain, 2025. URL https://arxiv.org/abs/2510.17801.

[9] Masatoshi Tateno, Gido Kato, Hirokatsu Kataoka, Yoichi Sato, and Takuma Yagi. HanDyVQA: A video QA benchmark for fine-grained hand-object interaction dynamics, 2025. URL https://arxiv.org/abs/2512.00885.

[10] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[11] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, et al. BridgeData V2: A dataset for robot learning at scale. In *Proceedings of the Conference on Robot Learning*, 2023. URL https://arxiv.org/abs/2308.12952.

[12] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning, 2022. URL https://arxiv.org/abs/2202.02005.

[13] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. RT-1: Robotics transformer for real-world control at scale, 2022. URL https://arxiv.org/abs/2212.06817.

[14] Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, et al. Galaxea open-world dataset and G0 dual-system VLA model, 2025. URL https://arxiv.org/abs/2509.00576.

[15] Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, Shichao Fan, Xinhua Wang, Fei Liao, Zhen Zhao, Guangyu Li, Zhao Jin, Lecheng Wang, Jilei Mao, Ning Liu, Pei Ren, Qiang Zhang, Yaoxu Lyu, Mengzhen Liu, He Jingyang, Yulin Luo, Zeyu Gao, Chenxuan Li, Chenyang Gu, Yankai Fu, Di Wu, Xingyu Wang, Sixiang Chen, Zhenyu Wang, Pengju An, Siyuan Qian, Shanghang Zhang, and Jian Tang. RoboMIND: Benchmark on multi-embodiment intelligence normative data for robot manipulation. In *Robotics: Science and Systems XXI*, RSS2025. Robotics: Science and Systems Foundation, June 2025. doi: 10.15607/rss.2025.xxi.152. URL http://dx.doi.org/10.15607/RSS.2025.XXI.152.

[16] Chengkai Hou, Kun Wu, Jiaming Liu, Zhengping Che, et al. RoboMIND 2.0: A multimodal, bimanual mobile manipulation dataset for generalizable embodied intelligence, 2025. URL https://arxiv.org/abs/2512.24653.

[17] Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, et al. RoboCOIN: An open-sourced bimanual robotic data collection for integrated manipulation, 2025. URL https://arxiv.org/abs/2511.17441.

[18] Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot, 2023. URL https://arxiv.org/abs/2307.00595.

[19] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset, 2024. URL https://arxiv.org/abs/2403.12945.

[20] StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing, 2026. URL https://arxiv.org/abs/2604.05014.

[21] Yao Mu, Tianxing Chen, Shijia Peng, Zanxin Chen, Zeyu Gao, Yude Zou, Lunkai Lin, Zhiqiang Xie, and Ping Luo. RoboTwin: Dual-arm robot benchmark with generative digital twins, 2024. URL https://arxiv.org/abs/2409.02920.

[22] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL https://arxiv.org/abs/2307.15818.

[23] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

[24] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[25] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

[26] Hao Li, Ziqin Wang, Zi-han Ding, Shuai Yang, et al. RoboInter: A holistic intermediate representation suite towards robotic manipulation, 2026. URL https://arxiv.org/abs/2602.09973.

[27] Laura Smith, Alex Irpan, Montserrat Gonzalez Arenas, Sean Kirmani, Dmitry Kalashnikov, Dhruv Shah, and Ted Xiao. STEER: Flexible robotic manipulation via dense language grounding, 2024. URL https://arxiv.org/abs/2411.03409.

[28] Yifan Yin, Zhengtao Han, Shivam Aarya, Jianxin Wang, Shuhang Xu, Jiawei Peng, Angtian Wang, Alan Yuille, and Tianmin Shu. PartInstruct: Part-level instruction following for fine-grained robot manipulation, 2025. URL https://arxiv.org/abs/2505.21652.

[29] Sicheng Xie, Haidong Cao, Zejia Weng, Zhen Xing, Haoran Chen, Shiwei Shen, Jiaqi Leng, Zuxuan Wu, and Yu-Gang Jiang. Human2robot: Learning robot actions from paired human-robot videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 11078–11086, 2026.

[30] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

[31] Qwen Team. Qwen3.5-Omni technical report, 2026. URL https://arxiv.org/abs/2604.15804.

[32] Boyi Li, Ligeng Zhu, Ran Tian, Shuhan Tan, Yuxiao Chen, Yao Lu, Yin Cui, Sushant Veer, Max Ehrlich, Jonah Philion, Xinshuo Weng, Fuzhao Xue, Linxi Fan, Yuke Zhu, Jan Kautz, Andrew Tao, Ming-Yu Liu, Sanja Fidler, Boris Ivanovic, Trevor Darrell, Jitendra Malik, Song Han, and Marco Pavone. Wolf: Dense video captioning with a world summarization framework, 2025. URL https://arxiv.org/abs/2407.18908.

[33] Ted Xiao, Harris Chan, Pierre Sermanet, Ayzaan Wahid, Anthony Brohan, Karol Hausman, Sergey Levine, and Jonathan Tompson. Robotic skill acquisition via instruction augmentation with vision-language models. In *Robotics: Science and Systems*, 2023. URL https://arxiv.org/abs/2211.11736.

[34] Longxin Kou, Fei Ni, Yan Zheng, Peilong Han, Jinyi Liu, Haiqin Cui, Rui Liu, and Jianye Hao. RoboAnnotatorX: A comprehensive and universal annotation framework for accurate understanding of long-horizon robot demonstration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10353–10363, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Kou_RoboAnnotatorX_A_Comprehensive_and_Universal_Annotation_Framework_for_Accurate_Understanding_ICCV_2025_paper.html.

A Appendix

Contents

A.1 FineVLA-Tool Details . . . 15
A.1.1 Data Sources and Format Conversion . . . 15
A.1.2 Action-State Canonicalization . . . 16
A.1.3 Quality Filtering and DTW Consistency Check . . . 16
A.1.4 Action-Based Clustering and Representative Sampling . . . 17
A.1.5 Fine-Grained Annotation Schema . . . 18
A.1.6 Human-in-the-Loop Verification . . . 20
A.2 RoboFine-VLM Details . . . 20
A.2.1 SFT Data Format . . . 20
A.2.2 Model and Training Details . . . 20
A.2.3 Video Sampling Configuration . . . 21
A.2.4 SFT Prompt Templates . . . 21
A.2.5 Inference and Scalable Annotation . . . 23
A.3 RoboFine-Bench Details . . . 23
A.3.1 Benchmark Construction and Statistics . . . 23
A.3.2 VQA Track Details . . . 23
A.3.3 Caption Track Details . . . 25
A.3.4 Prompt Templates and Evaluation Protocol . . . 26
A.3.5 Judge Robustness . . . 28
A.3.6 Human Alignment Study . . . 28
A.3.7 Caption Cost, Token, and Latency . . . 29
A.3.8 Detailed Benchmark Results . . . 29
A.4 FineVLA-Policy Setup . . . 30
A.4.1 Policy Frameworks . . . 30
A.4.2 Pretraining Datasets and Configuration . . . 31
A.4.3 RoboTwin Fine-Tuning Data and Configuration . . . 31
A.4.4 Instruction Mixing Construction . . . 31
A.4.5 Additional Training Details . . . 31
A.5 RoboTwin Details and Additional Analysis . . . 32
A.5.1 RoboTwin Evaluation Protocol . . . 32
A.5.2 Full RoboTwin Results . . . 32
A.5.3 Mixing-Ratio Analysis . . . 33
A.5.4 Architecture and Scale Analysis . . . 33
A.6 Real-World Policy Details . . . 33
A.6.1 Robot Hardware and Setup . . . 33
A.6.2 Real-World Tasks . . . 34
A.6.3 Real-World Evaluation Protocol . . . 35

A.6.4 Subgoal Definitions for Partial Scoring 35
A.7 Additional Analysis 35
A.8 Reproducibility, Limitations, and Ethics 36
A.8.1 Reproducibility Checklist Support 36
A.8.2 Compute Resources 36
A.8.3 Limitations 36
A.8.4 Societal Impact and Safety 36
A.9 Benchmark Overview and Human Alignment 36
A.10 Extended Policy Analysis 38
A.10.1 Fine-Grained Supervision Does Not Sacrifice Goal-Level Success 38
A.10.2 Raw and Fine-Grained Supervision Are Complementary 38
A.10.3 Architecture and Data-Scale Effects 39
A.10.4 Fine-Grained Language Enables Factor-Level Steerable Control 39
A.10.5 Limitations 40

This appendix is organized to mirror the logic of the main paper and collects detailed methodology, extended analyses, and reproducibility information that support the claims in Sections 2–5.

A.1 FineVLA-Tool Details

This section provides implementation and annotation details for Section 2. It explains how heterogeneous robot trajectories are converted into a unified action-aligned supervision corpus and how human review is used to maintain annotation quality.

A.1.1 Data Sources and Format Conversion

This subsection supports the data-construction claim in Section 2 by listing the source datasets and the unified-format conversion step used before filtering, clustering, and annotation.

Table 6 summarizes the ten open-source datasets used by FineVLA-Tool before clustering and representative sampling. We convert all trajectories to a unified LeRobot 2.1-style format that standardizes RGB videos, robot states, action sequences, and task metadata across embodiments. This conversion is a prerequisite for consistent filtering, action-state canonicalization, and later cross-dataset annotation.

Table 6: **Source datasets used by FineVLA-Tool.** The table reports the ten datasets retained for fine-grained annotation after selecting the final source list used in this paper.

Dataset	# Trajectories	# Tasks
BridgeData-V2	53,192	19,974
BC-Z	39,350	104
RT-1	87,212	599
Galaxea	113,000	19,627
RoboMIND-V1	73,909	436
RoboMIND-V2	258,786	723
RoboCOIN	104,857	807
RH20T-RoboInter	82,894	147
RDT-1B	6,061	296
DROID-RoboInter	152,986	43,026
Total	972,247	85,739

A.1.2 Action-State Canonicalization

This subsection supports the canonicalization step in Section 2. Its role is to make state and action sequences comparable across datasets that use different temporal conventions and different kinematic parameterizations.

Across datasets, action and state annotations differ in both temporal reference and kinematic representation. We organize these differences along two axes: (i) temporal convention, distinguishing absolute quantities, deltas relative to current state, and offsets relative to the first frame; and (ii) kinematic convention, distinguishing joint-space signals from end-effector-space signals with multiple rotation encodings. We standardize EEF rotations to quaternions in `xyzw` order.

Table 7: **Canonicalization tokens and their semantics.** Prefixes define temporal reference, while suffixes define the parameterization of robot state or action variables.

Token	Meaning	Notes
`abs`	Absolute value in a global/world frame	Used by all states
`delta`	Increment relative to the current state	Action only
`rel`	Offset relative to the first frame	Action only
`joint`	Joint / gripper / hand coordinates	Non-EEF modalities
`rotvec`	Rotation vector (axis-angle)	3D rotation code
`quat`	Quaternion in `xyzw` order	Canonical quaternion form
`wxyz`	Quaternion in `wxyz` order	Scalar-first order
`euler`	Euler angles in XYZ order	3D rotation code

For non-EEF variables such as joints, grippers, or hand states, the state is always treated as an absolute quantity. In contrast, the raw action may be stored as an absolute command or a delta command, and is therefore converted into one of `abs_joint`, `delta_joint`, or `rel_joint`. For EEF variables, each pose is represented as 3D position plus a rotation code; the state remains absolute, while the action may use any of the three temporal-reference prefixes.

