# EXECVLA: FOLLOWING FINE-GRAINED EXECUTION CONSTRAINTS IN VISION-LANGUAGE-ACTION MODELS WITH BI-LEVEL ACTION REPRESENTATION

**Anonymous authors**
Paper under double-blind review

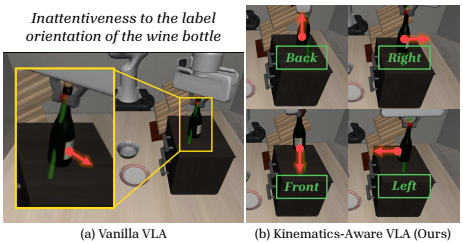
## ABSTRACT

We study fine-grained execution-constraint following in vision-language-action (VLA) models. Given an invariant task goal, the policy must follow instruction-specified execution constraints, including interaction targets, motion patterns, spatial relations, and terminal configurations. This setting exposes a limitation of goal-oriented VLAs: trajectories that complete the same task are not interchangeable when the instruction specifies how the task must be executed. We propose ExecVLA, a framework that separates a goal-oriented component from an execution-specific component through a bi-level action representation and supervised bi-level reasoning tokens. We further introduce explicit goal-invariance and execution-predictability objectives so that the goal-level representation remains stable across executions of the same goal, while the execution-level representation retains the constraints that distinguish those executions. We construct execution-constraint-following datasets in simulation and on a Realman-75 robot, with goal and fine-grained reasoning annotations. Experiments on LIBERO and the real robot show improved goal completion and, more importantly, substantially more reliable adherence to instruction-specified execution constraints.

## 1 INTRODUCTION

Vision-Language-Action (VLA) models have recently demonstrated strong performance in goal-oriented robotic manipulation. However, most existing approaches implicitly assume goal equivalence under instruction paraphrasing, meaning that instructions describing the same task goal can be executed with interchangeable motion trajectories. As a result, these models are largely insensitive to fine-grained execution constraints expressed in natural language. In real-world human robot interaction, this assumption frequently breaks down. Semantically equivalent goals may require distinct and non-interchangeable trajectories due to explicit execution constraints such as position, orientation, distance, and motion direction.

We formalize the setting as follows: *given an invariant task goal, the policy must follow instruction-specified execution constraints, including interaction targets, motion patterns, spatial relations, and terminal configurations.* The distinction is between what outcome must be achieved and how the policy is required to achieve it; it is not a claim about kinematics in the classical robotics sense.




(a) Vanilla VLA (b) Kinematics-Aware VLA (Ours)

Figure 1: Unlike vanilla VLAs Kim et al. (2025); Wang et al. (2025), ExecVLA follows the requested terminal orientation.

For example, as shown in Figure 1, instructions such as "place the bottle on the drawer" and "place the bottle on the drawer with the label facing right, front, or back" describe an identical task goal, yet they demand fundamentally different execution trajectories and end-effector motions. These trajectories cannot be treated as interchangeable refinements of the same action without violating the instruction semantics. This setting goes beyond conventional fine-grained goal specification and instead characterizes a class



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

of tasks that are goal invariant but execution variant. Such tasks are not captured by the implicit assumptions underlying most existing VLA formulations.

Despite the rapid progress in recent VLA models such as OpenVLA Kim et al. (2025) and $\pi_0$ Black et al. (2024), most existing approaches focus on semantic-level grounding, mapping visual observations and language descriptions to discrete action tokens or low-frequency control trajectories. While these models generalize well to diverse manipulation tasks, their generated actions typically reflect task goal rather than motion precision. In other words, current VLAs are optimized to determine what to do (e.g., grasp or place) but not how to do it (e.g., at which position, distance, or orientation). Moreover, although tokenization-based or diffusion-based decoders improve temporal smoothness, they remain agnostic to fine-grained execution cues in language. This gap prevents existing policies from achieving physically consistent, user-customized manipulations that depend on explicit execution-constraint instructions.

Naive binning struggles with complex actions, while diffusion-based policies are computationally expensive. We therefore adopt vector-quantized action representations Lee et al. (2024); Wang et al. (2025); Mete et al. (2024); Pertsch et al. (2025) to capture fine-grained motion and temporal consistency for more stable, sample efficient training. To avoid the entanglement of task identity and execution realization in single-level tokenizers, we further introduce a bi-level action representation that decouples goals and execution constraints, making it well suited for execution-constraint-rich tasks.

ExecVLA uses a bi-level vector-quantized action representation with a goal-level codebook for task identity and an execution-level codebook for instruction-required motion details. Because bi-level action tokens alone do not guarantee correct language grounding, we introduce supervised bi-level reasoning tokens that separately describe the task goal and its execution constraints. The model jointly generates these textual intermediate variables and the corresponding discrete action tokens. To ensure consistency between reasoning and control, we maximize the conditional mutual information $I(\textit{Reasoning}; \textit{Action} \mid C)$, which encourages the generated trajectory to remain semantically grounded, thus forming a coherent link between reasoning and execution. Our reasoning tokens are not intended to improve abstract planning ability, but to serve as explicit, supervised intermediate variables that align language and action.

To alleviate the data scarcity of this new task, we constructed three **Execution-Constraint-Rich Datasets** that systematically captures a variety of goals and multi-level execution details, covering diverse tabletop organization and object manipulation scenarios. The total number of frames are 378K. We collected paired VLA data using the LIBERO platform Liu et al. (2023) and the Realman-75 robot. The execution-constraint-rich language instructions are also annotated with reasoning texts at two levels: 1) general coarse-grained goal, and 2) fine-grained execution constraints. Our execution-constraint-rich VLA datasets capture the entire process of action execution, not only coarsely or partially, as illustrated in Figure 2. The dataset encompasses diverse actions that encode object-part interactions, target spatial relations, and execution constraints. Such a detailed annotation scheme enables the model to learn highly precise motion representations.

Extensive experiments show that ExecVLA achieves state-of-the-art performance on execution-constraint-aware benchmarks. Unlike previous models that produce relatively rigid, pre-defined, and goal-directed motions, our model performs flexible, interpretable, and execution-constraint-aware operations directly from execution-constraint-rich instructions.

In summary, our main contributions are as follows:

- We introduce a new execution-constraint-following manipulation task, which bridges semantic goals with diverse action execution trajectories, with the aim of meeting the needs of users' for customized table organization tasks.
- We present **ExecVLA**, a VLA model with bi-level vector quantized action representation that decouples task goals from execution variations, enabling interpretable and execution-sensitive control.
- We construct three new **Execution-Constraint-Aware Manipulation Dataset** covering both simulated and real-world scenarios, with detailed spatial, object-level, and motion-constraint annotations to facilitate execution-specific reasoning.

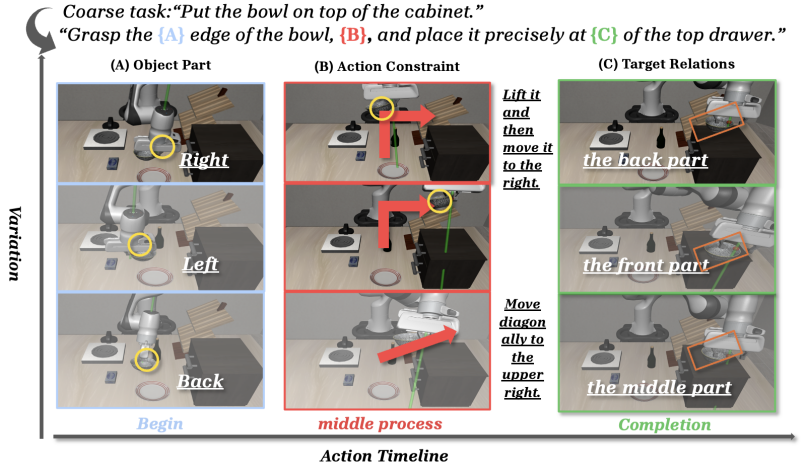



Figure 2: An example from our execution-constraint-rich datasets. For an invariant coarse goal, instructions vary the interaction target, motion pattern, and target relation over the action timeline. ExecVLA is trained to preserve the goal while following these execution-specific requirements.

## 2 Related Work

**Vision-Language-Action Models:** In recent years, VLA models have rapidly progressed from early end-to-end robotic policies grounded in paired visual perception and text to generalist systems capable of long-horizon, multi-stage manipulation. Works such as RT-2 Zitkovich et al. (2023) and RT-X O'Neill et al. (2024) demonstrated that grounding large vision–language models in real robot trajectories can yield surprisingly strong zero-shot control across tasks and embodiments. OpenVLA Kim et al. (2025) established an influential open-source baseline by coupling a 7B LLM with strong visual encoders and nearly a million real robot demonstrations, showing competitive, cross-embodiment generalization, and efficient task adaptation. More recently, the $\pi_0$ Black et al. (2024) family reframed action generation with flow-matching and high-frequency action chunking for dexterous control, yielding robust closed-loop visuomotor skills from mixed supervisory signals. Building on this, $\pi_{0.5}$ Intelligence et al. (2025b) co-trains across heterogeneous data sources—multi-robot logs, web-scale semantics, and auxiliary prediction tasks to widen open-world generalization in the wild.

