GuidedVLA: Specifying Task-Relevant Factors via Plug-and-Play Action Attention Specialization

Author Names Omitted for Anonymous Review. Paper-ID 74

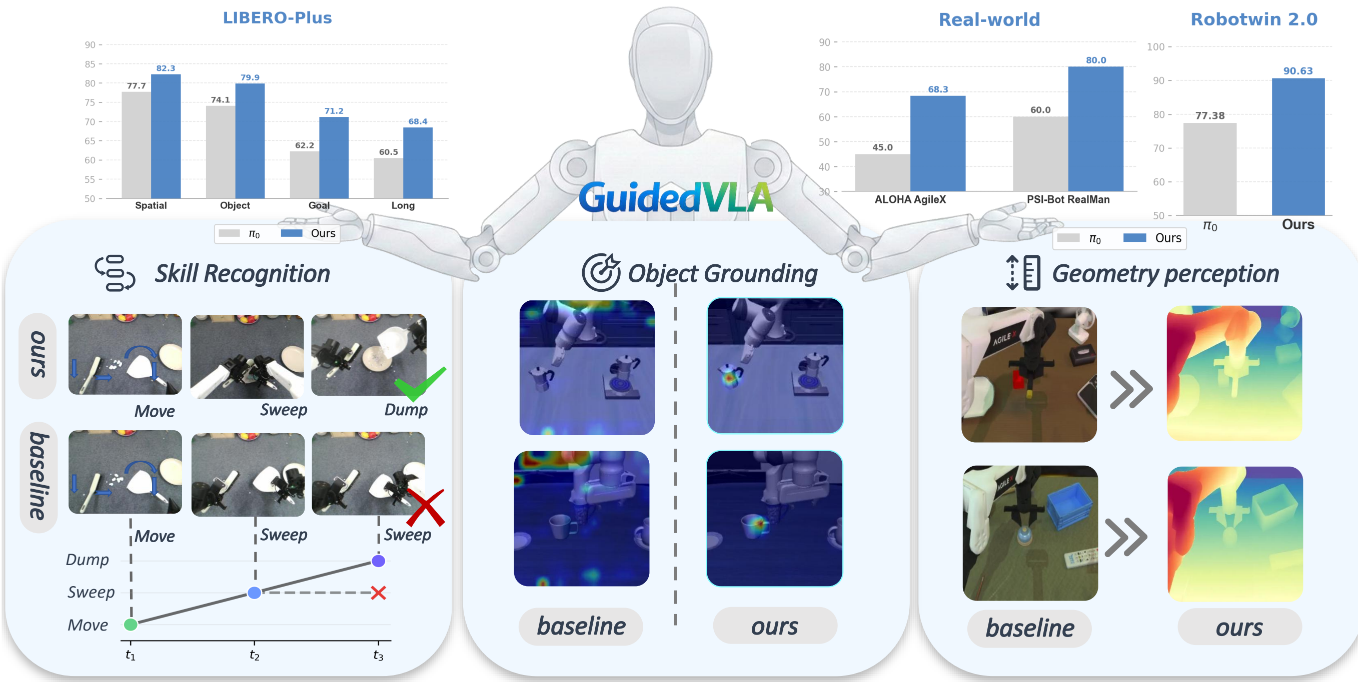


Fig. 1: We present **GuidedVLA**, a VLA paradigm, where the action decoder is explicitly guided to capture task-relevant information such as object grounding, spatial geometry, and temporal skill logic. Across simulation and real-robot experiments, GuidedVLA significantly improves success rates in both in-domain and out-of-domain settings, demonstrates effectiveness of specifying attention heads of the action decoder with explicit guidance.

***Abstract*—Vision-Language-Action (VLA) models aim for general robot learning by aligning action as a modality within powerful Vision-Language Models (VLM). Existing VLAs rely on end-to-end supervision to implicitly enable the action decoding process to learn task-relevant features. However, without explicit guidance, these models often overfit to spurious correlations, such as visual shortcuts or environmental noise, limiting their generalization. In this paper, we introduce GuidedVLA, a framework designed to manually guide the action generation to focus on task-relevant factors. Our core insight is to treat the action decoder not as a monolithic learner, but as an assembly of functional components. Individual attention heads are supervised by manually defined auxiliary signals to capture distinct factors. As an initial study, we instantiate this paradigm with three specialized heads: object grounding, spatial geometry, and temporal skill logic. Across simulation and real-robot experiments, GuidedVLA improves success rates in both in-domain and out-of-domain settings compared to strong VLA baselines. Finally, we show that the quality of these specialized factors correlates positively with task performance and that our mechanism yields decoupled, high-quality features. Our results suggest that explicitly guiding the learning of decision making for action decoder is a promising direction for building more robust and general VLA models.**

I. Introduction

Vision-Language-Action (VLA) models [99, 44, 8] represent a significant step toward generalist robot policies by integrating action as a specialized modality within the rich feature space of Vision-Language Models (VLMs). By leveraging the massive pre-training of VLMs [77, 78, 4, 50], these models can inherit high-level semantic knowledge and reasoning capabilities essential for complex tasks. However, current VLA training typically relies on end-to-end supervision where the action decoder is expected to implicitly learn task-relevant factors from demonstration data [10]. According to pioneering studies in the field of computer vision and imitation learning, end-to-end learning without explicit guidance may lead to short cut learning [29, 28] or causal confusion [20].

In practice, we observe that the action decoder of VLA often latches onto spurious correlations, such as background textures or incidental camera artifacts, as shown in Fig. 1. While some cross-attention heads in action decoder occasionally attend to relevant regions, this behavior is highly stochastic and varies across different heads and scenarios. This randomness suggests that though VLM backbones provide a robust feature stream, **the action decoder does not learn a stable, causal understanding of the task, but instead relies on a shifting set of features** and thus struggle to generalize.

Considering only end-to-end supervision makes the decision-making process of VLA opaque and inconsistent, to address this inconsistency, we propose **GuidedVLA**, a framework that transforms the action decoder from a monolithic learner into an assembly of functionally specialized components. Instead of allowing the cross-attention heads to develop roles implicitly, we manually specify the information each head should capture by supervising them with distinct, task-relevant auxiliary signals.

While this paradigm is designed to be general and extensible, in this work, we instantiate it by supervising three primary factors, as in Fig. 1: (i) **object grounding**, ensuring action tokens attend to task-relevant regions,; (ii) **skill recognition**, enabling action tokens to identify the intended sub-skill or phase of a multi-step behavior; and (iii) **geometry perception**, allowing action tokens to leverage 3D spatial information. Our probing experiments reveal that current VLAs are brittle across all three factors, and that the proposed guidance mechanism effectively resolves these deficiencies.

Across multiple simulation benchmarks and real-world experiments, GuidedVLA achieves a significant performance boost for π_0 [8], surpassing other recent feature-training methods in the field [92, 74]. Furthermore, we provide a quantitative evaluation showing a strong positive correlation between factor understanding and overall success rates. Finally, we validate that partitioning factors into specialized attention heads shows decoupled Features and produces better-decoupled features compared to a mixture approach where all heads are jointly supervised.

In summary, we make the following contributions:

- We propose GuidedVLA, a general paradigm for VLA that mitigates overfit risk by specifying task-relevant factors through functional attention specialization.
- We instantiate this framework by designing three specialized heads: object grounding, skill recognition, and geometry perception and demonstrate through probing that such explicit guidance resolves the inherent brittleness and stochasticity of unguided VLA decoders.
- We provide extensive evaluations across multiple simulation benchmarks and real-robot tasks, showing that GuidedVLA significantly improve state-of-the-art baselines in both in-domain and out-of-distribution scenarios.
- We offer quantitative insights into head specialization, validating that our approach yields high-quality features that correlate positively with task performance.

II. Related Work

Vision-Language-Action (VLA) Models: Vision Language Action (VLA) models aim to map visual observations and language instructions to low level robot actions by combining pretrained vision language models with large scale robot demonstrations [2, 10, 99, 22, 30, 44, 6, 97]. One important research direction focuses on scaling embodied data through multi source datasets [66, 80, 42, 19, 62, 39, 47], standardized multi task benchmarks [63, 58, 27, 38, 88], and evaluations under distribution shift [27, 64, 23]. Another line of work improves training and inference recipes, including multimodal prompting [41, 24], parameter efficient adaptation [43, 84, 73, 33], and inference time acceleration [85, 9, 40, 61, 95]. In parallel, prior work strengthens the action pathway through alternative action parameterizations and learning objectives, including diffusion or flow based generation [17, 8, 7, 59, 16, 54, 14, 83], action chunking for temporal abstraction [94], and discrete or compressed action tokenizers to better match control bandwidth [68, 82].