Table 8: **Canonicalization rules across modality types.** State and action variables admit different canonical type sets for non-EEF and EEF modalities.

Field type	State options	Action options
Non-EEF	`abs_joint`	`abs_joint`, `delta_joint`, `rel_joint`
EEF (left/right end effector)	`abs_rotvec`, `abs_quat`, `abs_wxyz`, `abs_euler`	All 12 combinations formed by {`abs`, `delta`, `rel`} $\times$ {`rotvec`, `quat`, `wxyz`, `euler`}
Count summary	5 state tags overall	15 action tags overall

A minimal example is as follows. Let $s_t^{\text{joint}} \in \mathbb{R}^d$ be the current joint state and let the raw action be a delta command $\Delta a_t^{\text{joint}}$. The absolute next-state target is

$$\hat{s}_{t+1}^{\text{joint}} = s_t^{\text{joint}} + \Delta a_t^{\text{joint}},$$

while the relative action with respect to the first frame is

$$a_{t+1,\text{rel}}^{\text{joint}} = \hat{s}_{t+1}^{\text{joint}} - s_1^{\text{joint}}.$$

For EEF trajectories, we first convert the raw orientation code to a canonical `xyzw` quaternion, compose delta poses with the current absolute pose if necessary, and only then derive the desired absolute, delta, or first-frame relative action target.

A.1.3 Quality Filtering and DTW Consistency Check

This subsection supports the cleaning step in Section 2. It explains how we remove invalid videos and trajectories whose actions and states are inconsistent after canonicalization.

We first perform video-level filtering to remove trajectories with invalid or missing videos, extremely short duration, large black-frame segments, or other obvious recording failures. We then apply

action-state consistency filtering on canonicalized trajectories. Intuitively, a valid demonstration should exhibit a state evolution that is compatible with the recorded action sequence after both have been converted into a common representation. We therefore measure trajectory-level consistency using DTW and reject samples whose action-state DTW distance exceeds a dataset-specific threshold. This stage removes corrupted logs, mismatched control conventions, and trajectories whose recorded actions do not explain the observed state change.

A.1.4 Action-Based Clustering and Representative Sampling

This subsection provides full implementation details for the representative-sampling step described in Section 2. The goal is to reduce redundancy in large robot corpora while preserving genuinely distinct execution patterns, so that a fixed annotation budget covers the widest possible range of manipulation strategies.

Pipeline overview. The clustering pipeline proceeds in four stages:

1. **Canonicalization.** All trajectories within a task are converted to their canonical action representation (joint-space or EEF-space with quaternion rotations) following the procedure in Appendix A.1.2.
2. **Pairwise DTW distance computation.** For each pair of trajectories within a task, we compute the DTW distance using a representation-specific frame cost function (defined below).
3. **Hierarchical clustering.** Agglomerative clustering with average linkage is applied to the pairwise distance matrix, and the number of clusters is determined automatically via the largest relative gap in merge heights.
4. **Representative selection.** Two to three high-quality trajectories are selected from each cluster based on proximity to the cluster medoid and trajectory quality metrics (video integrity, action smoothness).

DTW formulation. Open robot datasets are highly redundant: many demonstrations differ only in speed, minor spatial offsets, or camera viewpoint, while expressing the same underlying action pattern. Dynamic Time Warping (DTW) handles temporal misalignment by finding the optimal warping path between two action sequences. Given two sequences $\mathbf{x}_{1:T}$ and $\mathbf{y}_{1:U}$, DTW minimizes the cumulative frame-level distance according to

$$D_{\mathrm{DTW}}(i,j) = c(\mathbf{x}_i, \mathbf{y}_j) + \min\big\{ D_{\mathrm{DTW}}(i-1, j-1),\ D_{\mathrm{DTW}}(i-1, j),\ D_{\mathrm{DTW}}(i, j-1) \big\}.$$

Frame cost function. The frame cost $c(\cdot,\cdot)$ depends on the action-space representation:

- **Joint-space** (`rot_type` $=$ `none`): All joint values are min-max normalized to $[0,1]$ per dimension across the task group. The frame cost is

$$c_{\text{joint}}(\mathbf{x}, \mathbf{y}) = w_{\text{pos}} \cdot \|\mathbf{j}_x - \mathbf{j}_y\|_2 + w_{\text{grip}} \cdot |g_x - g_y|,$$

  where $\mathbf{j}$ denotes the normalized joint vector and g the gripper state.
- **EEF-space** (quaternion or Euler): The frame cost combines position, rotation, and gripper terms:

$$c_{\text{eef}}(\mathbf{x}, \mathbf{y}) = w_{\text{pos}} \cdot \|\mathbf{p}_x - \mathbf{p}_y\|_2 + w_{\text{rot}} \cdot d_{\text{geo}}(\mathbf{q}_x, \mathbf{q}_y) + w_{\text{grip}} \cdot |g_x - g_y|,$$

  where $\mathbf{p}$ is the 3D position, $d_{\text{geo}}(\mathbf{q}_x, \mathbf{q}_y) = 2\arccos(|\mathbf{q}_x \cdot \mathbf{q}_y|)$ is the quaternion geodesic distance (handling $\mathbf{q} \equiv -\mathbf{q}$), and g is the gripper state. For datasets using Euler angles, orientations are first converted to quaternions.

Default weights are $w_{\text{pos}} = 1.0$, $w_{\text{rot}} = 1.0$, $w_{\text{grip}} = 100.0$. The high gripper weight ensures that gripper open/close transitions—which are critical for distinguishing manipulation strategies—are not overwhelmed by continuous motion differences.

Pairwise distance computation. Since each task typically contains 100–200 trajectories, computing the full $N \times N$ pairwise DTW distance matrix is tractable ($\sim$5k–20k pairs per task). Each DTW distance is normalized by the warping path length to account for differences in trajectory duration. Computation is parallelized across CPU cores.

Hierarchical clustering and representative selection. We apply agglomerative hierarchical clustering (average linkage) on the DTW distance matrix. The number of clusters k is selected automatically by identifying the largest relative gap in the dendrogram merge heights. We then select two to three representative trajectories per cluster according to cluster size and trajectory quality (proximity to the cluster medoid). Applying this procedure to 972,247 raw demonstrations yields 47,159 representative trajectories for fine-grained annotation. This greatly reduces annotation cost while preserving diversity in manipulation strategy, object interaction, and motion pattern.

Figure 4 shows two task-level clustering examples. In both cases, the DTW distance matrix exhibits clear block structure, and the corresponding MDS embedding separates execution modes that differ in gripper timing, contact duration, or end-effector path. This is the key reason clustering improves annotation efficiency: a single fine-grained instruction can cover multiple redundant demonstrations that share the same execution pattern.

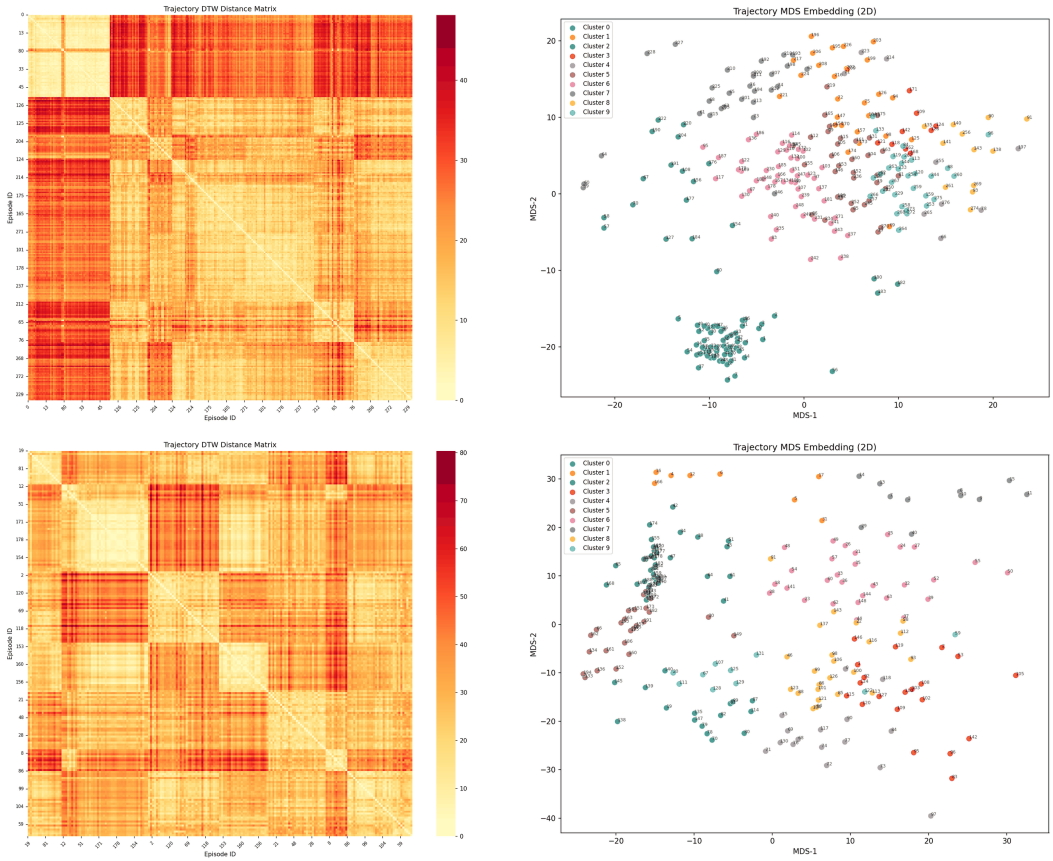


Figure 4: **Qualitative examples of DTW-based trajectory clustering in FineVLA-Tool.** For each task, the left panel shows the pairwise DTW distance matrix and the right panel shows a 2D MDS embedding of the same distances. Clear cluster structure indicates that trajectories with similar manipulation dynamics are grouped together, while differences in gripper timing and end-effector motion patterns are separated into different clusters. This allows one fine-grained instruction to cover multiple redundant demonstrations with the same execution pattern, substantially reducing annotation cost while improving data quality.

A.1.5 Fine-Grained Annotation Schema

This subsection supports the annotation schema referenced throughout the main paper. It defines the ten dimensions used both for data annotation and for benchmark construction.

Table 9: **Fine-grained annotation schema used by FineVLA-Tool.** Each representative trajectory in FineVLA-Data is annotated along ten control-relevant dimensions. The same schema is also used by RoboFine-Bench to construct VQA questions and caption-level atomic facts.