Based on their action decoding strategies, current VLA models can be broadly categorized into three classes: those employing per-timestep binning scheme (e.g., (Brohan et al., 2022; Zitkovich et al., 2023; Kim et al., 2025)), diffusion-based decoding (e.g., (Liu et al., 2024; 2025; Black et al., 2024; Intelligence et al., 2025b)), and those adopting vector-quantized action representations (e.g., (Belkhale & Sadigh; Lee et al., 2024; Mete et al., 2024; Pertsch et al., 2025; Wang et al., 2025)). In our work, we emphasize the structural nature of action primitives and their compositional semantics. To this end, we adopt vector-quantized action representations approach and introduce a bi-level RVQ-VAE training framework, which jointly captures global task context and fine-grained execution patterns, enabling temporally coherent and semantically grounded action decoding.

**Physically Reliable Robot Manipulation:** As the community increasingly seeks to enhance robots' ability to interact reliably with the real physical world, physically plausible robot manipulation has emerged as a key research frontier. Instead of relying solely on visual imitation or simulation priors, recent works integrate force–torque sensing Yu et al. (2025a); He et al. (2025); Huang et al. (2025), tactile feedback Huang et al. (2025); Wu et al. (2025); Bi et al. (2025), and affordance-guided Xu et al. (2024) reasoning to better capture the dynamics of contact and object interaction.

ForceVLA (Yu et al., 2025a) and FoAR (He et al., 2025) augment VLA pipelines with wrench data and impedance feedback, improving compliance, force regulation, and fine contact control in real manipulation. Tactile-VLA (Huang et al., 2025) and VLA-Touch (Bi et al., 2025) demonstrate the benefit of integrating high-frequency tactile embeddings and visuotactile fusion, enabling policies that can infer slip, deformation, and surface geometry directly from touch signals. FreeTacMan (Wu et al., 2025) further scales tactile-conditioned manipulation across diverse objects and sensors through

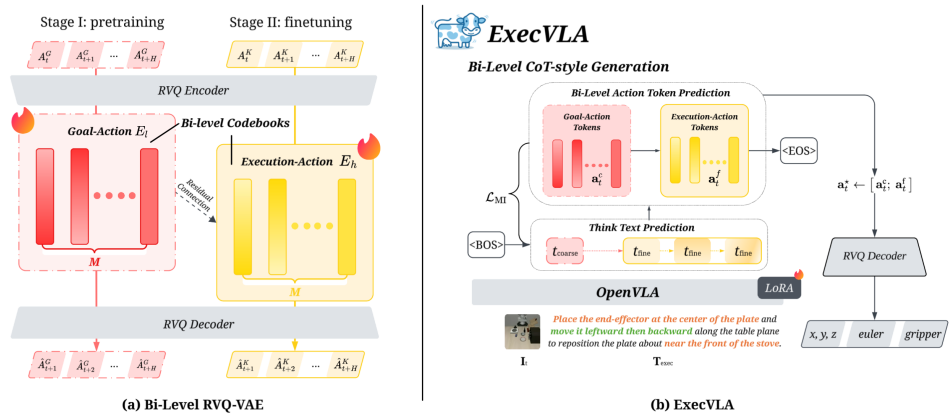



Figure 3: Overview of our framework. Our approach decouples goal-level control from execution-level refinements to handle execution-constraint-rich tasks. (a) The proposed Bi-Level RVQ-VAE learns a bi-level action representation (Sec. 3.1), while (b) ExecVLA predicts bi-level reasoning and action tokens (Sec. 3.2).

self-supervised visuotactile representation learning. In parallel, affordance-driven models such as NaturalVLM (Xu et al., 2024) bridge semantic perception and physical reasoning by predicting object parts, functional regions, and graspable affordances conditioned on natural-language queries.

Overall, these efforts can be viewed as emphasizing physical dynamics through the incorporation of additional sensory modalities. In contrast, our work focuses on fine-grained, user-customized instruction tasks, highlighting execution-specific reasoning of robot motion under diverse physics-aware settings, thereby bridging linguistic goal and physically grounded, precise action execution.

## 3 Method

In this section, we present our method, comprising a bi-level RVQ-VAE and ExecVLA. We begin by presenting bi-level vector quantized action representation (Sec. 3.1), detailing its implementation specifics of training a Bi-Level Residual VQ-VQE. We then describe the bi-level generation paradigm and the associated mutual-information regularization scheme (Sec. 3.2).

### 3.1 Bi-Level Action Representation

Uniform binning of each action dimension, as in OpenVLA-style discretization where token IDs lie in $[0, 255]$, fixes partition boundaries and ignores cross-joint correlations. This affects the robustness of low-frequency trends and sensitivity to high-frequency transients. We instead learn discrete *vector-quantized* (VQ) action tokens inspired by Zeghidour et al. (2021); Lee et al. (2024); Wang et al. (2025), then extend them to a *bi-level* design that targets both regimes.

**Vector-Quantized Action Representation.** Given a chunk of an action sequence $A_{t:t+H} \in \mathbb{R}^{T\times D}$, an encoder $f_\theta$ produces latents $z_t = f_\theta(A_{t:t+H}) \in \mathbb{R}^d$. It uses $M$ codebooks $E^{(m)} = \{e_k^{(m)}\}_{k=1}^{K_m}$ applied sequentially to quantify residuals by Residual Vector Quantized Variational AutoEncoder (RVQ-VAE) Zeghidour et al. (2021). Initialize $r_t^{(0)} = z_t$ and for $m = 1, \ldots, M$:

$$
\begin{aligned}
k_t^{(m)} &= \arg\min_k \left\| r_t^{(m-1)} - e_k^{(m)} \right\|_2^2, \\
\tilde{z}_t^{(m)} &= e_{k_t^{(m)}}^{(m)}, \\
r_t^{(m)} &= r_t^{(m-1)} - \tilde{z}_t^{(m)}.
\end{aligned}
\tag{1}
$$

The final quantized latent and discrete token tuple are

$$
\tilde{z}_t = \sum_{m=1}^{M} \tilde{z}_t^{(m)}, \qquad \mathbf{i}_t = \left(k_t^{(1)}, \ldots, k_t^{(M)}\right) \text{(tokens)}. \tag{2}
$$

A decoder $g_\phi$ reconstructs continuous actions $\hat{A}_{t:t+H} = g_\phi(\tilde{z}_{1:T})$. We train with a reconstruction term and RVQ codebook and commitment terms:

$$\begin{aligned}\mathcal{L}_{\text{rec}} &= \frac{1}{T}\sum_{t=1}^{T}\|A_t - \hat{A}_t\|_1, \\ \mathcal{L}_{\text{rvq}} &= \mathcal{L}_{\text{rec}} + \sum_{m=1}^{M}\sum_{t=1}^{T}\left\|\text{sg}(r_t^{(m-1)}) - \tilde{z}_t^{(m)}\right\|_2^2 \\ &\quad + \sum_{m=1}^{M}\sum_{t=1}^{T}\beta_m\left\|r_t^{(m-1)} - \text{sg}(\tilde{z}_t^{(m)})\right\|_2^2,\end{aligned} \tag{3}$$

where $\text{sg}(\cdot)$ is the stop-gradient operator and $\beta_m$ balances commitment at level $m$. Unlike OpenVLA's binning scheme, our partitions are learned, non-uniform, and compositional across $M$ residual levels, which couples joints and time while yielding compact semantic tokens.

**Bi-Level Action Representation.** Building on the vector-quantized formulation above, we factor an action chunk $A$ into two complementary latent variables,

$$z_g = E_g(A), \qquad z_e = E_e(A), \tag{4}$$

where $z_g$ is the goal-oriented component and $z_e$ is the execution-specific component. The goal codebook captures the task outcome shared by multiple valid executions, whereas the execution codebook captures interaction targets, motion patterns, spatial relations, and terminal configurations. Quantized versions of $z_g$ and $z_e$ are combined by the decoder to reconstruct the action chunk. This factorization is supervised by the goal and fine-grained reasoning annotations rather than being induced only through frequency separation.

**Goal-Invariance Contrastive Loss.** For two trajectories $A_{g,e_1}$ and $A_{g,e_2}$ that achieve the same goal $g$ under different execution constraints $e_1$ and $e_2$, we encourage their goal-oriented representations to agree:

$$\mathcal{L}_{\text{goal-inv}} = \left\|z_g^{(g,e_1)} - z_g^{(g,e_2)}\right\|_2^2 + \mathcal{L}_{\text{sep}}. \tag{5}$$

Here $\mathcal{L}_{\text{sep}}$ is a standard contrastive separation term for samples with different goals. It prevents a collapsed solution and gives direct supervision that the coarse representation should be invariant to execution variation.

**Execution-Predictability Loss.** Let $c_{\text{exec}}$ denote the annotated execution constraints, including trajectory direction, object part, target relation, and terminal orientation. A prediction head $q$ reads only the execution-specific latent:

$$\mathcal{L}_{\text{exec}} = \text{CE}(q(c_{\text{exec}} \mid z_e), c_{\text{exec}})\,. \tag{6}$$

This objective verifies that the fine representation contains instruction-relevant execution information rather than an undifferentiated reconstruction residual. The action-tokenizer objective is

$$\mathcal{L}_{\text{tokenizer}} = \mathcal{L}_{\text{rvq}} + \lambda_g\mathcal{L}_{\text{goal-inv}} + \lambda_e\mathcal{L}_{\text{exec}}. \tag{7}$$

**Bi-Level RVQ-VAE Training.** We adopt a two-stage training schedule with stepwise codebook learning, while keeping the two codebooks of equal size. The explicit losses above encourage the goal codebook to preserve goal identity and the execution codebook to retain constraint-specific variations:

**Stage I (goal-oriented pretraining).** Train the goal path with RVQ-VAE on broad, multi-task actions $A^G_{t:t+H}$ (e.g., Open X-Embodiment O'Neill et al. (2024), ManiSkill Mu et al. (2021)) to learn $E_g$ and its quantizer. This stage encourages tokens that stably represent task outcomes.