Auxilary Tasks for Robotics Models Structured intermediate representations improve policy robustness under distribution shift. Object centric methods factor manipulation around task relevant entities such as object poses, keypoints, and relations [35, 31, 57, 32, 86, 52, 49, 48, 67, 13]. Skill based representations support long horizon reasoning by decomposing tasks into reusable subgoals [2, 53, 60, 34, 26, 93, 1]. Geometry aware policies that operate on 3D representations, including point clouds and 3D scene tokens, achieve strong viewpoint and instance generalization [90, 81, 55, 96, 70, 18, 87, 69, 21, 25, 75, 89, 46, 51, 45, 36, 65, 98, 5]. Our work complements these approaches by supervising object centric structure, skills, and geometry into separable internal pathways within the action decoder of VLA.

III. Method

Algorithm 1 Decoupled Attention with Guided Heads

Require: Action hidden states X_L at layer L
Require: Attention head indices: $\mathcal{H}_{object}$, $\mathcal{H}_{skill}$, $\mathcal{H}_{depth}$
Require: VLM cache (K_L, V_L); Depth cache (K^d, V^d)
Ensure: Fused attention output A_L

$Q \leftarrow \mathrm{Proj_Q}(X_L)$

Stage 1: Factor-Specific Cross-Attention
$A[\mathcal{H}_{object}] \leftarrow \mathrm{Attn}(Q[\mathcal{H}_{object}], K_L, V_L)$ ▷ Object Head
$A[\mathcal{H}_{skill}] \leftarrow \mathrm{Attn}(Q[\mathcal{H}_{skill}], K_L, V_L)$ ▷ Skill Head
$A[\mathcal{H}_{depth}] \leftarrow \mathrm{Attn}(Q[\mathcal{H}_{depth}], K^d, V^d)$ ▷ Depth Head

Stage 2: Per-Head Supervision
Apply $\mathcal{L}_{object}$ (Eq. 2) to $A[\mathcal{H}_{object}]$
Apply $\mathcal{L}_{skill}$ (Eq. 4) to $A[\mathcal{H}_{skill}]$

Stage 3: Control-Net Fusion
$A_L^{\text{specified}} \leftarrow \mathrm{Proj_O}\big(\mathrm{Concat}(A[:])\big)$
$A_L \leftarrow \mathrm{ZeroConv}(A_L^{\text{specified}}) + A_L^{\text{main}}$ ▷ Merge with Main Branch

A. Problem Setup and Motivation

Recent representative Vision-Language-Action (VLA) models [8, 44, 99] extend Vision-Language Models (VLMs) by introducing action tokens $\boldsymbol{a}$ alongside vision tokens $\boldsymbol{v}$ and language tokens $\boldsymbol{l}$. The action generation process is trained to regress robot trajectories by denoising $\boldsymbol{a}$ conditioned on $(\boldsymbol{v}, \boldsymbol{l})$.

Although VLMs provide rich semantic features in $(\boldsymbol{v}, \boldsymbol{l})$, the action tokens $\boldsymbol{a}$ do not inherently learn to extract task-critical information. As in Figure 1, action token attention often

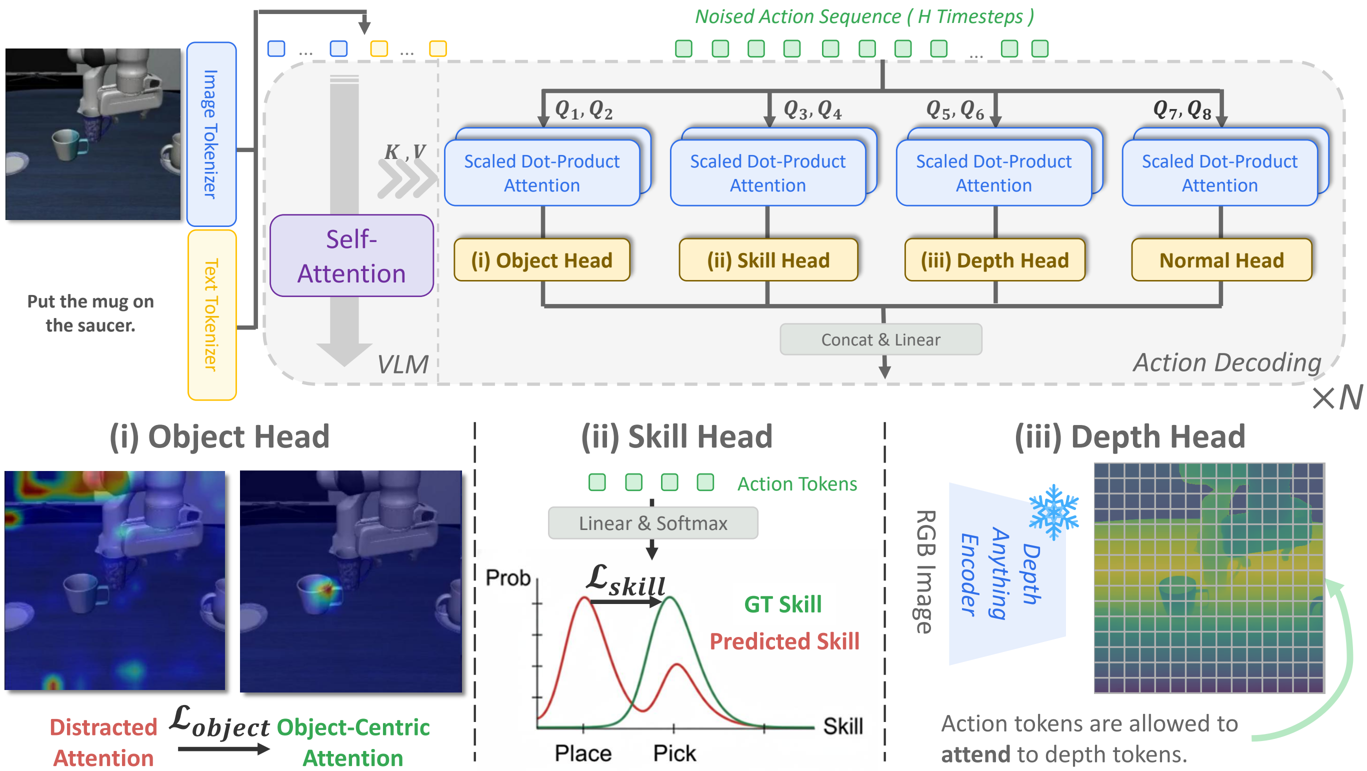


Fig. 2: **Architecture of GuidedVLA.** We introduce explicit, structured guidance into the multi-head attention layers of the VLA action decoder. Instead of relying on implicitly entangled representations, we repurpose dedicated attention heads to specialize in distinct task-relevant factors: **(i) Object Head** supervises its attention maps to explicitly ground task-relevant objects and suppress distractors via $\mathcal{L}_{object}$; **(ii) Skill Head** aligns internal feature representations with temporal skill phases (e.g., Pick → Place) through auxiliary classification $\mathcal{L}_{skill}$; **(iii) Depth Head** injects geometric cues via cross attention only to features from a depth encoder. These guidance forces the policy to explicitly aware spatial, temporal, and geometric structures.

diffuses over irrelevant background regions. This motivates our central designs: ***How can we guide action decoding to extract task-relevant information?***

B. *What to Guide: Three Task-Relevant Factors*

We identify three factors correlated with robotics tasks, based on our preliminary probing experiments in Sec. V-B:

1) **Object Grounding**: whether action tokens can attend to the correct task-relevant regions (e.g., affordance).
2) **Skill Recognition**: ahether action tokens correctly identifies current sub-skill or temporal phase within a complex robotics task (e.g., long-horizon).
3) **Geometry Perception**: whether action tokens can utilize 3D spatial information when performing fine-grained tasks (e.g., click bell).