Dimension	What it captures	Typical slots	Examples
Action sequence	The ordered manipulation primitives and gripper states that define the execution process. Captures which actions occur and their temporal order.	Primitive action; gripper state; step order; action transition.	Open gripper $\rightarrow$ approach mug $\rightarrow$ grasp handle $\rightarrow$ lift $\rightarrow$ place on tray $\rightarrow$ release.
Active actor	The robot component or body part that performs the action. Important for bimanual manipulation and multi-effector platforms.	Actor identity; left/right hand; both hands; gripper; finger; arm; end-effector.	The right gripper picks up the cup. Both hands lift the container together.
Target object	The object directly manipulated, with visual or spatial attributes to disambiguate among similar objects.	Object category; color; material; shape; size; spatial reference.	The red cup on the left. The small metal spoon inside the bowl.
Initial config.	The pose, state, or spatial relation of the target object before the action begins.	Initial pose; workspace location; attachment state; relational state.	The bottle is standing upright on the table. The drawer is partially open.
Final config.	The pose, state, or spatial relation of the target object after the action is completed.	Final pose; final location; relational state; object state.	The cup ends up on the tray. The drawer is fully closed.
Contact & approach	How the robot approaches the target and where physical contact occurs. Captures control-relevant interaction geometry.	Contact region; approach direction; grasp type; contact modality.	Grasp the mug by the handle from above. Push the drawer from its front panel.
Trajectory & orientation	The translational path, motion direction, path shape, and rotational behavior of the object or end-effector.	Translation direction; path shape; motion extent; rotation direction; rotation angle.	Lift upward. Move to the right along a straight path. Rotate clockwise $\sim 90^{\circ}$.
Object interaction	Incidental or relational effects on objects other than the target. Captures secondary contacts, collisions, or induced movements.	Interaction type; affected object; contact or collision; relational change.	Placing the book causes the stack to shift. The block brushes against its neighbor.
Failure & recovery	Visible execution failures and recovery behaviors, including retries, slippage, and fail-then-succeed patterns.	Failure type; recovery action; retry behavior; attempt count; slippage.	The robot first fails to grasp, then retries successfully. The object slips before being regrasped.
Body motion	Motion of the robot base, torso, camera, or other non-end-effector body parts that affects execution.	Base translation; base rotation; torso motion; camera motion.	The mobile base moves forward before the arm reaches the object.

A.1.6 Human-in-the-Loop Verification

This subsection supports the claim that FineVLA-Data is human verified. It summarizes the dimensions checked during review after the initial annotation or model-assisted draft is produced.

Human reviewers compare each step-level description against the corresponding video and verify both semantic correctness and temporal alignment. The review process focuses on the factors most critical for downstream control.

Table 10: **Verification dimensions in the human review stage.** These checks ensure that fine-grained instructions remain temporally aligned and do not introduce hallucinated events.

Verification dimension	What is checked
Temporal order	Whether step order matches the observed execution timeline
Object identity	Whether the manipulated object is correctly identified and disambiguated
Actor identity	Whether the responsible arm / gripper / body part is correctly assigned
Contact region	Whether grasp/contact location and approach side are faithful to the video
Motion direction	Whether translational and rotational directions are described correctly
State transition	Whether initial and final states match the observed change
Hallucinated events	Whether the instruction introduces unsupported actions or recoveries

A.2 RoboFine-VLM Details

This section provides supporting details for the RoboFine-VLM component in Section 2.4. It explains the training data format, the known model setup, and how the resulting model is used as a scalable annotator.

A.2.1 SFT Data Format

This subsection supports the supervised fine-tuning setup described in Section 2.4. Each training sample pairs a robot manipulation video with a prompt asking for a temporally ordered fine-grained action description.

Table 11: **Supervised fine-tuning sample format for RoboFine-VLM.** Each sample maps a robot video to a step-level, action-aligned description derived from FineVLA-Tool annotations.

Field	Description
Input video	Multi-frame robot manipulation video sampled from a trajectory
Prompt	Instruction asking the model to describe the manipulation process step by step
Output target	Temporally ordered fine-grained action description aligned with FineVLA-Tool annotations
Supervision source	Human-verified video-instruction pairs from FineVLA-Data

A.2.2 Model and Training Details

This subsection supports the model description in Section 2.4.

Table 12: **RoboFine-VLM training configuration.**

Item	Value
Base model	Qwen3.5-397B-A17B
Fine-tuning strategy	Full parameter fine-tuning
Training objective	Supervised fine-tuning for step-level robotic action understanding
Training corpus	FineVLA-Tool-generated and human-verified video-instruction pairs
Training sample count	47,159 representative trajectories / video-instruction pairs
Output format	Ordered fine-grained action description
GPUs	256×H200
Global batch size	512
Epochs	9
Training steps	903
Optimizer	AdamW
Learning rate	7e-6, decayed to 7e-7
LR schedule	Linear decay
Wall-clock training time	∼40 hours (1 day 16 hours)
Memory per GPU	∼105 GB

A.2.3 Video Sampling Configuration

This subsection details how raw robot videos are sampled into frame sequences for SFT. We use two configurations depending on the number of camera views available in each trajectory.

Table 13: **Video sampling configuration for RoboFine-VLM SFT.** Multi-view trajectories concatenate frames from all available cameras, while single-view trajectories allow longer temporal context per video.

Parameter	Multi-view	Single-view
Frame sampling rate	4 fps	4 fps
Max frames per video	512	1024
Max tokens per video	76k	224k

A.2.4 SFT Prompt Templates

This subsection provides the system prompts used during supervised fine-tuning of RoboFine-VLM. We use two prompt variants depending on the number of camera views.

Single-view prompt. The following prompt is used for trajectories with a single camera view:

```
You are a robot manipulation video annotator.

Task: Decompose the video into an ordered sequence of fine-grained steps. Each
step must describe exactly one visible physical movement by the robot.

For each step, include the following when visually confirmed:
- action and gripper state (e.g., grasp, pick up, place, push, rotate; open →
close, release, maintain grasp)
- active actor (left hand, right hand, both hands, gripper, finger)
- target object using the task instruction name; disambiguate similar objects by
position or appearance (e.g., the red cup on the left, the front-most block)
- contact region and approach direction (e.g., by the handle from above, at the
top edge from the right)
- object initial state or location if relevant (e.g., upright, lying flat,
inside the container)
- trajectory and orientation change (e.g., move forward, lift up, rotate
clockwise/counter-clockwise)
```

```
- final placement location and final pose if the object is repositioned
- interaction with other objects (e.g., collision, dragging, tipping,
displacement)
- failures, retries, slippage, or fail-then-succeed patterns
- body motion such as base, torso, or camera movement

Rules:
- One step = one physical movement.
- Describe only visible facts.  Do not infer hidden actions, intent, or occluded
contact.
- Use the robot's egocentric frame:  left, right, forward, backward, up, down.
- Keep descriptions short, action-oriented, and literal.
- If a detail is unclear or not visible, omit it.
- Mention dual-arm coordination only when both arms are active; label it as
stabilize-and-act, simultaneous, sequential, or handoff.

Output JSON only:
{  "Step1":  "...", "Step2":  "...", "StepN":  "..."   }
```

Multi-view prompt. The following prompt is used for trajectories with three synchronized camera views (one main view and two wrist views):

```
You are a robot manipulation video annotator.

Task:  Analyze three synchronized views and decompose the robot's behavior into
an ordered sequence of atomic physical movements.

View use:
- Main View:  primary view for all annotation decisions, including step
boundaries, action order, active arm, global motion, object relations, and final
results.
- Wrist Views (left/right):  auxiliary only.  Use them only to refine local
details such as gripper state, contact region, approach direction, or slight
slip.
- If Wrist Views conflict with the Main View, follow the Main View.

Step requirements:
For each step, include the following when visually confirmed:
- action and gripper state (e.g., grasp, pick up, place, push, rotate; open →
close, release, maintain grasp)
- active actor (left hand, right hand, both hands, gripper, finger)
- target object using the task instruction name; disambiguate similar objects by
position or appearance (e.g., the red cup on the left, the front-most block)
- contact region and approach direction (e.g., by the handle from above, at the
top edge from the right)
- object initial state or location if relevant (e.g., upright, lying flat,
inside the container)
- trajectory and orientation change (e.g., move forward, lift up, rotate
clockwise/counter-clockwise)
- final placement location and final pose if the object is repositioned
- interaction with other objects (e.g., collision, dragging, tipping,
displacement)
- failures, retries, slippage, or fail-then-succeed patterns
- body motion such as base, torso, or camera movement

Rules:
- One step = one physical movement.
- Start a new step when the robot changes primitive action, target, or contact
state.
- Describe only visible facts.  Do not infer hidden actions, intent, or occluded
```

```
contact.
- If a detail is unclear or not visually confirmed, omit it.
- Use the robot's egocentric frame:  left, right, forward, backward, up, down.
- Keep descriptions concise, action-oriented, and literal.
- Mention dual-arm coordination only when both arms actively contribute to the
same manipulation event; use stabilize-and-act, simultaneous, or handoff when
applicable.

Output JSON only:
{  "Step1":  "...", "Step2":  "...", "StepN":  "..."   }
```

A.2.5 Inference and Scalable Annotation

This subsection supports the scalable-annotation claim in Section 2.4. After fine-tuning, RoboFine-VLM serves both as an evaluation model on RoboFine-Bench and as a scalable annotator for new robot trajectories. Given a video, the model generates a temporally ordered action description in the same semantic space as the FineVLA-Tool annotations. These auto-generated descriptions can then be post-processed and human-verified, allowing fine-grained supervision to be extended beyond the manually reviewed subset without changing the schema.

A.3 RoboFine-Bench Details

This section provides supporting details for RoboFine-Bench, the benchmark component introduced in Section 2.3 and analyzed in Section 4.2. It covers benchmark construction, the VQA and caption tracks, evaluation robustness, and additional result analysis.

A.3.1 Benchmark Construction and Statistics

This subsection supports the benchmark-construction claim in Section 2.3. It summarizes the scale, sampling strategy, and granularity of the benchmark.

Sampling strategy. We sample exactly 50 trajectories from each of the 10 source datasets, yielding 500 benchmark videos in total. This uniform allocation ensures that no single dataset dominates the benchmark and that all 10 data sources contribute equally to the evaluation. The 500 benchmark trajectories are used exclusively for evaluation; RoboFine-Bench contains no training split, and all benchmark videos are disjoint from the RoboFine-VLM SFT training set (see Section 2.3).

Table 14: **Summary statistics of RoboFine-Bench.** The benchmark is built to test fine-grained robotic action understanding rather than only task-level recognition.

Statistic	Value
Videos	500 (50 per dataset $\times$ 10 datasets)
Source datasets	10
Robot embodiments	32
Atomic facts	11,631
Avg. annotated steps per sample	4.3
Avg. atomic facts per sample	23.3
VQA questions	1,030

A.3.2 VQA Track Details

This subsection supports the VQA track of Section 2.3. It explains how questions are constructed, how ten annotation dimensions are aggregated into three reporting axes, and how answers are scored.

We construct 1,030 questions that probe execution-level manipulation details across the same ten dimensions used in the annotation schema. The question counts per dimension are given in Table 15. For reporting, these dimensions are aggregated into three higher-level axes: Entity and Scene Grounding, Action and Motion Understanding, and Interaction and State Reasoning.

Table 15: **VQA question distribution across capability dimensions.** This table supports the scale and coverage of the VQA track.

Dimension	# Questions
Action sequence	218
Active actor	52
Body motion	10
Contact and approach	155
Failure and recovery	14
Final configuration	182
Initial configuration	77
Object interaction	26
Target object	85
Trajectory and orientation	211
Total	1030

Table 16: **VQA dimension mapping.** The ten fine-grained annotation dimensions are grouped into three reporting axes for RoboFine-Bench VQA evaluation.