**Stage II (execution-aware finetuning).** Freeze $E_g$ and its quantizer. Initialize an execution path with a new RVQ stack and codebook $E_e$ of the same size, warm-start the shared encoder/decoder from Stage I, and train on execution-constraint-rich actions $A^E_{t:t+H}$. The supervision in $\mathcal{L}_{\text{exec}}$ specializes $E_e$ for instruction-required execution details; $E_e$ and its quantizer are learned from scratch while the shared modules are finetuned.

### 3.2 BI-LEVEL GENERATION

After decoupling actions into a two-level representation, we predict bi-level reasoning tokens to align internal representations with language parsing. This is inspired by chain-of-thought in large language models Yang et al. (2025); Guo et al. (2025), this design improves interpretability, constraint injection and is trained end to end via supervised finetuning, avoiding the instability of reinforcement learning Yu et al. (2025b). To improve coherence between reasoning and action tokens, we introduce a mutual information regularization that aligns the two bi-level streams and reduces execution drift.

**Reasoning Tokens Prediction.** Given the input image $\mathbf{I_t}$ and a execution-constraint-rich instruction $\mathbf{T}_{\text{exec}}$, we fine-tune an end to end base VLM model $f_\theta$ to produce two reasoning text outputs $t_{\text{coarse}}$ and $t_{\text{fine}}$. The coarse reasoning text expresses the general task goal in natural language, while the fine reasoning text specifies key execution parameters and anchor points; see Section. 4 for concrete examples. Training uses a language cross entropy objective over both outputs,

$$\mathcal{L}_{reasoning}(\theta) = \sum_{m\in\{c,f\}} \sum_{t=1}^{L_m} \mathrm{CE}\Big(h_t^m,\, p_\theta\big(h_t^m \mid h_{<t}^m,\, \mathcal{C}\big)\Big), \tag{8}$$

where $\mathcal{C} = (\mathbf{I_t}, \mathbf{t}_{\text{exec}})$, $h_t^c$ and $h_t^f$ are tokens from the coarse and fine reasoning texts, $L_c$ and $L_f$ are their lengths.

**Action Tokens Prediction.** We use the bi-level action representation derived in Sec. 3.1 as ground-truth labels for supervised prediction. Because the action codebook indices learned by the bi-level RVQ-VAE reside in their own index space, we offset these indices to the tail of the text vocabulary. This lets the model reuse a shared output head and apply a unified loss across both text and action tokens, avoiding any extra task-specific head while keeping supervision consistent. Following the reasoning text prediction formulation, the prediction objective is

$$\mathcal{L}_{action}(\theta) = \sum_{m\in\{c,f\}} \sum_{t=1}^{L_m} \mathrm{CE}\Big(a_t^m,\, p_\theta\big(a_t^m \mid a_{<t}^m,\, \mathcal{C}\big)\Big), \tag{9}$$

where $a_t^c$ and $a_t^f$ are tokens from the coarse-grained and fine-grained action tokens.

**Mutual Information Regularization for reason Text and Action.** Although the proposed model jointly generates textual reasoning (*Reasoning*) and continuous control trajectories (*Action*), these two objectives are often optimized independently, which can lead to semantic disconnection. In such cases, textual reasoning may not faithfully reflect the underlying motor intention. To address this issue, we maximize conditional mutual information $I(\text{Reasoning}; \text{Action} \mid C)$, where $C$ denotes the given context (e.g., task condition or observation). We achieve this by pulling paired embeddings $(\text{Reasoning}, \text{Action})$ closer together and pushing mismatched pairs apart using an InfoNCE objective Oord et al. (2018); He et al. (2020); Chen et al. (2020).

**Algorithm 1:** Bi-Level Action Inference

**Input :** Sequence length $T$, chunk size $H{=}5$; observations $\{(\mathbf{I}_t, \mathbf{W}_t)\}_{t=1}^{T}$; execution-constraint instruction $\mathbf{T}_{\text{exec}}$; generator $\mathcal{G}$; bi-level decoder $\mathcal{D}$

**Output :** Predicted action chunks $\{\hat{\mathbf{A}}_{t:t+H}\}_{t=1}^{T-H}$

**for** $t \leftarrow 1$ **to** $T - H$ **do**

- $\mathbf{o}_t \leftarrow \mathrm{Enc}_{\text{obs}}(\mathbf{I}_t, \mathbf{W}_t)$
- $\mathbf{t}_{\text{exec}} \leftarrow \mathrm{Enc}_{\text{exec}}(\mathbf{T}_{\text{exec}})$
- $\mathcal{C}_t \leftarrow \mathrm{BuildContext}(\mathbf{o}_t, \mathbf{t}_{\text{exec}})$
- `# Bi-level generation`
- $(\mathbf{h}_t^{\text{c}}, \mathbf{a}_t^{\text{c}}),\ (\mathbf{h}_t^{\text{f}}, \mathbf{a}_t^{\text{f}}) \leftarrow \mathcal{G}(\mathcal{C}_t)$
- $\mathbf{a}_t^\star \leftarrow [\mathbf{a}_t^{\text{c}};\, \mathbf{a}_t^{\text{f}}]$
- `# Chunked action decoding`
- $\hat{\mathbf{A}}_{t:t+H} \leftarrow \mathcal{D}(\mathbf{a}_t^\star)$

Formally, given a batch $\{(T_i, A_i, C_i)\}_{i=1}^{N}$, we obtain embeddings $\mathbf{t}_i = f_T(T_i, C_i)$ and $\mathbf{a}_i = f_A(A_i, C_i)$, and define the similarity

$$s_{ij} = \frac{\mathbf{t}_i^\top \mathbf{a}_j}{\tau}, \tag{10}$$

where $\tau > 0$ is a temperature parameter.

The symmetric mutual information regularization loss is

$$\mathcal{L}_{\text{MI}} = -\frac{1}{2N} \sum_{i=1}^{N} \left[ \log \frac{e^{s_{ii}}}{\sum_{j=1}^{N} e^{s_{ij}}} + \log \frac{e^{s_{ii}}}{\sum_{j=1}^{N} e^{s_{ji}}} \right]. \tag{11}$$

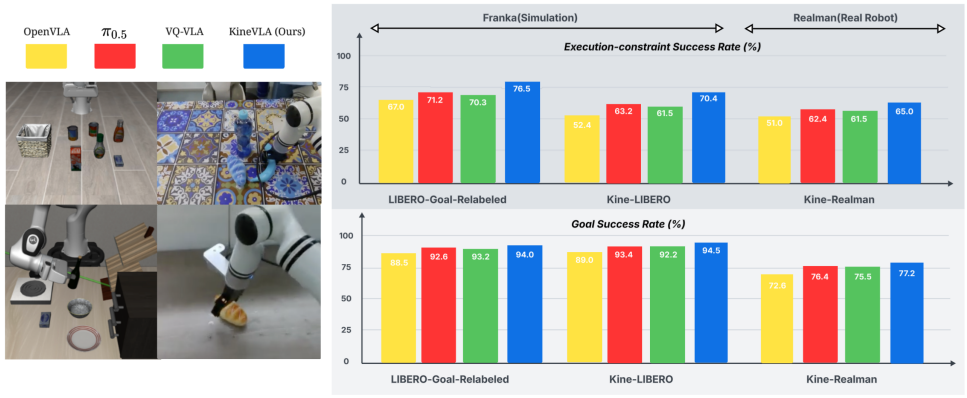



Figure 4: Experimental Results and Comparisons. We benchmark our method across the three proposed execution-constraint-aware datasets, encompassing both simulation and real-world robotic experiments. The left two figures illustrate example environments, while the bar chart on the right presents the goal and execution-constraint success rates. More results are provided in the Appendix.

**Bi-Level Action Coupling Inference.** As presented in Algorithm 1 , we perform action inference from a scene image, a wrist image, and a execution-constraint instruction using a generator–decoder pipeline. The generator emits bi-level chain-of-thought text and discrete action codebooks: a high-level plan and a low-level control trace. Each level is delimited with special tokens and paired with its own action codes. A bi-level decoder then fuses both codebooks to reconstruct a single action chunk.

### 3.3 The Base Vision-Language Model

We build ExecVLA on top of OpenVLA Kim et al. (2025) as the base vision–language model. Whereas OpenVLA originally adopts a discrete action tokenization scheme, we replace it with our proposed bi-level action representation. The action representations are learned via a residual VQ-VAE Zeghidour et al. (2021) trained for action reconstruction, enabling compact yet expressive control sequences. We retain OpenVLA's Prismatic-7B Karamcheti et al. (2024) vision–language backbone, pre-trained on large-scale image–text data, and perform a second-stage adaptation using LoRA (rank = 32) applied to all linear layers, with Gaussian initialization of the LoRA weights. The input pipeline mirrors OpenVLA except that the natural-language instruction channel is replaced by fine-grained execution constraints; other inputs (e.g., visual observations and task context) remain unchanged. The decoder jointly emits reason text and action tokens, both mapped into the same vocabulary index space as text, facilitating the use of a unified entropy loss. Unless otherwise noted, we adhere to OpenVLA's standard data preprocessing, and optimization hyperparameters.