These factors are complementary: grounding localizes the target, skill recognition defines the behavior, and geometry provides the spatial constraints for execution. Together, they constitute a comprehensive semantic interface between high-level VLM representations and low-level control.

C. *How to Guide: Attention Head Specialization*

To capture decoupled task-relevant information,the Multi-Head Attention (MHA) [79] adopted in the action decoder offers a natural solution with minium structure modification: we explicitly assign specific heads to capture certain factors by **applying different supervision signals on different heads**.

Further, as there already exists large-scaled pretrained backbones [44, 8], we could also empower them with specified heads without catastrophic forgetting pre-trained capabilities, thanks to the natural decoupling characteristics of multi-head attention. Specifically, for those pretrained backbones, we propose a **ControlNet-style** [91] residual adapter strategy. To add a supervised head $\mathrm{Attn}_{\mathrm{specified}}$, we introduce a zero-initialized projection $\mathrm{ZeroConv}$ before fusing with the main branch attnetion features $\mathrm{Attn}_{\mathrm{main}}$:

$$\mathrm{Attn}(\mathbf{x}) = \mathrm{Attn}_{\mathrm{main}}(\mathbf{x}) + \mathrm{ZeroConv}\left(\mathrm{Attn}_{\mathrm{specified}}(\mathbf{x})\right). \quad (1)$$

Since $\mathcal{Z}$ is initialized to zero, the control branch initially contributes no signal. This ensures the model retains its pre-trained behavior at the start of training, while gradually learning to inject factor-specific biases as optimization proceeds.

We now describe the specific objectives and mechanisms for the three specific factors, as in Alg. 1.

1) Object Head (Visual Grounding): Intuitively, action decoding benefits from attending to semantically meaningful regions, such as the object to be grasped and the target destination. To enforce this, we guide a subset of heads $\mathcal{H}_{\mathrm{obj}}$ to align their attention maps with ground-truth region masks. Given action tokens' attention weights for VLM features $\boldsymbol{A}$, and labels for attention weights $\boldsymbol{A}^{\mathrm{GT}}$, we minimize weighted

TABLE I: **LIBERO-Plus Benchmark Results.** The proposed model achieves the highest average success rate, with a significant boost compared to its base model π_0. Notably, **single-head ablations reveal task-specific alignment**: the object head excels in the *Goal* suite (requiring precise target grounding), the skill head dominates the *Long* suite (requiring sequential temporal consistency), and the depth head performs well on the Spatial and Object suite (requiring 3D understanding).

	Perturbation Dimensions							Task Suites				
Model	**Camera**	**Robot**	**Language**	**Light**	**Background**	**Noise**	**Layout**	**Spatial**	**Object**	**Goal**	**Long**	**Total**
OpenVLA [44]	0.8	3.5	23.0	8.1	34.8	15.2	28.5	19.4	14.0	15.1	14.3	15.6
OpenVLA-OFT [43]	56.4	31.9	79.5	88.7	93.3	75.8	74.2	84.0	66.5	63.0	66.4	69.6
NORA [37]	2.2	37.0	65.1	45.7	58.6	12.8	62.1	47.6	34.4	38.8	36.3	39.0
WorldVLA [12]	0.1	27.9	41.6	43.7	17.1	10.9	38.0	32.5	28.6	31.8	8.2	25.0
UniVLA [11]	1.8	46.2	69.6	69.0	81.0	21.2	31.9	55.5	36.7	40.7	39.9	43.9
π_0-Fast [68]	65.1	21.6	61.0	73.2	73.2	74.4	68.8	74.4	72.7	57.5	43.4	61.6
RIPT-VLA [76]	55.2	31.2	77.6	88.4	91.6	73.5	74.2	85.8	64.3	58.0	67.5	68.4
DreamVLA [92]	65.0	40.8	63.5	85.7	82.6	84.9	74.0	79.7	79.0	61.7	59.8	69.9
AdaMoE [74]	53.8	17.5	20.6	73.7	73.8	58.6	65.8	51.0	57.9	53.3	38.1	50.1
π_0 [8]	62.3	39.8	63.1	86.0	82.8	82.4	69.6	77.7	74.1	62.2	60.5	68.2
w/ object head	68.2	40.0	62.1	91.4	87.2	85.0	76.5	77.4	78.8	67.5	62.7	71.5
w/ skill head	69.3	40.5	63.2	90.2	87.6	85.5	75.5	79.8	78.6	66.6	63.6	71.8
w/ depth head	68.1	43.9	65.8	90.7	83.4	85.6	72.8	81.4	79.0	65.4	61.8	71.7
w/ all heads **(Ours)**	**70.8**	**49.4**	**66.8**	**92.9**	**88.1**	**89.3**	**78.4**	**82.3**	**79.9**	**71.2**	**68.4**	**75.4**

negative log likelihood loss:

$$\mathcal{L}_{\text{object}} = -\frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} a \log a^{\text{GT}}. \quad (2)$$

To construct $\boldsymbol{A}^{\text{GT}}$, we use foundation models like grounding SAM [72] to annotate the object to be grasped or the target destination as interested areas and then set all the other tokens as non-interested. We only set attention weight $\boldsymbol{A}^{\text{GT}}$ of non-interested tokens to be 0 while not applying loss on interested areas so that the model can flexibly decide how to attend to interested areas. Check Appendix for more details.

*2) **Skill Head (Temporal Logic Intent):*** Skills capture high-level, temporally extended semantics that modulate the model's action behaviors in long-horizon tasks. To encode this, we designate a subset of heads $\mathcal{H}_{\text{skill}}$ to specialize in intent recognition. We apply supervision *independently* to each selected head to classify current intent types. For a head $h \in \mathcal{H}_{\text{skill}}$, we project its output feature $\mathbf{f}_h$ to a skill probability distribution $\hat{\mathbf{p}}_h$ and apply a KL-divergence loss against the ground-truth soft label $\mathbf{y}$, which represents the skill distribution over a future horizon:

$$\hat{\mathbf{p}}_h = \text{softmax}(\boldsymbol{W}\mathbf{f}_h + \mathbf{b}) \quad (3)$$

$$\mathcal{L}_{\text{skill}} = \sum_k y_k \left(\log y_k - \log \hat{\mathbf{p}}_h\right) \quad (4)$$

Regarding the annotation of skill types at each time-step, we combine foundation models and manual correction. Please check Appendix for full details.

*3) **Depth Head (3D Structure):*** Since standard vision encoders (e.g., SigLIP) in VLA [8] is trained with 2D supervision and lacks explicit 3D awareness, we design specialized depth heads. Instead of a loss, we use a structural constraint: we extract features from a frozen depth encoder (e.g., DA3 [56]), F_{Depth} and project them into depth-aware keys and values, K_{Depth} and V_{Depth}. We then constrain a subset of heads $\mathcal{H}_{\text{depth}}$ to only attend to these depth-derived tokens:

$$\mathcal{H}_{\text{depth}} : \text{softmax}\left(\frac{Q^{\text{Depth}}(K_{\text{Depth}})^\top}{\sqrt{d_h}}\right) V_{\text{Depth}}. \quad (5)$$

This design forces specific heads to specialize in 3D geometry processing. Please check Appendix for more details.

In summary, we adopt a mixed loss:

$$\mathcal{L} = \mathcal{L}_{\text{FM}} + \mathcal{L}_{\text{object}} + \mathcal{L}_{\text{skill}}. \quad (6)$$

where $\mathcal{L}_{\text{FM}}$ is the flow matching loss, $\mathcal{L}_{\text{object}}$ and $\mathcal{L}_{\text{skill}}$ supervise distinct subsets of attention heads. For geometric perception, we inject depth keys and queries for depth heads rather than using a loss term. The remaining unsupervised heads are left free to capture purely data-driven patterns, preserving the model's flexibility and expressivity, as in Fig. 2.

D. Guidance Dataset Construction

For training dataset construction, the implementation details of our stage-aware object-mask and skill-label construction pipeline are provided in Appendix.

IV. Experiments

A. Simulation Experiments

We evaluate our method on two simulation benchmarks.