Axis	Abbr.	Constituent dimensions
Entity & Scene Grounding	Gnd.	Active actor, Target object, Initial configuration
Action & Motion Understanding	Act.	Action sequence, Contact & approach, Trajectory & orientation, Body motion
Interaction & State Reasoning	State	Object interaction, Final configuration, Failure & recovery
Overall	—	Average over all ten dimensions

Representative VQA prompts include action recognition, temporal ordering, object grounding, and state-change questions. Examples are shown below:

- **Action Recognition**: "What is the robot doing? (A) Grasping (B) Pushing (C) Lifting (D) Placing"
- **Temporal Ordering**: "Which happens first? (A) Gripper opens (B) Arm moves to cup (C) Cup lifted (D) Cup placed"
- **Object Interaction**: "Which object is the right gripper interacting with? (A) Red block (B) Blue cup (C) Green plate (D) None"
- **State Change**: "State of the drawer after action? (A) Fully open (B) Half open (C) Closed (D) Removed"

Answer scoring is deterministic. Multiple-choice questions are evaluated by option matching; yes/no questions are evaluated by normalized string comparison; and numeric questions are evaluated by value extraction.

VQA question generation prompt. The following prompt is used to generate VQA questions from human-reviewed ground-truth annotations. It operates in two modes: *conflict-based* (when model-generated steps disagree with GT) and *GT-only* (when the two are aligned), ensuring questions target both common errors and diverse fine-grained facts.

```
You are a fine-grained robotics video QA-set builder.

Input: A JSON array. Each sample contains: sample_id, fineGrainedSteps
(model-generated, may contain errors), GT (human-reviewed, always trustworthy).

Task Overview

Two modes of question generation:

Mode A -- Conflict-based QA (when GT and fineGrainedSteps conflict): Generate
questions targeting specific disagreements. The answer must always come from
```

```
GT. Facts that GT describes but fineGrainedSteps omits:  you MAY ask.  Facts
that fineGrainedSteps describes but GT omits:  do NOT ask.

Mode B -- GT-only QA (when GT and fineGrainedSteps are similar):  Generate
questions purely from GT content.  Randomly select 3-5 dimensions from the 13
dimensions below.  Do NOT always pick the same dimensions.

For every sample, generate 3-5 QA pairs.  At most 3 GT-only (Mode B) questions
per sample.

13 Capability Dimensions:

1. action_primitive -- fundamental action type (grasp, push, rotate, etc.)
2. actor_identity -- which arm/hand/gripper performs the action
3. object_recognition -- object category, color, material, shape, size
4. object_disambiguation -- distinguishing similar objects via spatial/attribute
   cues
5. contact_region -- specific part where gripper contacts the object
6. source_state_or_location -- initial state/position before manipulation
7. trajectory_and_orientation -- direction, path, or rotation during motion
8. placement_specification -- final target location or spatial relation
9. interaction_with_other_objects -- contact/disturbance of non-target objects
10. success_failure_retry -- whether the action succeeds, fails, or retries
11. gripper_state -- open/close/release state at a specific moment
12. temporal_order_and_step_boundary -- ordering of steps and boundaries
13. body_motion -- robot base/torso/camera movement


Dimension Balancing:  For Mode B, randomly select dimensions per sample.  Do NOT
ask two questions on the same dimension within one sample.  Across the batch,
aim for roughly equal coverage.

Answer Types:
-- multiple_choice:  4-8 mutually exclusive options, no ‘‘all/none of the above’’
-- yes_no:  answer exactly ‘‘yes’’ or ‘‘no’’
-- number:  concise Arabic numeral

Question Writing Rules:
-- Each question tests exactly ONE atomic fact
-- Must be answerable by watching the video
-- Do NOT ask broad questions (e.g., ‘‘What does the robot do?’’)
-- Do NOT use visually similar colors as distractors

Output:  Valid JSON array only.  Each item contains sample_id, status, qas
(3-5 items with question_id, mode, capability, answer_type, question, options,
answer, reference_text).
```

A.3.3 Caption Track Details

This subsection supports the caption track of Section 2.3. It defines the two evaluation settings, the atomic-fact representation, and the metrics used to score generated captions.

In the *easy* setting, the original task instruction is provided to the model. In the *hard* setting, the model must infer the manipulation process from visual observations alone. Human-reviewed annotations are converted into atomic facts, and each generated caption is aligned against this fact set using the labels *match*, *partial match*, *contradiction*, *omission*, and *hallucination*. We report three aspect-specific metrics computed from the alignment counts. Let M, P, C, and O denote the number of GT facts labeled as match, partial, contradiction, and omission, respectively. Define $A = M + P + C$ (the number of GT facts addressed by the caption) and $G = M + P + C + O$ (total GT facts). Let H be the number of hallucinated action events and S be the total number of action steps in the generated

caption. The metrics are:

$$\text{Consistency} = \frac{M + 0.5\,P}{A}, \tag{1}$$

$$\text{Coverage} = \frac{M + 0.5\,P}{G}, \tag{2}$$

$$\text{Anti-Hallucination} = 1 - \frac{H}{S}, \tag{3}$$

$$\text{Overall} = \frac{\text{Consistency} + \text{Coverage} + \text{Anti-Hallucination}}{3}. \tag{4}$$

Scope of the Anti-Hallucination metric. Anti-Hallucination is deliberately computed only over the *action_sequence* dimension: H counts fabricated *action events* and S is the number of action steps in the generated caption. We restrict hallucination scoring to action steps because this is the only dimension for which fabrication can be defined unambiguously. The ground-truth atomic actions are guaranteed to enumerate *all* actions executed in the trajectory, so any action event in the caption without a corresponding ground-truth action is genuinely fabricated. For the other nine dimensions this guarantee does not hold: the ground-truth atomic facts do not exhaustively enumerate every valid descriptive statement (e.g., appearance, spatial relations, or object-state details), so a caption statement absent from the ground-truth fact set may be a *correct supplementary observation* rather than a hallucination. Because these two cases cannot be reliably separated for the non-action dimensions, we report Anti-Hallucination only over action steps and treat this restricted scope as a limitation of the metric (Section 5).

Table 17: **Caption-track metric and alignment-label definitions.** This table clarifies how atomic-fact alignment is converted into the reported caption metrics.

Item	Definition
Consistency	Whether the facts stated in the generated caption are correct with respect to the ground truth
Coverage	How much of the ground-truth fact set is recovered by the generated caption
Anti-Hallucination	Whether the caption avoids fabricated actions or states unsupported by the ground truth
Match	Generated fact is correct and aligned to a ground-truth fact
Partial match	Generated fact overlaps with a ground-truth fact but is incomplete or slightly imprecise
Contradiction	Generated fact conflicts with the ground truth
Omission	Ground-truth fact is missing from the generated caption
Hallucination	Generated fact has no support in the ground truth

A.3.4 Prompt Templates and Evaluation Protocol

This subsection provides the exact prompts and evaluation pipelines used for RoboFine-Bench. All evaluated models (Qwen3-VL-Plus, Qwen3.5-Plus, Doubao-Seed-2.0-Pro, Gemini-3.1-Pro, GPT-5.4, and RoboFine-VLM) receive the same visual input format, prompting protocol, and sampling budget within each track.

VQA evaluation pipeline. For each benchmark sample, multi-view video frames are extracted and labeled (e.g., `[View: head_rgb]`, `[View: left_wrist_rgb]`). All questions for one sample are batched into a single prompt. The model is instructed to return structured JSON answers. Multiple-choice options are randomly shuffled per question (seeded by question ID) to prevent positional bias. Answers are scored deterministically: yes/no questions by normalized string comparison, number questions by value extraction, and multiple-choice questions by option-letter matching.

The VQA system prompt is:

```
You are an expert robot manipulation video analyst.  You will be shown frames
from a robot manipulation video with one or more camera views.  Each view is
labeled (e.g., [View:  head_rgb], [View:  left_wrist_rgb]).  Use information
from ALL views to answer the questions.  Answer based ONLY on what you observe
in the video frames.  Be precise and concise.
```

The VQA user prompt template is:

```
Watch the video frames carefully and answer ALL of the following questions about
this robot manipulation video.

{questions_block}

You MUST return your answers as a JSON object.  Use the FULL question_id shown
in brackets [] as the key (NOT the short Q1/Q2 label):

{"answers":  {"<full_question_id>":  "<your_answer>", ...}}

Rules:
- For yes_no questions:  answer EXACTLY "yes" or "no"
- For number questions:  answer with JUST a number (e.g.  "3")
- For multiple_choice questions:  answer with JUST the option letter (e.g.  "B")
- Do NOT include explanations in the JSON values -- only the short answer
- You MUST answer ALL questions
```

Caption evaluation pipeline. The caption track proceeds in two stages:

Stage 1: Caption generation. Given video frames and a prompt asking for temporally ordered step-level action descriptions, each evaluated model generates a manipulation caption. In the easy setting, the original task instruction is provided; in the hard setting, only video frames are given.

Stage 2: LLM-based alignment judging. The generated caption and the pre-extracted ground-truth atomic facts (grouped by capability dimension) are passed to an LLM judge (GPT-5.4-Pro by default). The judge evaluates each GT atomic fact against the raw caption and assigns one of four labels: *match* (caption correctly states the fact), *partial* (caption addresses the event but is materially coarser or incomplete), *contradiction* (caption gives a conflicting value), or *omission* (caption does not mention the fact). Additionally, the judge identifies *hallucinated actions*: action events in the caption that have no correspondence in any GT action_sequence fact.

The alignment judge prompt specifies detailed semantic tolerance policies (color-family equivalence, synonym matching, compatible spatial wording, actor naming equivalence) and strict rules for hallucination detection (only genuinely fabricated actions with no GT basis are flagged). The judge returns a structured JSON with per-fact labels, caption evidence, and aggregate counts. The three reported metrics are then computed from these counts:

- **Consistency**: fraction of caption-addressed GT facts that are match or partial (not contradiction).
- **Coverage**: fraction of GT facts labeled match or partial (not omission).
- **Anti-Hallucination**: penalizes fabricated action events not supported by GT.

The full alignment judge system prompt is provided below:

```
You are an expert evaluator for fine-grained robot manipulation captions.

You receive:
1.  Pre-extracted GT atomic facts (structured, grouped by capability dimension).
2.  A raw AI-generated caption (a list of step descriptions, NOT pre-extracted
into atomic facts).
```

```
Your task is to evaluate each GT atomic fact against the raw caption text and
determine:
- For each GT fact:  is it match, partial, contradiction, or omission?
- Additionally, identify any hallucinated action events in the caption that do
NOT appear in the GT action_sequence facts.

GT Fact Evaluation Rules:
- match:  caption clearly states or implies the same information as the GT fact.
- partial:  caption addresses the same event but is materially coarser or
incomplete.
- contradiction:  caption addresses the same event but gives a conflicting
value.
- omission:  caption does not address this GT fact at all.

Hallucination Detection (action_sequence only):
A hallucinated action must describe a distinct, meaningful action event that no
GT action_sequence fact covers, is not a sub-action of a matched GT action, and
is not a gripper state change accompanying a matched action.

Output:  structured JSON with per-fact labels, caption evidence, and summary
counts (match + partial + contradiction + omission == total_gt_facts).
```

A.3.5 Judge Robustness

This subsection supports the judge-robustness claim made in Section 4.2. It compares caption scores obtained with GPT-5.4-Pro and Gemini-3.1-Pro as alignment judges.