## 4 Experiments

### 4.1 Experimental Setup

We leverage three datasets (LIBERO-Goal-Relabeled, LIBERO-GOAL, Exec-Realman) to comprehensively cover a wide variety of goals and to annotate them with diverse, fine-grained execution-constraint-rich instructions. For details, please refer to Table 9 in Appendix for a summary of our proposed dataset. Additional details are provided in the Appendix. All execution-constraint descriptions in our datasets are defined in a consistent third-person camera coordinate frame aligned with human observation. Directional and positional terms such as left, right, near, and far are specified relative to the external camera view, rather than the robot base or end-effector frame. For each task, we curate a total of 50 demonstrations and assess performance across 20 evaluation trials. RVQ-VAE is trained on a single A100 GPU with a batch size of 1024. For a fair comparison, all fine-tuning is conducted for 50K gradient steps with a total batch size of 64, using 4 A100 GPUs and an action chunk length of H=5.

**Evaluation Metrics.** For action-success evaluation, we distinguish between the *execution-constraint success rate* and the *goal success rate*. The execution-constraint success rate automatically checks

whether the action satisfies fine-grained direction, distance, and position requirements. The goal success rate instead measures whether the model completes the intended high-level operation, without considering detailed execution constraints. For reasoning evaluation, we use three complementary metrics. BERTScore captures contextual semantic similarity and is robust to paraphrasing, while BLEU measures surface-level n-gram overlap. We also include an intervention metric, where predicted reasoning tokens are replaced with random or mismatched reasoning tokens from other instructions to test their effect on performance.

**Baselines.** We compare with OpenVLA Kim et al. (2025), $\pi_{0.5}$ Intelligence et al. (2025a), and VQ-VLA Wang et al. (2025). OpenVLA is a vision–language–action transformer that conditions on image features and a text goal to regress continuous low-level actions. $\pi_{0.5}$ is a diffusion-based model, which uses co-training on heterogeneous tasks to enable broad generalization. VQ-VLA discretizes the action space using a single-level vector-quantized action tokenizer. For fairness, these methods consume the same vision observations and the same execution-constraint-rich language instructions. More baselines (GR00T Bjorck et al. (2025) and FAST Pertsch et al. (2025)) are included in the Appendix.

### 4.2 Evaluations Results

**Quantivative Performance Analysis**. As shown in Figure 4, all methods achieve comparable performance in terms of goal success rate across both simulation and real-world benchmarks, with only marginal differences between models. This indicates that existing VLA approaches are generally effective at capturing high-level task objectives. In contrast, when evaluated on execution-constraint success rate, the performance gap becomes substantially larger. While baseline methods exhibit a notable drop in execution-constraint accuracy, ExecVLA consistently achieves significantly higher success rates across all datasets.

This discrepancy highlights a key limitation of prior approaches, which tend to collapse fine-grained execution variations once the task goal is reached. By contrast, ExecVLA explicitly decouples goal-level intent from execution-specific execution through its bi-level action representation and structured reasoning alignment, enabling precise adherence to instruction-level execution constraints. These results demonstrate that modeling execution constraints as a first-class component is essential for execution-constraint-rich tasks, and that improvements in such settings cannot be achieved by goal-oriented modeling alone.

Table 1: Bi-level reasoning evaluation using the BLEU, BERTScore and Intervention metrics.

| Dataset | BLEU | BERTScore | Intervention (%) |
|---|---|---|---|
| LIBERO-Goal-Relabled | 0.93 | 0.94 | 76.5→52.4 |
| Exec-LIBERO | 0.95 | 0.96 | 70.4→48.6 |
| Exec-Realman | 0.92 | 0.94 | 65.0→42.4 |

**Visualization Examples.** Figure 5 illustrates qualitative examples of task execution using ExecVLA, highlighting its ability to reason and act effectively in execution-constraint-aware manipulation tasks. From left to right, the figure demonstrates how ExecVLA processes a execution-constraint-rich instruction, interprets the initial environmental state, and generates bi-level reasoning text, comprising goal-level and execution-specific reasoning. The corresponding bi-level action tokens are then decoded into concrete robot motions, shown through key intermediate states and the final successful execution. This bi-level reasoning framework enables ExecVLA to bridge goal understanding with precise execution-specific control. For instance, it explicitly plans object contact, trajectory, and pose orientation, ensuring stability and accuracy during manipulation.

**Bi-level Reasoning Analysis.** Table 1 shows that ExecVLA achieves consistently high BLEU scores across all datasets, indicating strong alignment between the generated bi-level reasoning texts and the ground-truth annotations. To assess the necessity of bi-level reasoning, we conduct intervention-based analysis by replacing the predicted reasoning tokens with randomly sampled or mismatched ones while keeping visual inputs and task goals unchanged. This leads to a substantial drop in execution-constraint-following success rates from 76.5% to 52.4%, from 70.4% to 48.6%, and from 65.0% to 42.4% across the three datasets, respectively, while goal completion remains largely unaffected. These results demonstrate that bi-level reasoning tokens play a causal role in grounding execution constraints into action generation.

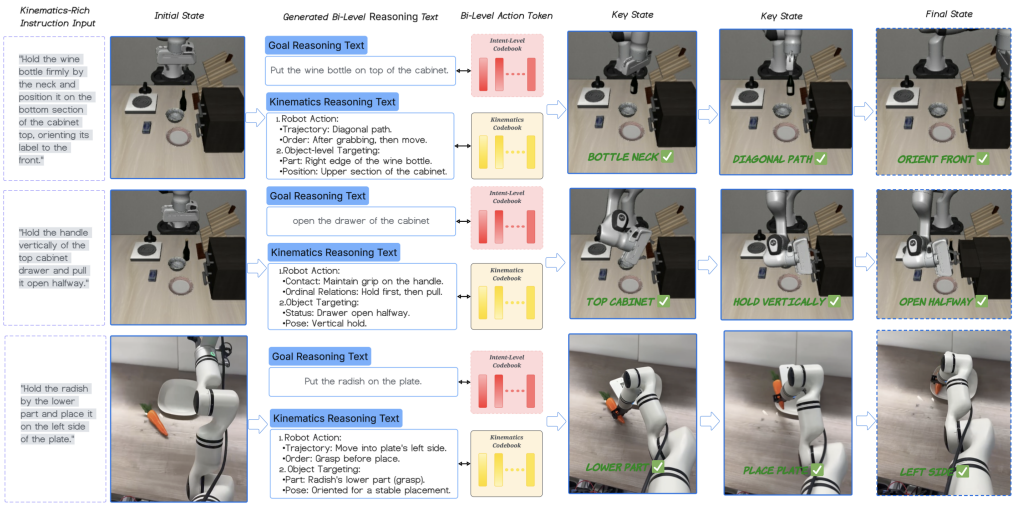



Figure 5: Task execution examples using ExecVLA. From left to right, the figure shows the execution-constraint-rich instruction input, the initial state of the environment, followed by the bi-level reasoning text and action tokens generated by the ExecVLA model. Next are two columns illustrating the key robot action states, and finally, the resulting final state after the robot executes the task.

### 4.3 Ablation Study

The ablation results in Table 2 show the contribution of each proposed component in ExecVLA. Starting from the *Baseline*, which uses standard vector-quantized action tokenization with OpenVLA, introducing Bi-Rep leads to a clear improvement in performance, increasing success rates from 70.0% to 73.4% on LIBERO-Goal-Relabeled and from 61.5% to 66.8% on Exec-LIBERO. This demonstrates that the bi-level action representation effectively captures finer-grained motion semantics.

Table 2: Ablation studies of ExecVLA components. Execution-constraint success rates (%) across two execution-constraint-rich datasets on LIBERO platform. "Baseline" refers to the standard vector-quantized action tokenization using OpenVLA. "+ Bi-Rep" denotes the use of our Bi-Level action representation. "+ Bi-Rea" indicates the incorporation of bi-level reasoning token. "+ MI" refers to the addition of mutual information optimization between the reasoning text and the action.

| Dataset | Baseline | + Bi-Rep | + Bi-Rea | + MI |
|---|---|---|---|---|
| LIBERO-Goal-Relabled | 70.0 | 73.4 | 75.5 | 76.5 |
| Exec-LIBERO | 61.5 | 66.8 | 68.6 | 70.4 |

Adding *Bi-Rea*, which introduces bi-level reasoning tokens, further improves performance to 75.5% and 68.6% on the two datasets. This demonstrates the importance of explicitly aligning language instructions with bi-level action representations, rather than relying on action modeling alone.

Finally, integrating *MI*, the mutual information optimization between the reasoning text and the action, achieves the best results of 76.5% on LIBERO-Goal-Relabeled and 70.4% on Exec-LIBERO. This shows that aligning textual reasoning with action execution leads to more consistent and robust behavior.

## 5 Conclusion and Limitation

In this work, we introduce ExecVLA, a execution-constraint-rich vision–language–action framework that enables robots to understand and execute fine-grained motion instructions by explicitly disentangling execution sensitivity from goal invariance. This design allows a single goal to be realized through actions with varying execution granularity, improving both flexibility and interpretability. To support this task, we construct comprehensive simulated and real-world datasets with detailed execution-constraint annotations. While our current evaluation focuses on robot arm benchmark, the proposed method is naturally extensible to humanoid robot manipulation with more complex execution dependencies, which we leave for future work.