LIBERO-Plus [27] is a robustness-oriented benchmark built upon LIBERO [58]. It is designed to **evaluate generalist manipulation policies under distribution shifts**. It introduces perturbations along seven dimensions: camera viewpoint, robot initial state, language variation, lighting condition, background texture, sensor noise, and object layout to expose failure modes under generalization scenario beyond in-domain evaluation. We compare with state-of-the-art baselines in Table I.

Robotwin 2.0 [15] offers a multi-task evaluation platform across diverse robot embodiments, and leverages extensive scene/object randomization to scale data and enable out-of-distribution testing. As in Fig. 3, we evaluate on eight representative tasks **under randomized, unseen settings (out-of-domain task instructions, environments, and object placements)** using the Agilex Piper dual-arm setup.

B. Real-World Experiments

We conduct real-world experiments on two dual-arm platforms to evaluate both in-domain action generation and cross-platform generalization against baselines.

Platforms: Platform A is a ALOHA AgileX dual-arm system, equipped with two Orbbec Dabai wrist cameras (one per arm) and an additional Orbbec Dabai third-person camera. Platform B is a PSI-Bot dual-arm platform, using Intel RealSense D435 cameras for visual observations. Figure 4 summarize the hardware setups and qualitative task rollouts.

Tasks: On ALOHA AgileX, we design three household tasks: (1) *pick up fruits and vegetables*: classify and place pepper/carrot on plate, strawberry in bowl, (2) *stack the bowls*: assemble two bowls and place on rack, and (3) *clean the tabletop*: sweep trash with broom/dustpan, pour into tray. On PSI-Bot RealMan, we design three tasks in chemistry lab: (4) *pick up beaker*: grasp and lift a beaker to heating mantle, (5) *stack beakers*: nest small beakers inside large one, and (6) *heat beaker*: place beaker on asbestos mesh atop iron stand.

Evaluation protocol: For each task and model, we perform 20 trials. A trial is successful if the entire task is completed.

Generalization evaluation: We evaluate three generalization settings on both platforms: *in-domain generalization*, *scene generalization*, and *lighting generalization*. Here, *in-domain generalization* focuses on object position variations within the training distribution, while preserving task semantics and scene layout. *Scene generalization* introduces distracting objects into the workspace, testing robustness to clutter and semantic interference. *Lighting generalization* varies illumination intensity and color temperature, assessing sensitivity to perceptual shifts. Results are summarized in Table II.

V. Analysis

In this section, we aim to answer the following questions:

1) Do VLAs under-utilize vision-language representations in action decoding process, and can explicit factor guidance close this gap? (Section V-B)
2) Does our proposed GuidedVLA improve baseline performance under both in-distribution and out-of-distribution evaluations? (Section V-A)
3) Which factors (object, skill, geometry) matter most for which task types? (Section V-A)
4) Does our attention head specialization indeed lead to learning decoupled features? (Section V-C)
5) How different architectural choices for guidance influence performance? (Section V-E)

A. Task-suite Analysis and Cross-benchmark Generalization

We analyze how each factor contributes to different task suites and use representative results from simulation and real-world evaluations to explain *why* each specified head helps.

Object Head: Visual Generalization. Tasks involving clutter or distractors necessitate a precise understanding of object instance identities. On the LIBERO-Plus *Goal* suite, which requires distinguishing a target object from visually similar distractors, the object head yields the highest single-head gain (+5.3% over π_0, Table I). This aligns with the intuition that explicit object-centric representations mitigate grounding failures, allowing the policy to filter out irrelevant visual cues that confuse the baseline.

Skill Head: Temporal Coherence. Long-horizon manipulation requires maintaining "stage awareness" to transition correctly between sub-skills. The skill head dominates the multi-stage LIBERO-Plus *Long* suite (63.6%, Table I), outperforming other variants. Similarly, for the *Lift Pot* task (Robotwin2.0) involving a strict sequence of grasping, stabilizing, and lifting, this head achieves near-perfect success (99%, Figure 3). These results validate that explicit skill recognitions provide the temporal scaffolding needed to prevent premature termination or mode collapse during phase transitions.

Depth Head: Geometric Precision. Tasks reliant on precise 3D localization, such as pressing or insertion, necessitate accurate depth estimation which 2D backbone alone often fails to provide. On Robotwin2.0, the *Click Bell* task requires precise Z-axis control to trigger the mechanism without collision; the depth head drastically improves performance from 35% to 63% (Figure 3). We observe a similar trend in *Beat Block Hammer* (78% $\rightarrow$ 99%), where height alignment is critical. These gains confirm that explicit geometric cues compensate for the lack of 3D observability in standard VLA inputs.

Full Model: Synergetic Generalization. The full model integrates these complementary strengths—visual grounding, temporal coherence, and geometric precision—to achieve robust generalization across diverse domains. It raises the average success on Robotwin2.0 from 77.38% to 90.63% (Figure 3) and demonstrates superior robustness in the real world (Table II). Crucially, while single heads excel in their respective niches, the full model is the only variant that reliably generalizes across *all* dimensions of variability (scene, lighting, and position), highlighting that these factors are not redundant but mutually reinforcing.

B. Sensitivity Analysis: Does Factor Quality Matter?

We move from a binary "with/without" comparison to a quantitative question: does better factor quality lead to higher success? We vary each factor's strength in controlled ablations and measure continuous proxies aligned with the intended semantics, as summarized in Figure 5.

Object Grounding. We measure the fraction of attention mass falling inside the object/gripper mask, using the same supervision target as Eq. 2. As the mask-aligned attention ratio increases from 0.25 to 1.0, success rises from 61.3% to 74.6% (Figure 5 (a)). In contrast, π_0 exhibits low intrinsic object focus (26.5%), indicating that stronger spatial grounding directly correlates with performance.

Skill Recognition. We use a linear probe to predict task skill labels from action features, making probe accuracy as the

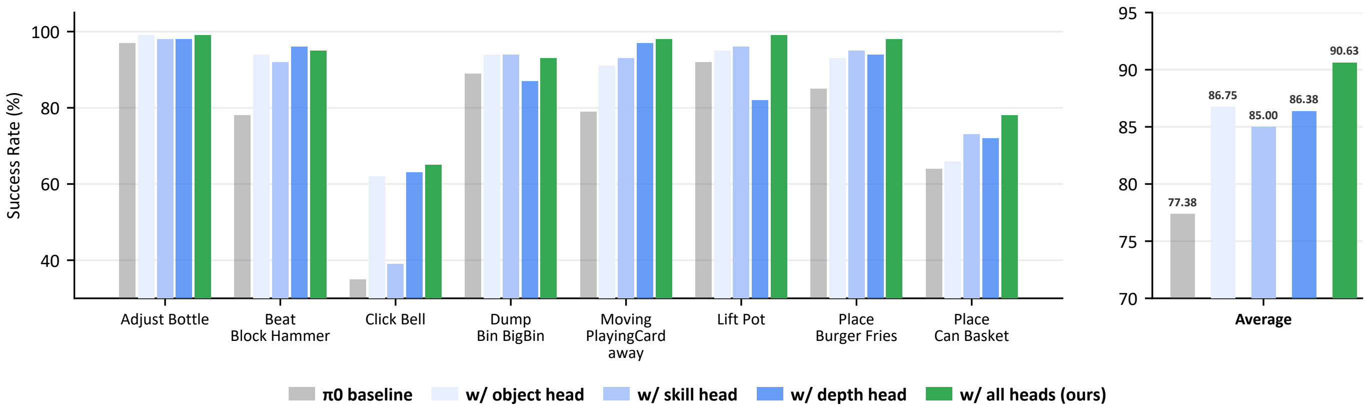


Fig. 3: **Robotwin 2.0 Benchmark Performance.** Success rates across 8 manipulation tasks comparing the π_0 baseline, single-head experts, and our full model. While specific heads excel at aligned tasks (e.g., depth head for geometry-heavy Beat Block Hammer), the full model (purple) integrates these capabilities to achieve the best overall average performance (90.63%).