The main-paper caption results use GPT-5.4-Pro as the alignment judge. To test whether the benchmark conclusions are sensitive to the evaluator, we additionally compute a Gemini-3.1-Pro-based counterpart of Table 3. Under Gemini-3.1-Pro, RoboFine-VLM remains the strongest model in both easy and hard settings, and the high-level conclusion is unchanged even though absolute scores and some close baseline orderings vary slightly.

Table 18: **Caption benchmark results on RoboFine-Bench (%) under a Gemini-3.1-Pro judge.** We report the same easy/hard caption metrics as in the main paper, but use Gemini-3.1-Pro for caption-fact alignment. **Cons.**: Consistency; **Cov.**: Coverage; **A-Hal.**: Anti-Hallucination. Best value per column is **bold**.

Model	Easy				Hard			
	Overall↑	Cons.↑	Cov.↑	A-Hal.↑	Overall↑	Cons.↑	Cov.↑	A-Hal.↑
Qwen3-VL-Plus	73.7	78.1	54.3	88.6	63.5	71.5	51.4	67.7
Qwen3.5-Plus	75.7	79.7	53.6	93.9	71.5	76.3	50.4	88.1
Doubao-Seed-2.0-Pro	77.1	80.9	53.0	**97.4**	73.7	75.2	62.9	82.7
Gemini-3.1-Pro	78.1	80.0	63.7	90.6	73.7	78.5	53.6	88.8
GPT-5.4	80.5	81.4	71.2	88.7	78.0	77.4	66.6	90.2
RoboFine-VLM (Ours)	**83.2**	**84.5**	**71.7**	93.4	**83.0**	**83.1**	**71.9**	**94.1**

A.3.6 Human Alignment Study

This subsection supports the human-alignment claim in Section 4.2. It explains how the 10-human-rater study is constructed and how the benchmark scores are normalized before correlation analysis.

We recruit 10 human raters. For each benchmark sample, annotators are shown the robot video together with six candidate captions, one from each evaluated model: Qwen3-VL-Plus, Qwen3.5-Plus, Doubao-Seed-2.0-Pro, Gemini-3.1-Pro, GPT-5.4, and RoboFine-VLM. Each caption is ranked from 1 to 6, where 1 denotes the best caption and 6 denotes the worst. The protocol is conducted on the same 500 benchmark videos used for automatic caption evaluation.



(a) Human ranking interface on a short tabletop manipulation sample.



(b) Human ranking interface on a longer multi-step kitchen task.

Figure 5: **Human ranking interface for caption evaluation.** Annotators watch the benchmark video and rank the six candidate captions from best to worst according to fine-grained faithfulness and usefulness. The protocol is designed to validate whether benchmark-induced model ranking is aligned with direct human judgment.

Annotators jointly consider factual correctness, process coverage, temporal coherence, object grounding, and resistance to hallucination. After annotation, we average the assigned ranks across raters and samples to obtain a single human score per model. These scores are normalized from the theoretical 1–6 range to $[0, 1]$, while benchmark caption Overall scores are normalized from 0–100 to $[0, 1]$. The resulting correlations reported in the main paper are high in both settings: easy Pearson **0.937** and Spearman ρ **0.943**; hard Pearson **0.922** and Spearman ρ **0.943**.

A.3.7 Caption Cost, Token, and Latency

This subsection supports the efficiency discussion in Section 4.2. It reports average token consumption and elapsed time per sample for the caption track.

Table 19 shows that RoboFine-VLM achieves the best caption quality while remaining substantially more token-efficient than several strong closed-source baselines, especially Doubao-Seed-2.0-Pro and Gemini-3.1-Pro.

Table 19: **Caption-track inference cost.** Average total token consumption and average elapsed time per sample on the caption track.

Model	Avg. total tokens	Avg. elapsed (s)
RoboFine-VLM	14,438.1	74.5
GPT-5.4	21,906.6	62.2
Doubao-Seed-2.0-Pro	93,792.2	104.8
Gemini-3.1-Pro	198,271.9	62.8
Qwen3.5-Plus	13,219.8	95.6
Qwen3-VL-Plus	8,352.0	6.4

A.3.8 Detailed Benchmark Results

This subsection supports the concise benchmark discussion in Section 4.2. It provides the more detailed result interpretation moved out of the main paper to save space.

VQA analysis. RoboFine-VLM reaches **68.2%** overall accuracy, outperforming the strongest general-purpose baseline, GPT-5.4, by **8.0** absolute points. The largest gain appears on Action and Motion Understanding, where RoboFine-VLM improves from **64.6%** to **75.7%**, indicating a substantial advantage in understanding execution order, contact patterns, and motion dynamics. The gains are also consistent on Entity and Scene Grounding (**64.7%** $\rightarrow$ **72.5%**) and Interaction and State Reasoning (**57.1%** $\rightarrow$ **68.5%**), showing that the benefit of fine-grained supervision extends beyond object recognition to process-level reasoning.

Relative to its base model, Qwen3.5-Plus, supervised fine-tuning on FineVLA-Data raises overall VQA accuracy from **55.9%** to **68.2%**. The improvement is broad-based across all three reporting

axes, with gains of **+8.6**, **+15.7**, and **+22.3** points on Gnd., Act., and State, respectively. This comparison isolates the effect of fine-grained action-aligned supervision more directly than cross-model comparison alone, and shows that FineVLA-Data substantially strengthens the robotic video understanding capability of the underlying VLM.

Caption analysis. Instruction input is beneficial for all evaluated models: caption Overall is consistently higher in the easy setting than in the hard setting. The easy-hard gap is especially large for Qwen3-VL-Plus and Doubao-Seed-2.0-Pro, suggesting that these models rely more heavily on the original task instruction. The same trend appears in Anti-Hallucination: some general-purpose VLMs depend strongly on instruction input to avoid fabricated action content, whereas RoboFine-VLM remains comparatively stable when the instruction is removed.

RoboFine-VLM remains the strongest model overall. In the easy setting, it achieves the best Overall, Consistency, and Coverage scores. In the hard setting, which requires the model to infer the manipulation process directly from video rather than from task-level language priors, RoboFine-VLM is SOTA on all four metrics and improves Overall from the strongest baseline score of **78.0%** to **82.2%**. These results indicate that the gain from fine-grained supervision is not limited to instruction-conditioned captioning, but extends to intrinsic process understanding.

Benchmark validity. Unless otherwise specified, the caption results in Table 3 use GPT-5.4-Pro as the alignment judge. Re-evaluating the same captions with Gemini-3.1-Pro preserves RoboFine-VLM as the strongest model in both easy and hard settings, despite small shifts in absolute scores and close baseline orderings; the corresponding robustness discussion is reported in Appendix A.3.5. We further compare automatic caption scores with human judgment by asking 10 raters to rank the six models on the same 500 benchmark videos. As shown in Figure 7, the resulting agreement is strong in both settings (easy: Pearson **0.937**, Spearman ρ **0.943**; hard: Pearson **0.922**, Spearman ρ **0.943**), indicating that the caption track is both aligned with human preference and robust to the choice of judge model.

A.4 FineVLA-Policy Setup

This section provides implementation details for the policy setup used in Section 3. It specifies the policy architectures, pretraining configuration, RoboTwin fine-tuning data, and instruction-mixing design used to study the effect of fine-grained language supervision.

Table 20: **Training configurations for FineVLA-Policy.** The table summarizes the backbone, training dataset, optimization length, and compute budget used in the pretraining, RoboTwin fine-tuning, and real-world fine-tuning stages. The variant names indicate the policy architecture.

Stage	Variant	Backbone	Dataset	Steps	GPUs	Global BS
Pretrain	RDT-OFT	Qwen3.5-4B	RDT	100k	64×A100	512
Pretrain	RDT-GR00T	Qwen3.5-4B	RDT	100k	64×A100	512
Pretrain	AlohaMix-OFT	Qwen3.5-4B	AlohaMix	100k	64×A100	512
RoboTwin SFT	All pretrained variants	Qwen3.5-4B	RoboTwin	100k	8×A100	128
Real-World SFT	AlohaMix-OFT	Qwen3.5-4B	Real (600 demos)	100k	8×A800	64

A.4.1 Policy Frameworks

We instantiate FineVLA-Policy with two action-decoding frameworks implemented in the StarVLA codebase. **StarVLA-GR00T** adopts a dual-system design where the VL backbone serves as System 2 for slow reasoning and a DiT-based flow-matching module serves as System 1 for fast action generation, consistent with GR00T N1.5. **StarVLA-OFT** attaches a lightweight MLP head that reads the hidden states of predefined action tokens and regresses continuous actions in parallel with an L1 objective, following OpenVLA-OFT. The two variants share the same Qwen3.5-4B backbone and the same visual observations and language inputs.

A.4.2 Pretraining Datasets and Configuration

We first pretrain three policy variants for 100k steps: **RDT-OFT**, **RDT-GR00T**, and **AlohaMix-OFT**. Here, OFT and GR00T denote the policy architecture, while RDT and AlohaMix denote the training dataset. **AlohaMix** is an ALOHA-only mixture constructed from open-source datasets such as RoboCOIN and RoboMIND, and contains 86,662 episodes across 598 tasks, approximately 13$\times$ larger than RDT. All pretraining runs use 64 A100 GPUs for 100k steps, with per-device batch size 8 and global batch size 512.

Table 21 details the composition of AlohaMix. We deliberately restrict the mixture to ALOHA-compatible dual-arm embodiments so that all trajectories share the same kinematic structure. This single-embodiment design avoids cross-embodiment confounds and lets us attribute performance differences solely to the language supervision signal. Of the 86,662 total episodes, 5,872 have fine-grained annotations produced by FineVLA-Tool and verified by human annotators; these form the FG dataset used in the instruction-mixing experiments (Section 3.2).

Table 21: **AlohaMix pretraining dataset composition.** All sources use ALOHA-compatible dual-arm embodiments. FG annotations denote the number of episodes with human-verified fine-grained instructions from FineVLA-Data. Because AlohaMix retains only the ALOHA-compatible dual-arm subset of each source, these per-source FG counts are subsets of the corresponding full-source representative counts in Table 1.

Source	Tasks	Episodes	Frames	FG Ann.
RDT	296	6,061	3,169,041	1,275
RoboCOIN	81	35,151	23,035,912	2,907
RoboMIND-V1.0	78	9,370	5,891,176	594
RoboMIND-V2.0	143	36,080	74,502,637	1,096
Total	**598**	**86,662**	**106,598,766**	**5,872**

A.4.3 RoboTwin Fine-Tuning Data and Configuration

We fine-tune the pretrained policies on RoboTwin using the union of the Clean and Random training sets. The resulting supervised fine-tuning corpus contains 27,500 trajectories and 6,075,103 transitions. All RoboTwin fine-tuning runs use 8 A100 GPUs for 100k steps, with per-device batch size 16 and global batch size 128.