## REFERENCES


Suneel Belkhale and Dorsa Sadigh. Minivla: A better vla with a smaller footprint, 2024. *URL https://github.com/Stanford-ILIAD/openvla-mini*.

Jianxin Bi, Kevin Yuchen Ma, Ce Hao, Mike Zheng Shou, and Harold Soh. Vla-touch: Enhancing vision-language-action models with dual-level tactile feedback. *arXiv preprint arXiv:2507.17294*, 2025.

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

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $pi0$: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PmLR, 2020.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Zihao He, Hongjie Fang, Jingjing Chen, Hao-Shu Fang, and Cewu Lu. Foar: Force-aware reactive policy for contact-rich robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

Jialei Huang, Shuo Wang, Fanqi Lin, Yihang Hu, Chuan Wen, and Yang Gao. Tactile-vla: Unlocking vision-language-action model's physical knowledge for tactile generalization. *arXiv preprint arXiv:2507.09160*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $pi0.5$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025a.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $pi0.5$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025b.

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

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *Conference on Robot Learning*, pp. 2679–2713. PMLR, 2025.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36: 44776–44791, 2023.

Jiaming Liu, Hao Chen, Pengju An, Zhuoyang Liu, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, et al. Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model. *arXiv preprint arXiv:2503.10631*, 2025.

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. *arXiv preprint arXiv:2410.07864*, 2024.

Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control. *Advances in Neural Information Processing Systems*, 37: 4062–4089, 2024.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Yating Wang, Haoyi Zhu, Mingyu Liu, Jiange Yang, Hao-Shu Fang, and Tong He. Vq-vla: Improving vision-language-action models via scaling vector-quantized action tokenizers. *arXiv preprint arXiv:2507.01016*, 2025.

Longyan Wu, Checheng Yu, Jieji Ren, Li Chen, Yufei Jiang, Ran Huang, Guoying Gu, and Hongyang Li. Freetacman: Robot-free visuo-tactile data collection system for contact-rich manipulation. *arXiv preprint arXiv:2506.01941*, 2025.

Ran Xu, Yan Shen, Xiaoqi Li, Ruihai Wu, and Hao Dong. Naturalvlm: Leveraging fine-grained natural language for affordance-guided visual manipulation. *IEEE Robotics and Automation Letters*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jiawen Yu, Hairuo Liu, Qiaojun Yu, Jieji Ren, Ce Hao, Haitong Ding, Guangyu Huang, Guofan Huang, Yan Song, Panpan Cai, et al. Forcevla: Enhancing vla models with a force-aware moe for contact-rich manipulation. *arXiv preprint arXiv:2505.22159*, 2025a.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025b.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30: 495–507, 2021.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

# A More Experimental Results

## A.1 Unseen and Compositional Generalization

We construct two evaluation splits to test generalization beyond execution constraints observed during training. In the *unseen-attribute* split, one value from each execution-constraint family (motion direction, terminal orientation, grasp part, distance, or trajectory pattern) is held out. In the *unseen-composition* split, individual attributes remain observed during training, but selected attribute pairs are held out. All methods use identical demonstrations, observations, fine-tuning steps, and evaluation episodes. We report execution-constraint success rate (ECSR), which measures whether the completed trajectory satisfies the instruction-specified execution constraints.

Table 3: Generalization to unseen execution attributes and unseen attribute compositions. All values are ECSR (%).

| Method | Unseen attribute | Unseen composition | Seen |
|---|---|---|---|
| OpenVLA | 48.5 | 39.0 | 67.0 |
| $\pi_{0.5}$ | 55.8 | 46.2 | 63.2 |
| FAST | 53.8 | 44.6 | 59.6 |
| VQ-VLA (official 4-level) | 56.4 | 48.1 | 61.5 |
| GR00T | 58.1 | 49.2 | 62.0 |
| **ExecVLA** | **66.7** | **59.2** | **70.4** |

ExecVLA improves over the strongest baseline by 8.6 percentage points on unseen attributes and by 10.0 points on unseen compositions. These results support systematic generalization beyond memorizing individual execution terms or combinations.

## A.2 Additional Recent Baselines

We additionally compare with FAST, which tests whether efficient action tokenization alone explains the gain, and GR00T, a recent generalist robot foundation model. All baselines are fine-tuned on the same execution-constraint-rich data for 50K steps with a total batch size of 64 and are evaluated on the same 20 trials per task.

Table 4: Execution-constraint success rate (ECSR, %) against additional recent baselines.

| Method | LIBERO-Goal-Relabeled | Exec-LIBERO | Exec-Realman |
|---|---|---|---|
| OpenVLA | 67.0 | 52.4 | 51.0 |
| $\pi_{0.5}$ | 71.2 | 63.2 | 62.4 |
| FAST | 69.2 | 59.8 | 59.6 |
| VQ-VLA | 70.3 | 61.5 | 61.5 |
| GR00T | 71.4 | 60.9 | 62.0 |
| **ExecVLA** | **76.5** | **70.4** | **65.0** |

FAST remains below VQ-VLA on all three datasets. GR00T is stronger than VQ-VLA on LIBERO-Goal-Relabeled and Exec-Realman, but is slightly weaker on Exec-LIBERO. Both remain below ExecVLA, suggesting that neither efficient tokenization alone nor a stronger generalist backbone fully addresses instruction-specified execution variation.

### A.3 REAL-ROBOT TASK DIVERSITY

The seven tasks reported for Exec-Realman refer to seven high-level task families rather than seven fixed instruction–trajectory pairs. Each family contains more than 20 execution-distinct sub-combinations generated by varying multiple execution attributes while keeping the semantic goal unchanged. These variations include grasped object part, grasp approach or manner, target region, motion direction, and terminal object orientation. For example, "grasp the carrot and place it on the plate" is one high-level family, but its instances combine different grasp regions, grasp approaches, plate regions, and terminal facing directions.

Table 5: Execution-constraint diversity in the Exec-Realman benchmark.

| **Evaluation level** | **Coverage** |
|---|---|
| High-level task families | 7 |
| Execution sub-combinations per family | $> 20$ |
| Total evaluated execution combinations | $> 140$ |
| Main varying attributes | Grasp part/manner, target position, motion direction, terminal orientation |

Table 6: Real-robot evaluation on additional execution-constraint compositions. Results are ECSR (%).

| **Evaluation family** | **New attribute composition** | **Trials** | $\boldsymbol{\pi_{0.5}}$ | **GR00T** | **ExecVLA** |
|---|---|---|---|---|---|
| Obstacle-aware transfer | Detour direction $\times$ target side | 40 | 57.5 | 60.0 | **62.5** |
| Orientation-preserving placement | Grasp manner $\times$ terminal orientation | 40 | 62.5 | 65.0 | **67.5** |
| Drawer extent control | Approach direction $\times$ opening extent | 40 | 60.0 | 62.5 | **65.0** |
| Part-conditioned relocation | Grasp part $\times$ target region $\times$ facing direction | 40 | 65.0 | 67.5 | **70.0** |
| **Average** | All additional families | 160 | 61.3 | 63.8 | **66.3** |

ExecVLA outperforms $\pi_{0.5}$ and GR00T on every additional family, with average gains of 5.0 and 2.5 percentage points, respectively. Thus, Exec-Realman contains dense within-task execution diversity, and the additional evaluation further expands the tested attribute compositions. Whole-body, deformable-object, and mobile-manipulation settings remain outside the evaluated single-arm tabletop scope.

### A.4 BI-LEVEL REPRESENTATION ANALYSIS

We separately decode the goal-level code, the execution-level residual code, and their combination to examine the role of each component.

Table 7: Analysis of the two action-representation levels. Lower normalized $L_1$ reconstruction error is better.

| **Representation** | **Goal success (%)** | **ECSR (%)** | **Normalized $L_1 \downarrow$** |
|---|---|---|---|
| Goal-level code only | 92.8 | 57.6 | 0.029 |
| Execution-level code only | 41.5 | 45.2 | 0.083 |
| **Bi-level codes** | **94.0** | **76.5** | **0.014** |

The goal level retains most goal-completion performance but lacks precise execution control. Adding the execution residual reduces reconstruction error by 51.7% and improves ECSR by 18.9 percentage points, supporting the complementary roles of the two representation levels.

### A.5 Inference Speed Analysis

ExecVLA introduces little inference overhead compared with single-level VQ-VAE and diffusion-based methods. As shown in Table 8, it is slightly slower than VQ-VLA and $\pi_{0.5}$ but remains in the same latency range. The bi-level design operates at the feature and token level and requires only a small increase in output tokens rather than an additional decoding stage.