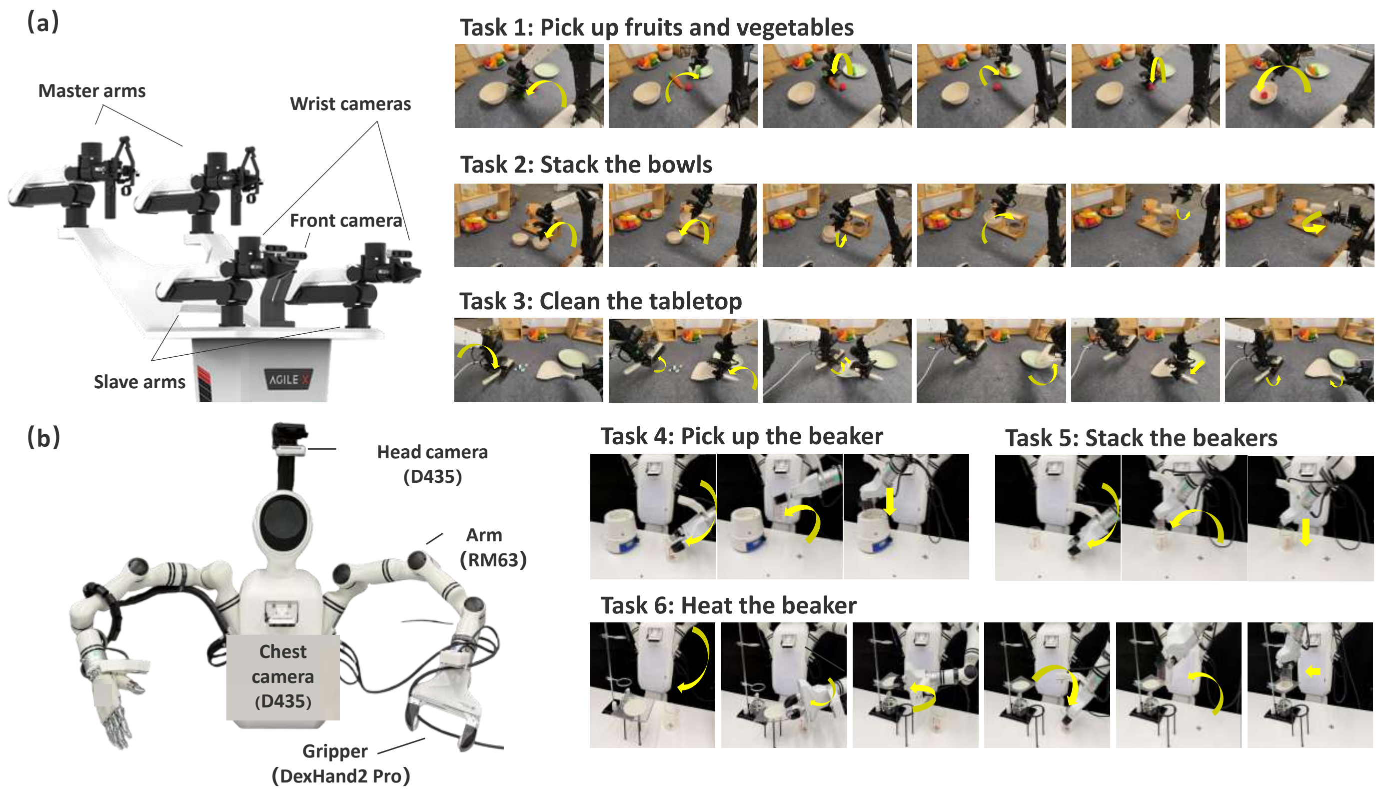


Fig. 4: **Real-world Robot Platforms and Evaluation Tasks.** (a) ALOHA AgileX dual-arm mobile manipulator with left/right wrist RealSense D435 cameras and a third-person RealSense D435i camera; we evaluate three household tasks: pick up fruits and vegetables, stack the bowls, clean the tabletop. (b) PSI-Bot equipped with RealMan RM63 arm(s) and DexHand2 Pro hands, with head/chest RealSense D435 cameras; we evaluate three lab tasks: pick up beaker, stack beakers, and heat beaker.

quality metric. Raising the skill accuracy threshold from 0.25 to 1.0 increases success from 66.2% to 72.9% (Figure 5 (b)). The baseline π_0 remains low at 48.4%, confirming improved temporal representations translate into better performance.

Geometry Perception. We modulate the true depth feature ratio in the geometry stream as a proxy for geometric signal strength. Increasing this ratio from 0 to 1.0 yields a large success gain (15.6% $\rightarrow$ 76.7%, Figure 5 (c)), demonstrating that richer geometric cues substantially improve task outcomes.

Figure 5 summarizes these trends, showing that success increases monotonically with each factor's quality, not merely its presence. For a qualitative analysis of our factor feature with respect to time in a task, as in Figure 6. For details of these metrics and ablations, please refer to Appendix.

C. *Specialization Enables Decoupled Feature Learning*

We have shown that each factor (object grounding, geometry, and skill) correlates with task success. A natural next

TABLE II: **Cross-Platform Real-World Generalization.** Success rates ($N = 20$) across four generalization scenarios on ALOHA and PSI-Bot platforms. Our method consistently outperforms baseline, achieving performance gains across all scenarios (up to 52.7%) and demonstrating robustness under challenging out-of-domain conditions. Task 1–6 correspond to: (1) pick up fruits and vegetables (2) stack the bowls (3) clean the tabletop (4) pick up the beaker (5) stack the beakers and (6) heat the beaker. In-domain generalization includes variations in object positions within training distribution.

Generalization Setting	**Method**	**ALOHA AgileX** Task 1	Task 2	Task 3	**PSI-Bot RealMan** Task 4	Task 5	Task 6	**Average** (%)
In-Domain†	Base Policy	10/20	11/20	9/20	12/20	12/20	13/20	55.8
	Ours	**14/20**	**15/20**	**14/20**	**16/20**	**17/20**	**15/20**	**75.8**
Scene	Base Policy	7/20	8/20	6/20	12/20	11/20	9/20	44.2
	Ours	**13/20**	**12/20**	**11/20**	**15/20**	**16/20**	**14/20**	**67.5**
Lighting	Base Policy	11/20	9/20	10/20	14/20	12/20	13/20	57.5
	Ours	**13/20**	**16/20**	**15/20**	**17/20**	**18/20**	**16/20**	**79.2**

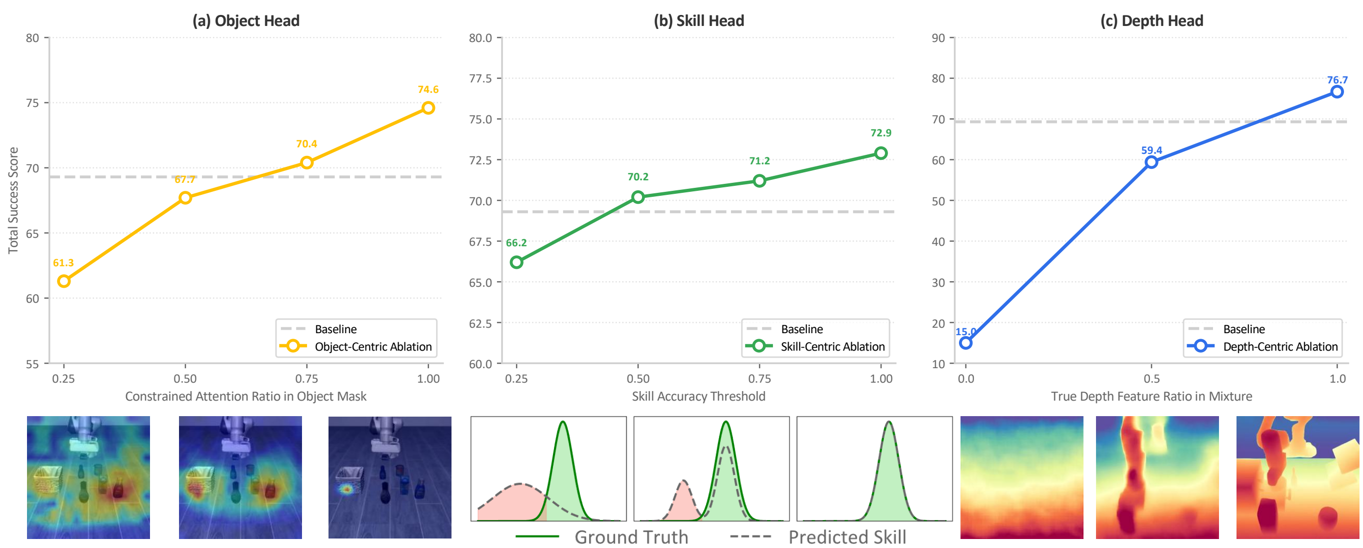


Fig. 5: **Higher Factor Quality Leads to Better Task Performance. Top:** Quantitative analysis on the LIBERO-Plus layout perturbation track shows that improving the quality of each specialized head consistently boosts success rates. **(a) Object Head:** as the proportion of attention focused on task-relevant object regions increases, success rises from 61.3% to 74.6%, highlighting the importance of precise object-centric attention. **(b) Skill Head:** higher skill-recognition accuracy, measured by a linear probe, correlates with improved performance (66.2% to 72.9%), indicating that better temporal understanding enhances control. **(c) Depth Head:** increasing the ratio of true depth features (versus noise) dramatically improves both qualitative depth estimation and quantitative success (15.6% to 76.7%), confirming that explicit 3D cues are critical for robust manipulation. **Bottom:** Qualitative visualizations show how changes along the x-axis metrics are reflected in the corresponding feature representations.