A.4.4 Instruction Mixing Construction

For each dataset and architecture, pretraining samples from two fixed pools: a **Raw** pool containing all source trajectories with goal-level instructions, and an **FG** pool containing the representative subset with fine-grained process-level instructions. These contain 6,061/1,275 Raw/FG trajectories for RDT and 86,662/5,872 for AlohaMix. A task can have both Raw and FG instruction examples; not every source trajectory has an FG annotation. Shared trajectories retain identical action labels and visual observations in both pools.

The FG:Raw ratio specifies their relative sampling weights during *pretraining*. For example, at 1:4, a training sample is drawn from the FG pool with probability $1/5$ and the Raw pool with probability $4/5$. Raw-only and FG-only sample exclusively from their respective pools. This changes both language-supervision proportions and the sampling frequency of trajectories represented in the FG subset, rather than switching instruction types on an identical trajectory distribution.

We compare Raw-only, FG:Raw = 1:4, 1:2, 1:1, 2:1, 4:1, and FG-only, with fixed data pools and pretraining budgets within each architecture/data setting. For real-world evaluation, each resulting AlohaMix-OFT pretrained checkpoint is fine-tuned on the same real-robot dataset and instruction annotations with identical SFT settings. The ratio labels in the real-world results refer to pretraining, not to different real-world SFT mixtures.

A.4.5 Additional Training Details

We use the same backbone, observation interface, and action interface across all instruction-mixing settings within each framework. For real-world fine-tuning, we use StarVLA-OFT with a Qwen3.5-4B

backbone, initialized from the AlohaMix-OFT pretrained checkpoint corresponding to each FG:Raw ratio. The training set contains 600 demonstrations from 12 tabletop tasks (50 demonstrations per task); robot observations and actions are described in Appendix A.6.1. Table 22 reports the finalized optimization configuration used for the real-world results in Table 5. These results were obtained from checkpoints trained and evaluated before submission.

Table 22: **Real-world policy fine-tuning configuration.** The same SFT configuration is applied to checkpoints pretrained with different FG:Raw ratios.

Parameter	**Value**
Policy	StarVLA-OFT with Qwen3.5-4B
Training data	12 tasks, 50 demonstrations per task (600 total)
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$)
Base/VLM learning rate	5×10^{-6}
Action-model learning rate	5×10^{-5}
Learning-rate schedule	Constant, no warmup
Weight decay	10^{-8}
Optimization steps	100k
GPUs	8×A800 (80 GB each)
Batch size	8 per device; 64 global
Mixed precision	bfloat16
Gradient clipping	1.0

The real-world batch size is constrained by GPU memory: higher-resolution, synchronized multi-view observations contain more visual tokens per sample than the RoboTwin inputs. A per-device batch size of 16 (global batch size 128), as used in RoboTwin fine-tuning, exceeded the 80 GB memory limit. We therefore use the largest tested configuration that fits reliably, with 8 samples per device and a global batch size of 64. We did not conduct an extensive training-step sweep; the schedule is fixed at 100k steps for all variants. The real-world SFT trajectories, observations, actions, instruction annotations, optimization settings, training budget, checkpoint selection procedure, and evaluation are held fixed. Initialization differs: each variant starts from its corresponding instruction-mixture pretrained checkpoint. No FG:Raw ratio sweep is performed on the real-world SFT data.

A.5 RoboTwin Details and Additional Analysis

This section supports the RoboTwin results in Section 4.3. It provides the evaluation protocol, the full mixing-ratio result table, and the compact analyses that explain the inverted-U trend and the interaction with architecture and dataset scale.

A.5.1 RoboTwin Evaluation Protocol

This subsection supports the evaluation setup in Section 4.3. We evaluate policies on the official RoboTwin **Easy** and **Hard** splits and report success rate averaged over 20 episodes per task. A trial is counted as successful only if the task-specific goal condition is completed at the end of the rollout; reported scores are the average success rates on the corresponding split.

A.5.2 Full RoboTwin Results

This subsection provides the complete mixing-ratio table underlying the RoboTwin discussion in the main paper.

Table 23: **Full RoboTwin simulation success rates (%).** We compare three training settings (RDT-OFT, RDT-GR00T, and AlohaMix-OFT) under seven FG:Raw instruction ratios. Easy/Hard follow the official RoboTwin splits. Best value per column is **bold**.

FG:Raw	RDT-OFT		RDT-GR00T		AlohaMix-OFT	
	Easy↑	Hard↑	Easy↑	Hard↑	Easy↑	Hard↑
Raw-only	61.5	60.0	55.1	53.4	71.8	71.4
FG : Raw = 1 : 4	68.2	66.5	58.2	55.7	75.3	74.3
FG : Raw = 1 : 2	**74.1**	72.1	61.7	60.9	82.8	78.6
FG : Raw = 1 : 1	73.9	**72.4**	**69.4**	**68.2**	**86.8**	**82.5**
FG : Raw = 2 : 1	70.4	68.3	65.9	63.1	80.9	79.3
FG : Raw = 4 : 1	68.6	67.5	64.9	63.2	79.5	78.5
FG-only	62.9	62.0	62.1	61.5	78.3	76.1

A.5.3 Mixing-Ratio Analysis

This subsection supports the mixing-ratio interpretation in Section 4.3. Across all three evaluated settings, success rate first rises and then falls as the FG proportion increases, with a peak around FG : Raw = 1 : 2 to 1 : 1. The trend is therefore consistent with an inverted-U relationship rather than a monotonic preference for either raw-only or FG-only supervision. Empirically, this means that raw instructions primarily specify *what* to achieve, while fine-grained instructions specify *how* to achieve it; the best performance emerges when both signals are present.

A.5.4 Architecture and Scale Analysis

This subsection provides additional analysis on architectural sensitivity and data-scale interaction, complementing the main findings in Section 4.3.

Table 24 summarizes the detailed numbers used in the main-paper discussion. Panel A reports the three representative supervision regimes for each (dataset, framework) setting: Raw-only, the best mixed FG:Raw ratio, and FG-only. Panel B reports the derived comparison gaps used to support the architectural-equalization and data-scale analyses.

Table 24: **RoboTwin analysis summary.** Panel A lists the raw-only, best mixed, and FG-only performance for each setting. Panel B reports the derived gaps used in the architectural and data-scale comparisons. All values are success rates (%).

Panel A: Representative supervision regimes

Setting	Raw-only	Best mixed ratio	Best mixed	FG-only	Δ (Raw→Best)	Δ (Raw→FG)
RDT-OFT	61.5 / 60.0	1:2 / 1:1	74.1 / 72.4	62.9 / 62.0	+12.6 / +12.4	+1.4 / +2.0
RDT-GR00T	55.1 / 53.4	1:1 / 1:1	69.4 / 68.2	62.1 / 61.5	+14.3 / +14.8	+7.0 / +8.1
AlohaMix-OFT	71.8 / 71.4	1:1 / 1:1	86.8 / 82.5	78.3 / 76.1	+15.0 / +11.1	+6.5 / +4.7

Values are reported as Easy / Hard.

Panel B: Derived comparison gaps

Comparison	Raw-only gap	Best mixed gap	FG-only gap	Interpretation
OFT – GR00T on RDT	6.4 / 6.6	4.7 / 4.2	0.8 / 0.5	Framework gap narrows as FG ratio increases.
AlohaMix – RDT under OFT	10.3 / 11.4	12.7 / 10.1	15.4 / 14.1	FG benefit is larger at bigger data scale.

A.6 Real-World Policy Details

This section supports the real-world policy experiments in Section 4.4. It records the task definitions and protocol used for the real dual-arm evaluation; the quantitative results are reported in the main text (Table 5).

A.6.1 Robot Hardware and Setup

This subsection supports the real-world evaluation setting in Section 4.4.

Hardware. We use a Cobot Magic dual-arm robot with three synchronized RGB cameras (two wrist-mounted, one third-person). The action space consists of 14 joint-position commands (7 per arm) and two continuous gripper commands. Policies are trained with action chunks of length 50 and executed asynchronously at inference time. The low-level controller runs at 30 Hz, and inference is served remotely on an $8\times$A800 GPU server.

Training data. We collect 50 teleoperated demonstrations for each of 12 tabletop tasks (600 episodes total). All demonstrations are recorded with joint-space actions at 30 Hz and synchronized multi-view video. For each pretraining mixture, we initialize a language-conditioned policy from its corresponding AlohaMix-OFT pretrained checkpoint (Section 3) and fine-tune on this same dataset and its fixed instruction annotations for 100k steps on $8\times$A800 (80 GB) GPUs with global batch size 64 ($\sim$1.5 days wall-clock time).

Inference. At test time, the policy receives three RGB frames (one per camera) and a language instruction, and outputs an action chunk of 50 steps. Actions are dispatched asynchronously to the low-level controller. No post-processing or action filtering is applied.

A.6.2 Real-World Tasks

This subsection supports the task selection described in Section 4.4. The real-world evaluation suite contains two general manipulation tasks, five in-distribution instruction-sensitive task families (each comprising a paired variant), and one out-of-distribution compositional probe, each probing a specific control factor: *Clean Table* and *Stack Block* (routine manipulation), *Color* (object color grounding), *Pose* (initial-state grounding), *Approach* (approach direction), *Rotate* (rotation direction), *Arm* (active-arm selection, $\mathrm{R}\to\mathrm{R}$ / $\mathrm{L}\to\mathrm{L}$), and *Arm+Target* (active-arm selection with unseen actor-target binding, OOD probe). Table 25 lists the paired variants and their corresponding language instructions.

Table 25: **Real-world evaluation tasks and paired variants.** Each instruction-sensitive factor is tested with two complementary language variants under the same visual scene. † L $\rightarrow$ R uses an unseen actor-target binding (OOD probe).

Factor	Instruction A	Instruction B	Type
Clean Table	Clean up the table.	—	General
Stack Block	Stack the blue block on top of the red block.	—	General
Color	Put the red pen into the pen cup.	Put the blue pen into the pen cup.	ID
Pose	Pick up the pen lying on the table and place it into the box.	Pick up the pen standing on the table and place it into the box.	ID
Approach	Grasp the block from above, move it over the pink bowl, and release it.	Grasp the block from the right side, move it over the pink bowl, and release it.	ID
Rotate	Rotate the pen clockwise for 90 degrees.	Rotate the pen counter-clockwise for 90 degrees.	ID
Arm	Right hand pick up the block and place it into the right bowl.	Left hand pick up the block and place it into the left bowl.	ID
Arm+Target†	Right hand pick up the block and place it into the right bowl.	Left hand pick up the block and place it into the right bowl.	OOD

A.6.3 Real-World Evaluation Protocol

This subsection supports the real-world partial-score numbers reported in the main paper. Each task is evaluated over 10 trials in total, including its instruction variants where applicable, rather than 10 trials per variant. The eight tasks therefore yield 80 physical trials per policy setting. A trial is scored by manually checking ordered subgoals; a completed subgoal receives proportional credit. Between trials, the scene is reset to the designated initial configuration before the next evaluation begins.

A.6.4 Subgoal Definitions for Partial Scoring

Each task is decomposed into ordered subgoals. A trial receives credit proportional to the fraction of completed subgoals. Table 26 lists the subgoal sequence for each task. The **language-critical subgoal** (marked with $\star$) is the one whose completion requires resolving the fine-grained instruction factor.