Table 8: Inference speed comparison.

| Method | Inference time |
|---|---|
| OpenVLA | 237 ms |
| $\pi_{0.5}$ | 86 ms |
| VQ-VLA | 84 ms |
| ExecVLA | 96 ms |

## B Datasets and Visualization

**LIBERO (Simulation).** Build on LIBERO platform Liu et al. (2023), operators teleoperate the end-effector with a SpaceMouse. For each rollout, we record time-stamped RGB and wrist frames, joint states, forward-kinematics end-effector pose, and the action stream. We adopt the same preprocessing pipeline as in Liu et al. (2023), which includes: (1) removing idle segments from the trajectories, (2) resizing all images to a resolution of 256×256 pixels, and (3) rotating each image by 180 degrees. We build two splits:

- **LIBERO-Goal-Relabeled:** LIBERO-GOAL covers 10 tasks and focuses on scenarios where the robot needs to continually acquire new knowledge about motions and behaviors. We retain the visual observations and action trajectories from LIBERO-GOAL and relabel the language instructions into their corresponding execution-constraint-rich versions based on human observation, making each action more specific. For example, "put the wine bottle on the rack" is relabeled as "grab the wine bottle by the neck and place it at the front of the rack." More details can be found in the supplementary material.
- **Exec-LIBERO (ours):** We build upon the LIBERO platform and collect a larger set of execution-constraint-rich tasks in table manipulation scene, consisting 266K frames in total. Each episode is annotated with execution-constraint-rich language instructions and bi-level reasoning texts (goal and execution constraints). As illustrated in Figure 2, our dataset features variations across the entire timeline, with diverse object parts, action constraints, and target relations. Further details are provided in the supplementary material.

**Realman-75 (Real Robot).** We construct the *Exec-Realman* dataset using a Realman-75 robotic arm equipped with a parallel gripper, a calibrated RGB camera observing a fixed tabletop workspace, and an additional wrist-mounted camera. The robot is teleoperated via puppet manipulation, and we collect data for seven tasks featuring rich, fine-grained execution-constraint descriptions. For example, one instruction specifies "grasp the lower half of the carrot and place it on the left side of the plate." Further details about the tasks are provided in the supplementary material.

**Visualization.** As detailed in Tables 12,11, and 10, we have enumerated text examples from our annotated dataset alongside their corresponding bi-level Chain-of-Thought (CoT) annotations. Figure 6 and 7 illustrates the execution outcomes of our method on the Exec-LIBERO and Exec-Realman-75 benchmarks. The results from both benchmarks confirm that our method achieves precise execution of fine-grained execution constraints operations in a consistent environment. **Figure 6** demonstrates the executed tasks, categorized into two groups. The first group (Tasks 7-10) is concerned with opening drawers of different layers to various degrees (completely, halfway, slightly). The second group (Tasks 12,14) focuses on moving a plate following distinct trajectories (leftward then backward, diagonally toward the back-right). **Figure 7** demonstrates the successful execution of two distinct task sets. The first set (Tasks 1, 3) demonstrates relocation of a piece of bread to different specified locations. The second set (Tasks 4-7) shows the grasping of a carrot at different parts (top/bottom) and its placement into different areas of a plate.

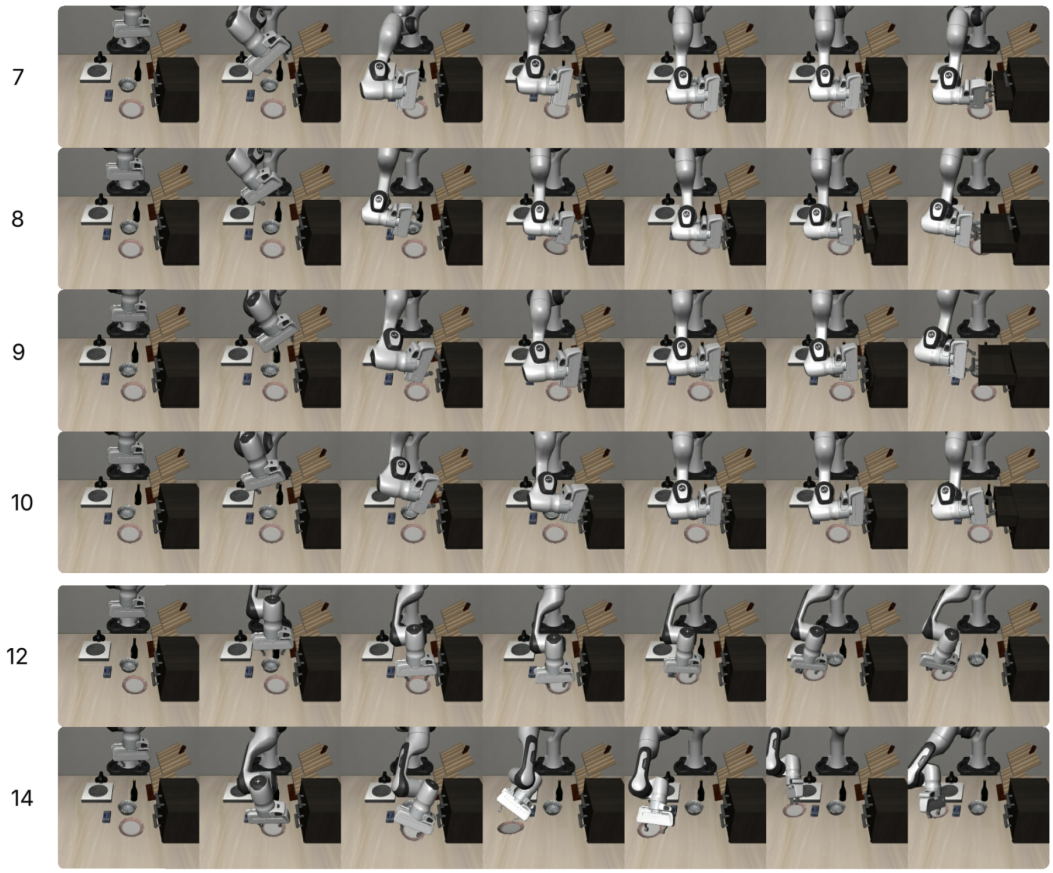



Figure 6: Execution results on the Exec-LIBERO dataset. The serial number corresponds to the entry in Table 11.

Table 9: Summary of our proposed datasets collected via SpaceMouse and puppet manipulation. In each task, we gather 50 demonstrations.

| Split | Setting | Frames | Language | Reasoning Text | Vision | Action Space |
|---|---|---|---|---|---|---|
| LIBERO-Goal-Relabeled | Simulation | 63K | Execution-constraint-rich | Bi-Level | Front, Wrist | EEF Pose |
| Exec-LIBERO | Simulation | 266K | Execution-constraint-rich | Bi-Level | Front, Wrist | EEF Pose |
| Exec-Realman-75 | Real robot | 49K | Execution-constraint-rich | Bi-Level | Front, Wrist | Joint Positions |




Figure 7: Execution results on the Exec-Realman-75 dataset. The serial number corresponds to the entry in Table 12.

Table 10: **LIBERO-Goal-Relabeled Dataset.**

| No. | Execution-constraint instruction | Coarse-grained CoT | Fine-grained CoT |
|---|---|---|---|
| 1 | grasp the left edge of the bowl, move it forward, and place it precisely at the center of the plate | put the bowl on the plate | 1. Robot Action-level Constraint: Trajectory shape - forward linear movement; Contact timing - grasp then place; Ordinal relations - sequential grasp, move, place. 2. Object-level Targeting: Target - bowl's left edge for grasp, plate's center for placement; Pose/Facing/Angle - precise placement with implied alignment. |
| 2 | grab the wine bottle by the neck and place it on the rack | put the wine bottle on the rack | 1. Robot Action-level Constraint: Contact Timing - Must initiate/break contact (grab/release). Trajectory Shape - Requires transfer motion from source (neck) to target (rack). Ordinal Relations - Strict sequence: grab before place. 2. Object-level Targeting: Grasp Target - The neck (specific part) of wine bottle. Place Target - The rack (specific object); final pose constrained by rack structure (must be placed on it). |
| 3 | press the handle to open the top drawer to its maximum extension, then grasp the left edge of the bowl and place it in the center of the drawer | open the top drawer and put the bowl inside | 1. Robot Action-level Constraint: State/Trajectory: Press, then grasp and place. Contact Timing: Press handle first, then grasp bowl after drawer is open. Ordinal Relations: Sequential: open drawer, then grasp bowl, then place bowl. 2. Object-level Targeting: Handle/Surface/Edge: Handle (of drawer), left edge (of bowl). Pose/Facing/Angle: Maximum extension (drawer state), center (target placement pose). |
| 4 | grab the cream cheese by the middle and place it in the center of the bowl | put the cream cheese in the bowl | 1. Robot Action-level Constraint: Trajectory shape - Straight-line placement motion; Contact timing - Grasp first, then place; Ordinal relations - Sequential (grab, then place). 2.Object-level Targeting: Handle/Surface - Target middle of object (cream cheese); Pose/Facing - Orientation-agnostic grasp; Placement - Center of bowl (specific positional target). |
| 5 | hold the wine bottle firmly by the neck and position it on the upper section of the cabinet top, orienting the bottle so that its label is facing to the left | put the wine bottle on top of the cabinet | 1. Robot Action-level Constraint: State/Trajectory: Hold firmly, position on upper section. Contact Timing: Grip neck before positioning. Ordinal Relations: Orient after positioning. 2. Object-level Targeting: Handle/Edge/Surface: Neck (grip), cabinet top (placement). Pose/Facing/Angle: Label facing left. |