question is: *can we guide multiple factors by mixedly training all the heads with all the factors?* Our answer is **NO**: a naive mixed training protocol consistently underperforms (Figure 8). When object grounding, geometry, and skill objectives are all supervised through all attention heads, their features become entangled, as in Fig. 9. This coupling means that information from different factors is mixed, making it difficult to capture each factor clearly, thus leading to degenerated performance.

D. Comparison to Other Factor Guidance Approaches

There exists two representative paradigms in providing task-relevant factors to VLA: DreamVLA [92] lets VLM predict dynamic regions, depth map, and semantic knowledge with extra query. AdaMoE [74] applies a Mixture of Experts to automatically learn experts for different tasks.

As shown in Table I, our method outperforms both DreamVLA (69.9%) and AdaMoE (50.1%) across all perturbation dimensions and task suites, achieving the highest overall average success rate (75.4%), especially in challenging settings such as camera, robot, and layout perturbations. We attribute these gains to our explicit attention head specialization, which enables the model to disentangle and robustly capture object grounding, skill recognition, and geometric cues.

E. Ablation of Design Choices

In this section, we discuss the design choices for each specified head and the plug-and-play residual adapter - ControlNet. We systematically conduct experiments regarding these points

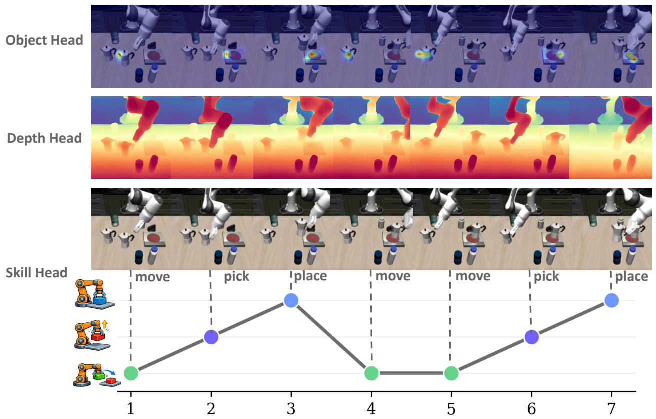


Fig. 6: **Visualization of Learned Representations in GuidedVLA.** From top to bottom: **(i)** Object attention focuses on the manipulation target (e.g., pot handle); **(ii)** Depth features encode explicit 3D structure; **(iii)** Skill predictions track the temporal progress of task phases. This confirms that each head specializes in its designated semantic factor as intended.

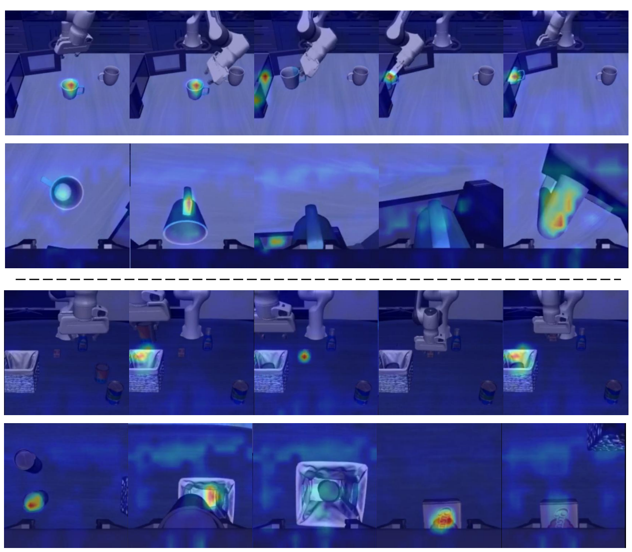


Fig. 7: **Visualization of Object Heads Attention for Chest and Wrist Cameras.** **Cunxin: Please update this figure before arxiv.**

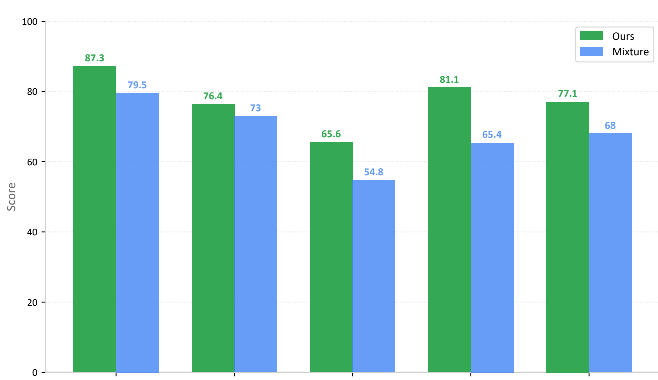
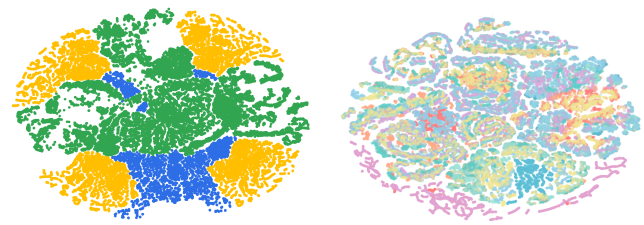


Fig. 8: **Comparison of GuidedVLA against Mixture Alternative.** Attention head specialization explicitly outperforms learning all objectives in a mixture.



(a) Ours (b) Mixture

Fig. 9: **t-SNE Visualization of Attention Outputs.** (a) Specialized attention heads (object: yellow, depth: blue, skill: green) form well-separated clusters, demonstrating factor disentanglement and minimal interference. (b) The mixture alternative shows overlapping clusters (different colors representing different heads), indicating entangled representations.

in RoboTwin2.0, with detailed results in Appendix.

Object Head. Suppressing attention of non-interested area outperforms enforcing a fixed spatial prior (e.g., Gaussian) for interested areas. This allows the model to flexibly learn which object parts are most relevant at each task stage.

Skill Head. Soft targets outperforms one-hot labels, which better handles ambiguous or mixed-intent segments and thus leads to more stable training and better performance.

Depth Head. Adopting an extra lightweight, downsampling adapter outperforms directly using original, non-downsampled depth features, since the number of depth tokens could be huge which makes learning process difficult.

Extra Branch. With zero-initialized control branch, auxiliary signals are introduced gradually and do not disrupt the base model at the start of training, outperforming not using.

VI. Conclusion

We present **GuidedVLA**, a VLA paradigm, where the action decoder is explicitly guided to capture task-relevant information such as object grounding, spatial geometry, and temporal skill logic. Across simulation and real-robot experiments, GuidedVLA significantly improves success rates in both in-domain and out-of-domain settings, demonstrates effectiveness of specifying attention heads of the action decoder with explicit guidance. Our work paves the way to train more interpretable and generalizable VLA.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[3] 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.

[4] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[5] Vineet Bhat, Yu-Hsiang Lan, Prashanth Krishnamurthy, Ramesh Karri, and Farshad Khorrami. 3d cavla: Leveraging depth and 3d context to generalize vision language action models for unseen tasks. *arXiv preprint arXiv:2505.05800*, 2025.

[6] 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.

[7] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $0.5\pi_{0.5} : Avision - language - actionmodelwithopen - worldgeneralization.In$*9th Annual Conference on Robot Learning*, 2025.