Table 26: **Subgoal definitions for real-world partial scoring.** $\star$ marks the language-critical subgoal for each task.

Task	Ordered Subgoals
Clean Table	Reach $\rightarrow$ Grasp $\rightarrow$ Lift $\rightarrow$ Place in bin
Stack Block	Identify target block $\rightarrow$ Reach $\rightarrow$ Grasp $\rightarrow$ Transport $\rightarrow$ Align & Stack
Color	$\star$Identify correct color $\rightarrow$ Reach $\rightarrow$ Grasp $\rightarrow$ Lift $\rightarrow$ Place in pen cup
Pose	$\star$Identify pose $\rightarrow$ Approach from correct angle $\rightarrow$ Grasp $\rightarrow$ Lift $\rightarrow$ Place in box
Approach	$\star$Approach from the specified direction $\rightarrow$ Grasp $\rightarrow$ Lift $\rightarrow$ Move over pink bowl $\rightarrow$ Release
Rotate	Grasp $\rightarrow$ $\star$Rotate 90° in the specified direction $\rightarrow$ Release
Arm	$\star$Select the correct arm $\rightarrow$ Reach $\rightarrow$ Grasp $\rightarrow$ Transport $\rightarrow$ Place in the correct bowl
Arm+Target (OOD)	$\star$Select left arm $\rightarrow$ Reach $\rightarrow$ Grasp $\rightarrow$ Transport $\rightarrow$ Place in right bowl

A.7 Additional Analysis

This section collects supplementary observations from the RoboTwin experiments that are useful for interpretation but not required for the main narrative.

FG supervision narrows the architecture gap. Comparing StarVLA-OFT and StarVLA-GR00T on the same dataset (RDT), OFT is clearly stronger under Raw-only supervision (gap of 6.4/6.6 on Easy/Hard), but the gap shrinks as FG ratio increases and nearly vanishes under FG-only (0.8/0.5). This suggests that dense language supervision alleviates a supervision bottleneck, reducing the policy's dependence on decoder architecture choice.

FG supervision benefits more from larger data scale. Comparing RDT-OFT and AlohaMix-OFT, the gain from FG supervision is larger on the bigger AlohaMix dataset. The FG-only improvement over Raw-only grows from +1.4/+2.0 (RDT) to +6.5/+4.7 (AlohaMix). As trajectory diversity grows, dense action-aligned language has more distinct patterns to bind to, suggesting that FG supervision should become even more valuable at larger training scale.

Detailed per-setting numbers supporting both observations are reported in Table 24.

A.8 Reproducibility, Limitations, and Ethics

This section supports the reproducibility, limitation, and ethics-related claims referenced by the NeurIPS checklist. It summarizes the compute budget used in the main training experiments, clarifies the current limitations of the pipeline, and records the intended release scope.

A.8.1 Reproducibility Checklist Support

The main paper and appendix together document the data-construction pipeline, benchmark protocol, policy architectures, instruction-mixing setup, and the training configurations used in the experiments. In particular, Appendix A.1 details the construction of FineVLA-Tool, Appendix A.3 details RoboFine-Bench, and Appendix A.4 details the policy-training setup used in RoboTwin and the real-world experiments.

A.8.2 Compute Resources

RoboFine-VLM supervised fine-tuning is performed for 903 steps on 256 NVIDIA H200 GPUs with global batch size 512, learning rate decayed from 7e-6 to 7e-7, taking approximately 40 hours and using roughly 105 GB of memory per GPU. Policy pretraining is performed for 100k steps on 64 A100 GPUs with global batch size 512, taking approximately 48 hours per run and using roughly 70 GB of memory per GPU. RoboTwin fine-tuning is performed for 100k steps on 8 A100 GPUs with global batch size 128, taking approximately 48 hours per run and using roughly 75 GB of memory per GPU. Real-world fine-tuning is performed for 100k steps on $8\times$A800 (80 GB) GPUs with global batch size 64, taking approximately 1.5 days per run.

A.8.3 Limitations

The current work has two main limitations. First, although RoboFine-VLM provides high-quality scalable annotation, a small portion of generated annotations still requires manual verification, so the pipeline does not fully eliminate human-in-the-loop supervision. Second, while the method is validated on multiple datasets, RoboTwin, and a set of real-world tasks, broader validation across additional robot embodiments and a larger set of real-world tasks remains future work.

A.8.4 Societal Impact and Safety

Fine-grained language supervision can improve the controllability and transparency of robot behavior by making execution constraints more explicit. At the same time, deployment on real robotic systems still requires external safety constraints, because incorrect grounding, hallucinated action details, or control failures may lead to unintended physical interactions.

A.9 Benchmark Overview and Human Alignment

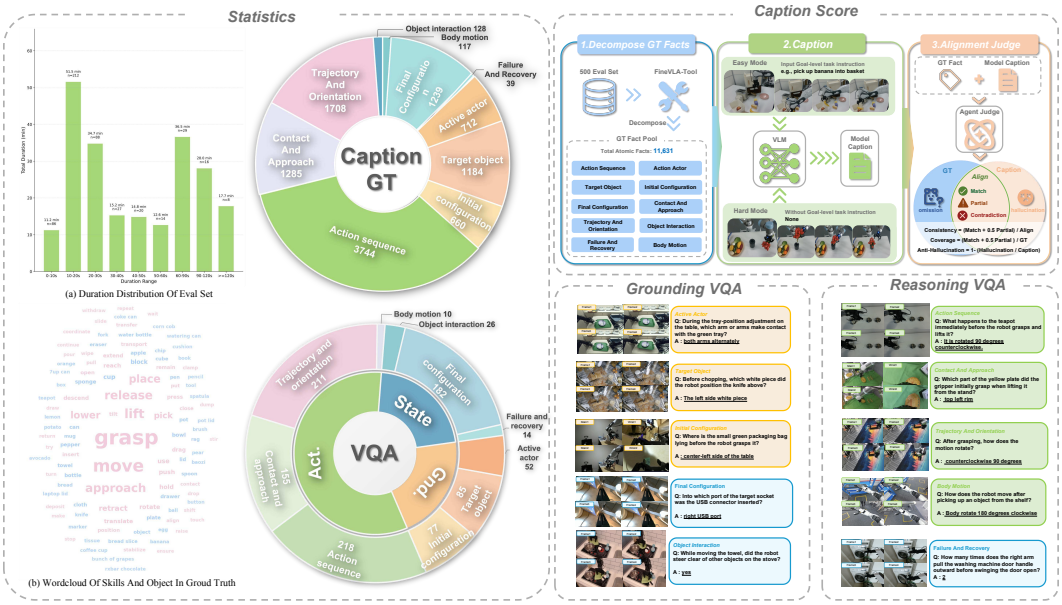
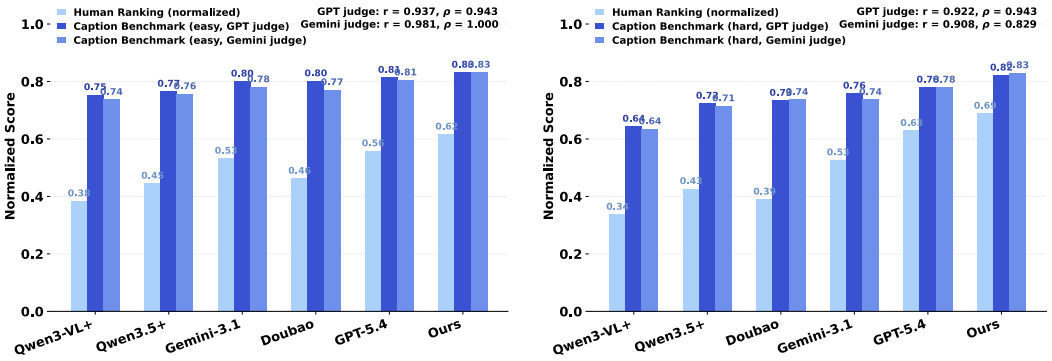


Figure 6: **Overview of RoboFine-Bench.** RoboFine-Bench evaluates fine-grained robotic video understanding through complementary VQA and captioning tracks. **Left**: benchmark statistics, including the video-duration distribution, the word cloud of manipulation skills and objects, and the distribution of ground-truth atomic facts across the ten FineVLA dimensions for captioning and VQA. **Top right**: the captioning track decomposes human-reviewed annotations into a pool of 11,631 atomic facts, asks VLMs to generate ordered step-level action descriptions under easy and hard settings, and uses an LLM judge to align model captions with ground-truth facts, producing Consistency, Coverage, and Anti-Hallucination scores. **Bottom right**: representative Grounding VQA and Reasoning VQA examples probe object/scene grounding, action/motion understanding, and interaction/state reasoning. RoboFine-Bench contains 500 held-out robot videos and 1,030 VQA questions across diverse embodiments, camera views, and manipulation scenarios.



(a) Comparison on easy mode.

(b) Comparison on hard mode.

Figure 7: **Correlation between benchmark caption scores and human ranking.** We recruit 10 human raters to rank the six models on the 500 benchmark videos, and average the resulting subjective scores. Human ranks are normalized from the 1–6 range to $[0, 1]$, while benchmark caption Overall scores are normalized from 0–100 to $[0, 1]$.

A.10 Extended Policy Analysis

This section analyzes *why* fine-grained supervision improves performance, how it should be mixed with raw goal-level instructions, and which control factors benefit most from action-aligned language.

A.10.1 Fine-Grained Supervision Does Not Sacrifice Goal-Level Success

A natural concern is that fine-grained instructions may over-specify execution details and distract the policy from completing the high-level goal. Our results suggest the opposite. In RoboTwin (Table 4), FG-only improves over Raw-only across all three (dataset, framework) combinations, with gains ranging from +1.4/+2.0 (RDT-OFT, Easy/Hard) to +7.0/+8.1 (RDT-GR00T) and +6.5/+4.7 (AlohaMix-OFT). In the real-world evaluation (Table 5), mixed supervision also matches or exceeds Raw-only on the two general manipulation tasks (Clean Table, Stack Block), where no fine-grained control factor is explicitly tested. The pattern holds regardless of decoder architecture (OFT vs. GR00T), pretraining data scale (RDT vs. AlohaMix), and environment (simulation vs. physical), indicating that process-level language provides additional action constraints without sacrificing goal-level task completion.

A.10.2 Raw and Fine-Grained Supervision Are Complementary

Although FG-only outperforms Raw-only, the *best* performance is achieved by mixing both supervision types. As the FG proportion increases from 0% to 100%, success rate traces a clear inverted-U curve in all three RoboTwin settings, peaking around $\text{FG} : \text{Raw} = 1 : 2$ to $1 : 1$ (Figure 8). The same trend transfers to the real world (Table 5): $\text{FG} : \text{Raw} = 1 : 1$ achieves the highest in-domain score (62.7/100), outperforming both Raw-only (49.9) and FG-only (54.4).