*Continued on next page*

| No. | Execution-constraint instruction | Coarse-grained CoT | Fine-grained CoT |
|---|---|---|---|
| 6 | place the end-effector at the center of the plate and move it leftward then backward along the table plane to reposition the plate about close in front of the stove | push the plate to the front of the stove | 1. Robot Action-level Constraint: State/Trajectory: Move end-effector leftward then backward along the table plane. Contact Timing: Maintain contact with the plate. Ordinal Relations: Sequential motion: first leftward, then backward. 2. Object-level Targeting: Target Part: Center of the plate (initial contact). Pose/Facing/Angle: Align end-effector to plate center; movement constrained to table plane. Final Placement: close in front of the stove. |
| 7 | move left to the stove knob, then turn it clockwise to ignite | turn on the stove | Robot Action-level Constraint: Trajectory shape - Linear translation (move left) followed by rotational (turn clockwise). Contact timing - Sequential (move then turn). Ordinal relations - Strict order: translation before rotation. Object-level Targeting: Target object part - Stove knob (specific component). Pose/facing/angle - Clockwise rotation (specific angular direction). |
| 8 | grab the left edge of this bowl and move it along a diagonal path to the center of the stove | put the bowl on the stove | Robot Action-level Constraint: Trajectory shape - Diagonal path; Contact timing - Grab (initial contact) Object-level Targeting: Target part - Left edge; Object - Bowl; Final pose - Center of the stove. |
| 9 | grab the right edge of this bowl and move it along a diagonal path to the upper section of the cabinet | put the bowl on top of the cabinet | 1. Robot Action-level Constraint: Trajectory shape - Diagonal path; Contact timing - After grabbing, then move 2. Object-level Targeting: Target component - Right edge of the bowl; Spatial relation - Upper section of the cabinet. |
| 10 | hold the handle of the middle cabinet drawer and pull it open completely | open the middle drawer of the cabinet | 1. Robot Action-level Constraint: Contact Timing: Initiate contact Trajectory Shape: Linear pulling Ordinal Relations: Grasp then pull 2. Object-level Targeting: Target Part: Middle cabinet drawer handle Action on Target: Pull open completely. |

Table 11: **Exec-LIBERO Dataset**

| No. | Execution-constraint instruction | Coarse-grained CoT | Fine-grained CoT |
|---|---|---|---|
| 1 | grab the right edge of this bowl and move it along a diagonal path to the upper section of the cabinet | put the bowl on top of the cabinet | 1. Robot Action-level Constraint: Trajectory shape - Diagonal path; Contact timing - After grabbing, then move 2. Object-level Targeting: Target component - Right edge of the bowl; Spatial relation - Upper section of the cabinet |
| 2 | grab the left edge of this bowl and move it along a diagonal path to the upper section of the cabinet | put the bowl on top of the cabinet | 1. Robot Action-level Constraint: Trajectory shape - Diagonal path; Contact timing - After grabbing, then move 2. Object-level Targeting: Target component - Left edge of the bowl; Spatial relation - Upper section of the cabinet |
| 3 | grab the right edge of this bowl and move it along a diagonal path to the bottom section of the cabinet | put the bowl on top of the cabinet | 1. Robot Action-level Constraint: Trajectory shape - Diagonal path; Contact timing - After grabbing, then move 2. Object-level Targeting: Target component - Right edge of the bowl; Spatial relation - Bottom section of the cabinet |
| 4 | grab the left edge of this bowl and move it along a diagonal path to the bottom section of the cabinet | put the bowl on top of the cabinet | 1. Robot Action-level Constraint: Trajectory shape - Diagonal path; Contact timing - After grabbing, then move 2. Object-level Targeting: Target component - Left edge of the bowl; Spatial relation - Bottom section of the cabinet |
| 5 | hold the handle of the middle cabinet drawer and pull it open completely | open the middle drawer of the cabinet | 1. Robot Action-level Constraint: Contact Timing: Initiate contact Trajectory Shape: Linear pulling. Ordinal Relations: Grasp then pull 2. Object-level Targeting: Target Part: Middle cabinet drawer handle. Action on Target: Pull open completely |
| 6 | vertically hold the handle of the middle cabinet drawer and pull it open completely | open the middle drawer of the cabinet | 1. Robot Action-level Constraint: Contact Timing: Initiate contact. Trajectory Shape: Linear pulling. Ordinal Relations: Grasp then pull. 2. Object-level Targeting: Target Part: Middle cabinet drawer handle. Action on Target: Pull open completely |
| 7 | vertically hold the handle of the middle cabinet drawer and pull it open halfway | open the middle drawer of the cabinet | 1. Robot Action-level Constraint: Contact Timing: Initiate contact. Trajectory Shape: Linear pulling. Ordinal Relations: Grasp then pull. 2. Object-level Targeting: Target Part: Middle cabinet drawer handle. Action on Target: Pull open halfway |

*Continued on next page*

| No. | Execution-constraint instruction | Coarse-grained CoT | Fine-grained CoT |
|---|---|---|---|
| 8 | vertically hold the handle of the middle cabinet drawer and pull it slightly open | open the middle drawer of the cabinet | 1. Robot Action-level Constraint: Contact Timing: Initiate contact. Trajectory Shape: Linear pulling. Ordinal Relations: Grasp then pull. 2. Object-level Targeting: Target Part: Middle cabinet drawer handle. Action on Target: Pull open slightly |
| 9 | vertically hold the handle of the top cabinet drawer and pull it open completely | open the top drawer of the cabinet | 1. Robot Action-level Constraint: Contact Timing: Initiate contact. Trajectory Shape: Linear pulling. Ordinal Relations: Grasp then pull. 2. Object-level Targeting: Target Part: Top cabinet drawer handle. Action on Target: Pull open completely |
| 10 | vertically hold the handle of the top cabinet drawer and pull it open halfway | open the top drawer of the cabinet | 1. Robot Action-level Constraint: Contact Timing: Initiate contact. Trajectory Shape: Linear pulling. Ordinal Relations: Grasp then pull. 2. Object-level Targeting: Target Part: Top cabinet drawer handle. Action on Target: Pull open halfway |
| 11 | vertically hold the handle of the top cabinet drawer and pull it slightly open | open the top drawer of the cabinet | 1. Robot Action-level Constraint: Contact Timing: Initiate contact. Trajectory Shape: Linear pulling. Ordinal Relations: Grasp then pull. 2. Object-level Targeting: Target Part: Top cabinet drawer handle. Action on Target: Pull open slightly |
| 12 | push the plate and move it leftward then backward along the table plane to reposition the plate about close in front of the stove | push the plate to the front of the stove | 1. Robot Action-level Constraint: State/Trajectory: Move end-effector leftward then backward along the table plane. Contact Timing: Maintain contact with the plate. Ordinal Relations: Sequential motion: first leftward, then backward. 2. Object-level Targeting: Target Part: Center of the plate (initial contact). Pose/Facing/Angle: Align end-effector to plate center; movement constrained to table plane. Final Placement: ~close in front of the stove. |

*Continued on next page*

| No. | Execution-constraint instruction | Coarse-grained CoT | Fine-grained CoT |
|---|---|---|---|
| 13 | push the plate and move it rightward then backward along the table plane to reposition the plate about far in front of the stove | push the plate to the front of the stove | 1. Robot Action-level Constraint: State/Trajectory: Move end-effector rightward then backward along the table plane. Contact Timing: Maintain contact with the plate. Ordinal Relations: Sequential motion: first rightward, then backward. 2. Object-level Targeting: Target Part: Center of the plate (initial contact). Pose/Facing/Angle: Align end-effector to plate center; movement constrained to table plane. Final Placement: ~far in front of the stove. |
| 14 | push the plate and move it diagonally toward the back-right along the table plane to reposition the plate about close in front of the stove | push the plate to the front of the stove | 1. Robot Action-level Constraint: State/Trajectory: Move end-effector diagonally toward back-right along the table plane. Contact Timing: Maintain contact with the plate. Ordinal Relations: Sequential motion: first diagonally back-right. 2. Object-level Targeting: Target Part: Center of the plate (initial contact). Pose/Facing/Angle: Align end-effector to plate center; movement constrained to table plane. Final Placement: ~close in front of the stove. |
| 15 | push the plate and move it diagonally toward the back-right along the table plane to reposition the plate about far in front of the stove | push the plate to the front of the stove | 1. Robot Action-level Constraint: State/Trajectory: Move end-effector diagonally toward back-right along the table plane. Contact Timing: Maintain contact with the plate. Ordinal Relations: Sequential motion: first diagonally back-right. 2. Object-level Targeting: Target Part: Center of the plate (initial contact). Pose/Facing/Angle: Align end-effector to plate center; movement constrained to table plane. Final Placement: ~far in front of the stove. |
| 16 | push the plate and move it rightward then backward along the table plane to reposition the plate about close in right of the stove | push the plate to the right the stove | 1. Robot Action-level Constraint: State/Trajectory: Move end-effector rightward then backward along the table plane. Contact Timing: Maintain contact with the plate. Ordinal Relations: Sequential motion: first rightward, then backward. 2. Object-level Targeting: Target Part: Center of the plate (initial contact). Pose/Facing/Angle: Align end-effector to plate center; movement constrained to table plane. Final Placement: ~close in right of the stove. |