[8] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $0\pi_0 : Avision - language - actionflowmodelforgeneralrobotcontrol.InRSS$, 2025.

[9] Kevin Black, Manuel Y Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *arXiv preprint arXiv:2506.07339*, 2025.

[10] 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.

[11] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

[12] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[13] Alexandre Chapin, Emmanuel Dellandréa, and Liming Chen. Storm: Slot-based task-aware object-centric representation for robotic manipulation. *arXiv preprint arXiv:2601.20381*, 2026.

[14] Jiayi Chen, Wenxuan Song, Pengxiang Ding, Ziyang Zhou, Han Zhao, Feilong Tang, Donglin Wang, and Haoang Li. Unified diffusion vla: Vision-language-action model via joint discrete denoising diffusion process. *arXiv preprint arXiv:2511.01718*, 2025.

[15] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088.

[16] Baiye Cheng, Tianhai Liang, Suning Huang, Maanping Shao, Feihong Zhang, Botian Xu, Zhengrong Xue, and Huazhe Xu. Moe-dp: An moe-enhanced diffusion policy for robust long-horizon robotic manipulation with skill decomposition and failure recovery. *arXiv preprint arXiv:2511.05007*, 2025.

[17] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[18] Yinpei Dai, Jayjun Lee, Yichi Zhang, Ziqiao Ma, Jed Yang, Amir Zadeh, Chuan Li, Nima Fazeli, and Joyce Chai. Aimbot: A simple auxiliary visual cue to enhance spatial awareness of visuomotor policies. *arXiv preprint arXiv:2508.08113*, 2025.

[19] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[20] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in neural information processing systems*, 32, 2019.

[21] Shengliang Deng, Mi Yan, Yixin Zheng, Jiayi Su, Wenhao Zhang, Xiaoguang Zhao, Heming Cui, Zhizheng Zhang, and He Wang. Stereovla: Enhancing vision-language-action models with stereo vision. *arXiv preprint arXiv:2512.21970*, 2025.

[22] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-

e: An embodied multimodal language model. In *International Conference on Machine Learning*, pages 8469–8488. PMLR, 2023.

[23] Frederik Ebert et al. Bridgedata v2: A dataset for robot learning at scale. *arXiv preprint arXiv:2308.12952*, 2023.

[24] Cunxin Fan, Xiaosong Jia, Yihang Sun, Yixiao Wang, Jianglan Wei, Ziyang Gong, Xiangyu Zhao, Masayoshi Tomizuka, Xue Yang, Junchi Yan, et al. Interleave-vla: Enhancing robot manipulation with interleaved image-text instructions. In *ICLR*, 2026.

[25] Qingyu Fan, Zhaoxiang Li, Yi Lu, Wang Chen, Qiu Shen, Xiao-xiao Long, Yinghao Cai, Tao Lu, Shuo Wang, and Xun Cao. Peafowl: Perception-enhanced multi-view vision-language-action for bimanual manipulation. *arXiv preprint arXiv:2601.17885*, 2026.

[26] Hung-Chieh Fang, Kuo-Han Hung, Chu-Rong Chen, Po-Jung Chou, Chun-Kai Yang, Po-Chen Ko, Yu-Chiang Wang, Yueh-Hua Wu, Min-Hung Chen, and Shao-Hua Sun. Learning skills from action-free videos. *arXiv preprint arXiv:2512.20052*, 2025.

[27] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

[28] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International conference on learning representations*, 2018.

[29] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2 (11):665–673, 2020.

[30] Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024.

[31] Siddhant Haldar and Lerrel Pinto. Point policy: Unifying observations and actions with key points for robot manipulation. *arXiv preprint arXiv:2502.20391*, 2025.

[32] Cheng-Chun Hsu, Bowen Wen, Jie Xu, Yashraj Narang, Xiaolong Wang, Yuke Zhu, Joydeep Biswas, and Stan Birchfield. Spot: Se(3) pose trajectory diffusion for object-centric manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4853–4860, 2025. doi: 10.1109/ICRA55743.2025.11127562. arXiv:2411.00965.

[33] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[34] Aoshen Huang, Jiaming Chen, Jiyu Cheng, Ran Song, Wei Pan, and Wei Zhang. Skill-aware diffusion for generalizable robotic manipulation. *arXiv preprint arXiv:2601.11266*, 2026.

[35] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Conference on Robot Learning*, pages 4573–4602. PMLR, 2025.

[36] Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. Pointworld: Scaling 3d world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782*, 2026.

[37] Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U Tan, Navonil Majumder, Soujanya Poria, et al. Nora: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*, 2025.

[38] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[39] Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and g0 dual-system vla model. *arXiv preprint arXiv:2509.00576*, 2025.

[40] Yuhua Jiang, Shuang Cheng, Yan Ding, Feifei Gao, and Biqing Qi. Asyncvla: Asynchronous flow matching for vision-language-action models. *arXiv preprint arXiv:2511.14148*, 2025.

[41] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2(3):6, 2022.

[42] 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. *arXiv preprint arXiv:2403.12945*, 2024.

[43] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[44] 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*, pages 2679–2713. PMLR, 2025.

[45] Seungjae Lee, Yoonkyo Jung, Inkook Chun, Yao-Chih Lee, Zikui Cai, Hongjia Huang, Aayush Talreja, Tan Dat Dao, Yongyuan Liang, Jia-Bin Huang, et al. Tracegen: World modeling in 3d trace space enables learning from cross-embodiment videos. *arXiv preprint*

arXiv:2511.21690, 2025.

[46] Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. *arXiv preprint arXiv:2510.12276*, 2025.

[47] Guangrun Li, Yaoxu Lyu, Zhuoyang Liu, Chengkai Hou, Jieyu Zhang, and Shanghang Zhang. H2r: A human-to-robot data augmentation for robot pre-training from videos. *arXiv preprint arXiv:2505.11920*, 2025.

[48] Hang Li, Qian Feng, Zhi Zheng, Jianxiang Feng, Zhaopeng Chen, and Alois Knoll. Language-guided object-centric diffusion policy for generalizable and collision-aware manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12834–12841. IEEE, 2025.

[49] Jinming Li, Yichen Zhu, Zhibin Tang, Junjie Wen, Minjie Zhu, Xiaoyu Liu, Chengmeng Li, Ran Cheng, Yaxin Peng, Yan Peng, et al. Coa-vla: Improving vision-language-action models via visual-text chain-of-affordance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9759–9769, 2025.

[50] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[51] Peiyan Li, Yixiang Chen, Hongtao Wu, Xiao Ma, Xiangnan Wu, Yan Huang, Liang Wang, Tao Kong, and Tieniu Tan. Bridgevla: Input-output alignment for efficient 3d manipulation learning with vision-language models. *arXiv preprint arXiv:2506.07961*, 2025.

[52] Ziwen Li, Xin Wang, Hanlue Zhang, Runnan Chen, Runqi Lin, Xiao He, Han Huang, Yandong Guo, Fakhri Karray, Tongliang Liu, et al. Posa-vla: Enhancing action generation via pose-conditioned anchor attention. *arXiv preprint arXiv:2512.03724*, 2025.

[53] Yixing Liang, Anna Xie, Ziyun Feng, Yuke Zhu, Song-Chun Zhu, and Yunzhu Li. Skilldiffuser: Interpretable skill planning for latent diffusion-based manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16467–16476, 2024.

[54] Zhixuan Liang, Yizhuo Li, Tianshuo Yang, Chengyue Wu, Sitong Mao, Tian Nian, Liuao Pei, Shunbo Zhou, Xiaokang Yang, Jiangmiao Pang, et al. Discrete diffusion vla: Bringing discrete diffusion to action decoding in vision-language-action policies. *arXiv preprint arXiv:2508.20072*, 2025.

[55] Fanqi Lin, Haojie Lu, Haojian Fang, and Ping Luo. Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation. *arXiv preprint arXiv:2406.01586*, 2024.

[56] Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

[57] Kevin Lin, Varun Ragunath, Andrew McAlinden, Aaditya Prasad, Jimmy Wu, Yuke Zhu, and Jeannette Bohg. Constraint-preserving data generation for one-shot visuomotor policy generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 3631–3646. PMLR, 2025. URL https://proceedings.mlr.press/v305/lin25b.html.