We attribute this inverted-U to the complementary roles of the two instruction types. Raw instructions preserve compact goal semantics—*what* task should be completed—while fine-grained instructions expose execution constraints—*how* the task should be performed. Under *Raw-only*, execution-level choices (arm, approach, rotation) are left to implicit co-occurrence statistics. Under *FG-only*, the policy has explicit process-level guidance, but losing goal-level abstractions may weaken generalization to instruction phrasings not seen during training. In addition, FG descriptions are longer and more distributionally different from natural user commands, so training exclusively on FG language may reduce exposure to compact goal-level task phrasing. Under *Mixed* supervision, the policy simultaneously learns task semantics from raw instructions and execution constraints from fine-grained descriptions, retaining the strengths of both. The inverted-U trend suggests that fine-grained language should augment, not replace, raw task instructions.

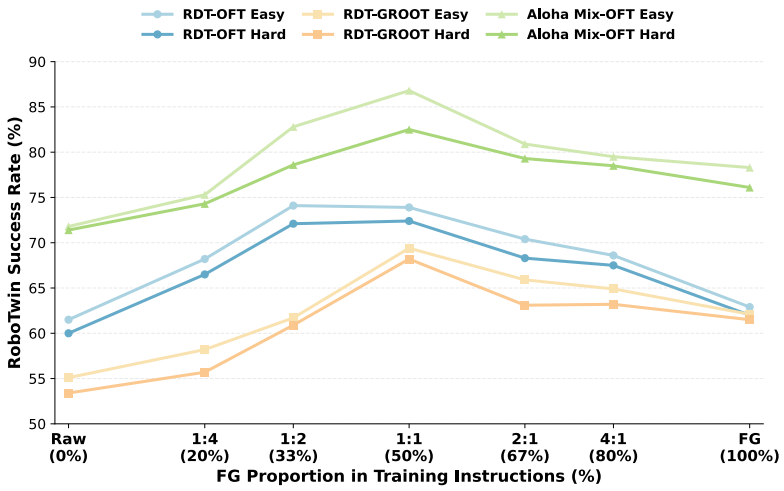


Figure 8: **RoboTwin mixing-ratio curves.** Performance peaks around FG : Raw = 1 : 2 to 1 : 1 across all settings, yielding a consistent inverted-U trend.

A.10.3 Architecture and Data-Scale Effects

FG supervision narrows the architecture gap. Comparing StarVLA-OFT and StarVLA-GR00T on the same dataset (RDT), OFT is clearly stronger under Raw-only supervision (gap of 6.4/6.6 on Easy/Hard), but the gap shrinks as FG ratio increases and nearly vanishes under FG-only (0.8/0.5). This suggests that dense language supervision alleviates a supervision bottleneck, reducing the policy's dependence on decoder architecture choice.

FG supervision benefits more from larger data scale. Comparing RDT-OFT and AlohaMix-OFT, the gain from FG supervision is larger on the bigger AlohaMix dataset. The FG-only improvement over Raw-only grows from +1.4/+2.0 (RDT) to +6.5/+4.7 (AlohaMix). As trajectory diversity grows, dense action-aligned language has more distinct execution patterns to bind to. Together with the architecture result above, this suggests that fine-grained supervision is not merely an incidental improvement for current-scale training: it represents a scalable supervision axis beyond a single architecture or dataset scale.

Detailed per-setting numbers supporting both observations are reported in Appendix A.5, Table 24.

A.10.4 Fine-Grained Language Enables Factor-Level Steerable Control

Overall task success can hide instruction violations: a policy may complete the goal-level task while using the wrong arm, approaching from the wrong direction, or rotating in the wrong direction. We therefore examine the five single-factor columns in Table 5, where each column isolates exactly one language-specified control attribute while holding the visual scene fixed.

Table 5 shows that FG : Raw = 1 : 1 improves every instruction-sensitive factor over Raw-only. The largest gains appear on attributes invisible to goal-level language: Pose ($24 \rightarrow 47$, +23), Color ($22 \rightarrow 40$, +18), and Approach ($60 \rightarrow 78$, +18). Rotate improves from 76 to 86 (+10), and Arm from 60 to 64 (+4). The gain magnitude correlates with how much each factor is underspecified by raw instructions: object pose, color, and approach direction receive no guidance in goal-level language, while rotation direction and arm selection are occasionally implied by task context. These results show that fine-grained supervision improves not only overall task completion, but also execution compliance on the specific control attribute specified by the instruction.

The OOD actor-target probe reveals a different pattern. The FG : Raw = 1 : 2 and 1 : 1 settings achieve the highest OOD scores (5 and 10 respectively), compared with 0 for Raw-only, suggesting that mixed supervision strengthens individual factor grounding. However, this does not translate into full task completion because the policy still fails to bind the selected arm to the unseen target receptacle. Thus, FineVLA improves factor-level controllability, but full compositional generalization remains unsolved.

A.10.5 Limitations

The remaining real-world failures fall into two categories. The first is *grounding failure*, where the policy selects the wrong object, arm, or target despite the language specifying the correct factor. The second is *execution failure*, where the correct factor is selected but the physical manipulation fails, such as unstable grasping, incomplete rotation, or inaccurate placement. The OOD actor-target probe further shows a compositional limitation: increasing FG supervision improves active-arm grounding, but does not reliably solve novel actor-target binding.

Our framework also has several limitations. RoboFine-VLM reduces annotation cost but does not fully remove human verification. Real-world validation is still limited to a tabletop dual-arm platform and a small set of targeted steerability tasks. Our benchmark metrics also have a scope limitation: the caption-track Anti-Hallucination score is measured only over action steps, because the ground-truth atomic actions exhaustively cover all executed actions and thus make fabrication well-defined, whereas the other nine fine-grained dimensions cannot be exhaustively enumerated—an unmatched description there may be a correct supplementary detail rather than a fabrication, so hallucination cannot be reliably defined for them. Finally, following fine-grained execution instructions in physical environments raises safety concerns; future systems should combine fine-grained language following with feasibility and safety checks.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction accurately summarize the paper's scope and contributions: FineVLA-Tool for action-aligned supervision construction, RoboFine-VLM for robotic video understanding, RoboFine-Bench for fine-grained evaluation, and FineVLA-Policy for downstream steerable control. These claims are supported by the benchmark and policy results reported in Section 4 and Appendix A.10.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discusses two main limitations: (1) RoboFine-VLM greatly reduces annotation cost but still requires limited human verification for quality control, and therefore does not fully remove human-in-the-loop supervision; and (2) although the method is validated on multiple benchmarks and real-world tasks, broader validation across more robot embodiments and a larger set of real-world tasks remains future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results, theorems, or proofs. Its contributions are empirical and system-oriented, focusing on fine-grained annotation, benchmark construction, robotic video understanding, and downstream policy learning.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper provides the key information needed to reproduce the main experimental results, including the FineVLA-Tool data-construction pipeline (Section 2), RoboFine-VLM and RoboFine-Bench protocols (Sections 2.4 and 2.3), and the policy architectures, instruction-mixing setup, and training configurations used in downstream control (Sections 3 and 4.3, Appendix A.1, Appendix A.2, Appendix A.3, Appendix A.4, and Appendix A.5).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: The project code, including FineVLA-Tool, the RoboFine-VLM weights, and RoboFine-Bench are publicly available through `https://finevla.xlang.ai/`, which links to the code repository and Hugging Face releases. The released resources include accompanying documentation and usage instructions; experimental protocols and training configurations are described in Appendix A.3 and Appendix A.4.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper specifies the benchmark construction, evaluation protocols, training datasets, instruction-mixing settings, policy architectures, and the main training configurations needed to understand the reported results. The relevant details are described in Sections 2.3, 3, 4.3, and Appendix A.3, Appendix A.4, and Appendix A.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: The paper reports deterministic benchmark scores and policy success rates for the main experiments, but does not currently include error bars, confidence intervals, or statistical significance tests. We report results over fixed benchmark sets and fixed RoboTwin evaluation protocols, and we leave a more extensive uncertainty analysis to future work.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper reports the main compute resources used in the policy-training experiments, including GPU type, number of devices, memory usage per GPU, and approximate wall-clock training time. These details are provided in Appendix A.4 and Appendix A.8.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

Justification: The research described in this paper is consistent with the NeurIPS Code of Ethics. The submission is anonymized for double-blind review, discusses limitations and deployment-related safety considerations, and does not involve any intentional deviation from the ethical guidelines.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

Answer: [Yes]

Justification: The paper discusses societal impact and safety in Appendix A.8. On the positive side, fine-grained language supervision can improve the controllability and transparency of robot behavior by making execution constraints more explicit. On the negative side, incorrect grounding, hallucinated action details, or control failures may lead to unintended physical interactions when deployed on real robotic systems, and additional external safety constraints are needed.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [N/A]

Justification: The assets introduced in this paper—a robotic annotation tool, a robotic video understanding model, a manipulation benchmark, and a VLA policy for tabletop manipulation—are domain-specific to robot control and do not pose high risks for misuse comparable to general-purpose language models or image generators. The training data consists of robot manipulation videos from existing open-source datasets. We do not anticipate significant dual-use concerns.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All source datasets (BridgeData-V2, BC-Z, RT-1, Galaxea, RoboMIND-V1/V2, RoboCOIN, RH20T-RoboInter, RDT-1B, DROID-RoboInter) and pre-trained models (Qwen3.5-397B-A17B, Qwen3.5-4B) used in this work are properly cited with their original publications. The StarVLA codebase is also cited. The source datasets are publicly available open-source robot datasets, and we use them in accordance with their intended research purposes.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The paper introduces four new assets: (1) FineVLA-Tool, a data-construction pipeline documented in Section 2 and Appendix A.1; (2) RoboFine-VLM, a robotic video understanding model documented in Section 2.4 and Appendix A.2; (3) RoboFine-Bench, a fine-grained evaluation benchmark documented in Section 2.3 and Appendix A.3; and (4) FineVLA-Policy, a VLA policy documented in Sections 3–4.4 and Appendix A.4. Each asset is described with its methodology, construction details, and intended usage. The released code, benchmark, model weights, and accompanying documentation are accessible through `https://finevla.xlang.ai/`, with links to the code repository and Hugging Face resources.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [Yes]

    Justification: The paper involves 10 human raters for caption evaluation and human reviewers for annotation verification. The human ranking protocol, including what annotators see, the ranking procedure (ranking six model captions from 1 to 6), and evaluation criteria (factual correctness, process coverage, temporal coherence, object grounding, and resistance to hallucination), is described in Appendix A.3.6. Screenshots of the ranking interface are provided in Figure 5. The annotation verification dimensions are documented in Appendix A.1.6, Table 10.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [N/A]

    Justification: The human evaluation in this work involves ranking model-generated captions and verifying annotation quality—standard evaluation tasks that do not expose participants to any physical, psychological, or privacy risks. The tasks consist of watching publicly available robot manipulation videos and comparing text outputs. This does not constitute human-subjects research requiring IRB review.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: LLMs are used as core components in this work: (1) Qwen3.5-Plus is used for initial fine-grained caption generation in the FineVLA-Tool annotation pipeline (Section 2); (2) Qwen3.5-397B-A17B serves as the base model for RoboFine-VLM, which is fine-tuned for robotic action understanding and scalable annotation (Section 2.4); and (3) GPT-5.4-Pro and Gemini-3.1-Pro are used as alignment judges for the caption track of RoboFine-Bench (Section 2.3 and Appendix A.3.5). All usages are described in the respective sections.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.