*Continued on next page*

| No. | Execution-constraint instruction | Coarse-grained CoT | Fine-grained CoT |
|---|---|---|---|
| 17 | push the plate and move it diagonally toward the back-right along the table plane to reposition the plate about close in right of the stove | push the plate to the right the stove | 1. Robot Action-level Constraint: State/Trajectory: Move end-effector diagonally toward back-right along the table plane. Contact Timing: Maintain contact with the plate. Ordinal Relations: Sequential motion: first diagonally back-right.2. Object-level Targeting: Target Part: Center of the plate (initial contact). Pose/Facing/Angle: Align end-effector to plate center; movement constrained to table plane. Final Placement: ~close in right of the stove. |
| 18 | grab the wine bottle by the neck and place it on the rack | put the wine bottle on the rack | 1. Robot Action-level Constraint: Contact Timing - Must initiate/break contact (grab/release). Trajectory Shape - Requires transfer motion from source (neck) to target (rack). Ordinal Relations - Strict sequence: grab before place. 2. Object-level Targeting: Grasp Target - The neck (specific part) of wine bottle. Place Target - The rack (specific object); final pose constrained by rack structure (must be placed on it). |
| 19 | grab the wine bottle by the body and place it at the front of the rack | put the wine bottle on the rack | 1. Robot Action-level Constraint: Contact Timing - Must initiate/break contact (grab/release). Trajectory Shape - Requires transfer motion from source (body) to target (rack front). Ordinal Relations - Strict sequence: grab before place. 2. Object-level Targeting: Grasp Target - The body (specific part) of wine bottle. Place Target - The front of the rack (specific location); final pose constrained by rack structure (must be placed on it). |
| 20 | grab the wine bottle by the neck and place it at the back of the rack | put the wine bottle on the rack | 1. Robot Action-level Constraint: Contact Timing - Must initiate/break contact (grab/release). Trajectory Shape - Requires transfer motion from source (neck) to target (rack back). Ordinal Relations - Strict sequence: grab before place. 2. Object-level Targeting: Grasp Target - The neck (specific part) of wine bottle. Place Target - The back of the rack (specific location); final pose constrained by rack structure (must be placed on it). |

*Continued on next page*

| No. | Execution-constraint instruction | Coarse-grained CoT | Fine-grained CoT |
|---|---|---|---|
| 21 | grab the wine bottle by the body and place it at the back of the rack | put the wine bottle on the rack | 1. Robot Action-level Constraint: Contact Timing - Must initiate/break contact (grab/release). Trajectory Shape - Requires transfer motion from source (body) to target (rack back). Ordinal Relations - Strict sequence: grab before place. 2. Object-level Targeting: Grasp Target - The body (specific part) of wine bottle. Place Target - The back of the rack (specific location); final pose constrained by rack structure (must be placed on it). |
| 22 | grab the right edge of this bowl and move it along a diagonal path to the center of the stove | put the bowl on the stove | Robot Action-level Constraint: Trajectory shape - Diagonal path; Contact timing - Grab (initial contact) Object-level Targeting: Target part - Right edge; Object - Bowl; Final pose - Center of the stove |
| 23 | grab the left edge of this bowl and move it along a diagonal path to the center of the stove | put the bowl on the stove | Robot Action-level Constraint: Trajectory shape - Diagonal path; Contact timing - Grab (initial contact). Object-level Targeting: Target part - Left edge; Object - Bowl; Final pose - Center of the stove |
| 24 | hold the wine bottle firmly by the neck and position it on the upper section of the cabinet top, orienting the bottle so that its label is facing to the left | put the wine bottle on top of the cabinet | 1. Robot Action-level Constraint: State/Trajectory: Hold firmly, position on upper section. Contact Timing: Grip neck before positioning. Ordinal Relations: Orient after positioning. 2. Object-level Targeting: Handle/Edge/Surface: Neck (grip), cabinet top (placement). Pose/Facing/Angle: Label facing left. |
| 25 | hold the wine bottle firmly by the neck and position it on the bottom section of the cabinet top, orienting the bottle so that its label is facing to the right | put the wine bottle on top of the cabinet | 1. Robot Action-level Constraint: State/Trajectory: Hold firmly, position on lower section. Contact Timing: Grip neck before positioning. Ordinal Relations: Orient after positioning. 2. Object-level Targeting: Handle/Edge/Surface: Neck (grip), cabinet top (placement). Pose/Facing/Angle: Label facing right. |
| 26 | hold the wine bottle firmly by the neck and position it on the upper section of the cabinet top, orienting the bottle so that its label is facing to the left | put the wine bottle on top of the cabinet | 1. Robot Action-level Constraint: State/Trajectory: Hold firmly, position on upper section. Contact Timing: Grip neck before positioning. Ordinal Relations: Orient after positioning. 2. Object-level Targeting: Handle/Edge/Surface: Neck (grip), cabinet top (placement). Pose/Facing/Angle: Label facing left. |

*Continued on next page*

| No. | Execution-constraint instruction | Coarse-grained CoT | Fine-grained CoT |
|---|---|---|---|
| 27 | hold the wine bottle firmly by the neck and position it on the bottom section of the cabinet top, orienting the bottle so that its label is facing to the left | put the wine bottle on top of the cabinet | 1. Robot Action-level Constraint: State/Trajectory: Hold firmly, position on lower section. Contact Timing: Grip neck before positioning. Ordinal Relations: Orient after positioning. 2. Object-level Targeting: Handle/Edge/Surface: Neck (grip), cabinet top (placement). Pose/Facing/Angle: Label facing left. |
| 28 | hold the wine bottle firmly by the neck and position it on the upper section of the cabinet top, orienting the bottle so that its label is facing to the front | put the wine bottle on top of the cabinet | 1. Robot Action-level Constraint: State/Trajectory: Hold firmly, position on upper section. Contact Timing: Grip neck before positioning. Ordinal Relations: Orient after positioning. 2. Object-level Targeting: Handle/Edge/Surface: Neck (grip), cabinet top (placement). Pose/Facing/Angle: Label facing front. |
| 29 | hold the wine bottle firmly by the neck and position it on the bottom section of the cabinet top, orienting the bottle so that its label is facing to the front | put the wine bottle on top of the cabinet | 1. Robot Action-level Constraint: State/Trajectory: Hold firmly, position on lower section. Contact Timing: Grip neck before positioning. Ordinal Relations: Orient after positioning. 2. Object-level Targeting: Handle/Edge/Surface: Neck (grip), cabinet top (placement). Pose/Facing/Angle: Label facing front. |
| 30 | hold the wine bottle firmly by the neck and position it on the upper section of the cabinet top, orienting the bottle so that its label is facing to the back | put the wine bottle on top of the cabinet | 1. Robot Action-level Constraint: State/Trajectory: Hold firmly, position on upper section. Contact Timing: Grip neck before positioning. Ordinal Relations: Orient after positioning. 2. Object-level Targeting: Handle/Edge/Surface: Neck (grip), cabinet top (placement). Pose/Facing/Angle: Label facing back. |
| 31 | hold the wine bottle firmly by the neck and position it on the bottom section of the cabinet top, orienting the bottle so that its label is facing to the back | put the wine bottle on top of the cabinet | 1. Robot Action-level Constraint: State/Trajectory: Hold firmly, position on lower section. Contact Timing: Grip neck before positioning. Ordinal Relations: Orient after positioning. 2. Object-level Targeting: Handle/Edge/Surface: Neck (grip), cabinet top (placement). Pose/Facing/Angle: Label facing back. |

Table 12: **Exec-Realman Dataset**

| No. | Execution-constraint instruction | Coarse-grained CoT | Fine-grained CoT |
|---|---|---|---|
| 1 | pick up the bread and move it to the left front | move the bread | Robot Action: Trajectory Shape - To the left front; Ordinal Relations - Pick first, then move. Object Targeting: Target Object - The bread; Spatial Relation - Move it to a location (left front). |
| 2 | pick up the bread and move it to the right front | move the bread | Robot Action: Trajectory Shape - To the right front; Ordinal Relations - Pick first, then move. Object Targeting: Target Object - The bread; Spatial Relation - Move it to a location (right front). |
| 3 | pick up the bread and move it to the front | move the bread | Robot Action: Trajectory Shape - To the front; Ordinal Relations - Pick first, then move. Object Targeting: Target Object - The bread; Spatial Relation - Move it to a location (front). |
| 4 | hold the carrot by the upper part and place it on the right side of the plate | place the carrot in the plate | Robot Action-level Constraint: State/Trajectory - Move carrot from held to placement pose. Ordinal Relations - Grasp first, then place. Object-level Targeting: Carrot - Target handle (upper part). Plate - Target surface (top side). Spatial Relation - Final pose on the right side of the plate. |
| 5 | hold the carrot by the upper part and place it on the left side of the plate | place the carrot in the plate | Robot Action-level Constraint: State/Trajectory - Move carrot from held to placement pose. Ordinal Relations - Grasp first, then place. Object-level Targeting: Carrot - Target handle (upper part). Plate - Target surface (top side). Spatial Relation - Final pose on the left side of the plate. |
| 6 | hold the carrot by the lower part and place it on the left side of the plate | place the carrot in the plate | Robot Action-level Constraint: State/Trajectory - Move carrot from held to placement pose. Ordinal Relations - Grasp first, then place. Object-level Targeting: Carrot - Target handle (lower part). Plate - Target surface (top side). Spatial Relation - Final pose on the left side of the plate. |
| 7 | hold the carrot by the lower part and place it on the right side of the plate | place the carrot in the plate | Robot Action-level Constraint: State/Trajectory - Move carrot from held to placement pose. Ordinal Relations - Grasp first, then place. Object-level Targeting: Carrot - Target handle (lower part). Plate - Target surface (top side). Spatial Relation - Final pose on the right side of the plate. |