[58] 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.

[59] 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. In *The Thirteenth International Conference on Learning Representations*, 2025.

[60] Jing Ma, Zhengyi Jiang, Rifat Hoque, Sangwoo Ahn, Pulkit Agrawal, and Kaiming Lee. Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18081–18090, 2024.

[61] Yunchao Ma, Yizhuang Zhou, Yunhuan Yang, Tiancai Wang, and Haoqiang Fan. Running vlas at real-time speed. *arXiv preprint arXiv:2510.26742*, 2025.

[62] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[63] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

[64] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[65] Zehao Ni, Yonghao He, Lingfeng Qian, Jilei Mao, Fa Fu, Wei Sui, Hu Su, Junran Peng, Zhipeng Wang, and Bin He. Vo-dp: Semantic-geometric adaptive diffusion policy for vision-only robotic manipulation. *arXiv preprint arXiv:2510.15530*, 2025.

[66] 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)*, pages 6892–6903. IEEE, 2024.

[67] Mingjie Pan, Jiyao Zhang, Tianshu Wu, Yinghao Zhao, Wenlong Gao, and Hao Dong. Omnimanip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17359–17369, 2025.

[68] 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.

[69] Jingjing Qian, Boyao Han, Chen Shi, Lei Xiao, Long Yang, Shaoshuai Shi, and Li Jiang. Geopredict: Leveraging predictive kinematics and 3d gaussian geometry for precise vla manipulation. *arXiv preprint arXiv:2512.16811*, 2025.

[70] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. In *Robotics: Science and Systems*, 2025.

[71] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ha6RTeWMd0.

[72] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024.

[73] Ralf Römer, Yi Zhang, and Angela P Schoellig. Clare: Continual learning for vision-language-action models via autonomous adapter routing and expansion. *arXiv preprint arXiv:2601.09512*, 2026.

[74] Weijie Shen, Yitian Liu, Yuhao Wu, Zhixuan Liang, Sijia Gu, Dehui Wang, Tian Nian, Lei Xu, Yusen Qin, Jiangmiao Pang, et al. Expertise need not monopolize: Action-specialized mixture of experts for vision-language-action learning. *arXiv preprint arXiv:2510.14300*, 2025.

[75] Lin Sun, Bin Xie, Yingfei Liu, Hao Shi, Tiancai Wang, and Jiale Cao. Geovla: Empowering 3d representations in vision-language-action models. *arXiv preprint arXiv:2508.09071*, 2025.

[76] Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*, 2025.

[77] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[78] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[80] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[81] Jason Z Wang, Kuan Fang, Tonghe Zhang, and Yunzhu Li. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[82] 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.

[83] Junjie Wen, Minjie Zhu, Jiaming Liu, Zhiyuan Liu, Yicun Yang, Linfeng Zhang, Shanghang Zhang, Yichen Zhu, and Yi Xu. dvla: Diffusion vision-language-action model with multimodal chain-of-thought. *arXiv preprint arXiv:2509.25681*, 2025.

[84] Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

[85] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

[86] Shijie Wu, Yihang Zhu, Yunao Huang, Kaizhen Zhu, Jiayuan Gu, Jingyi Yu, Ye Shi, and Jingya Wang. Afforddp: Generalizable diffusion policy with transferable affordance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6971–6980, June 2025.

[87] Hang Yu, Juntu Zhao, Yufeng Liu, Kaiyu Li, Cheng Ma, Di Zhang, Yingdong Hu, Guang Chen, Junyuan Xie, Junliang Guo, et al. Point what you mean: Visually grounded instruction policy. *arXiv preprint arXiv:2512.18933*, 2025.

[88] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[89] Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Zhuoguang

Chen, Tao Jiang, and Hang Zhao. Depthvla: Enhancing vision-language-action models with depth-aware spatial reasoning. *arXiv preprint arXiv:2510.13375*, 2025.

[90] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Robotics: Science and Systems (RSS)*, 2024. doi: 10.15607/RSS.2024.XX.067.

[91] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847, 2023.

[92] Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, Fan Lu, He Wang, et al. Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. In *NeurIPS*, 2025.

[93] Ruihan Zhao, Tyler Ingebrand, Sandeep Chinchali, and Ufuk Topcu. Mos-vla: A vision-language-action model with one-shot skill adaptation. *arXiv preprint arXiv:2510.16617*, 2025.

[94] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[95] Yongsheng Zhao, Lei Zhao, Baoping Cheng, Gongxin Yao, Xuanzhang Wen, and Han Gao. Vla-rail: A real-time asynchronous inference linker for vla models and robots. *arXiv preprint arXiv:2512.24673*, 2025.

[96] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: A 3d vision-language-action generative world model. In *International Conference on Machine Learning*, pages 61229–61245. PMLR, 2024.

[97] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

[98] Hongyan Zhi, Peihao Chen, Siyuan Zhou, Yubo Dong, Quanxi Wu, Lei Han, and Mingkui Tan. 3dflowaction: Learning cross-embodiment manipulation from 3d flow world model. *arXiv preprint arXiv:2506.06199*, 2025.

[99] 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*, pages 2165–2183. PMLR, 2023.

Appendix

A. Implementation Details of Each Head and Adapter

Object Head. We implement object-level attention supervision by aligning predicted attention weights with stage-specific object masks. Algorithm 2 outlines the procedure for computing the object grounding loss. It selects a subset of attention heads, indexes image-view patches, and compares attention activations with ground-truth spatial distributions. The supervision is applied across multiple transformer layers and averaged to produce the final loss.

Depth Head. To integrate depth information into cross-modal attention, we use a specialized Key-Value (KV) projector that maps depth tokens into compatible representations for selected attention heads. Algorithm 3 provides the implementation of the `DepthKVProjector` and how it is used to modify attention computation. Standard heads use key and value states from VLM backbone, while selected heads attend to projected depth tokens, supporting geometry-aware reasoning.

Skill Head. The Skill Head encourages semantic grounding by matching attention-derived features to a soft skill distribution target. Algorithm 4 describes the KL loss computation pipeline. For each transformer layer, we extract the action-query attention output, average features, and apply a classification head. The output is compared to smoothed histogram targets, capturing the distribution of skill labels across the trajectory.

ControlNet-style Adapter. To enable fine-grained control signal injection, we design a ControlNet-inspired dual-path attention mechanism. Algorithm 5 shows the implementation of ‘ControlAttention‘, which splits the attention computation into a main path and a control-specific branch. The outputs from both branches are fused using a zero-initialized linear projection, allowing conditional modulation without disrupting pretrained behavior.

B. Implementation Details of Factor Quality Ablation

Object Head. To assess the impact of factor quality, we require precise control over the model's grounding strength. While the standard supervision in Eq. 2 encourages the model to maximize attention on the object, this ablation study necessitates clamping the intensity to specific scalar values $\alpha \in \{0.25, 0.5, 0.75, 1.0\}$.We define the grounding strength m as the cumulative attention mass falling within the ground-truth region mask $\mathcal{M}$:

$$m = \sum_{j \in \mathcal{M}} A_j, \tag{7}$$

where A_j are the attention weights of the action token. Since the total attention is normalized, m directly represents the concentration of focus on the target object. For this experiment, we replace the standard objective with a regression loss to force m towards the target α:

$$\mathcal{L}_{\text{ablation}} = \begin{cases} \dfrac{0.5\,(m-\alpha)^2}{\beta}, & \text{if } |m-\alpha| < \beta, \\ |m-\alpha| - 0.5\,\beta, & \text{otherwise.} \end{cases} \tag{8}$$

where the smoothing parameter β is set to 0.05. This objective allows us to strictly regulate the grounding quality for sensitivity analysis.

For the baseline model π_0 (which lacks explicit object supervision), we measure its intrinsic grounding capability using the same metric m. The reported value (26.5%) is obtained by averaging m over 200 evaluation steps during inference, reflecting the model's natural tendency to attend to relevant objects without auxiliary guidance.

Depth Head. Unlike the Object and Skill heads, the Depth head is enforced via architectural constraints (attention injection) rather than optimization objectives. Therefore, we cannot regulate its quality through loss scaling. Instead, we control the strength of geometric cues by modulating the signal-to-noise ratio of the input features.Let $\mathbf{f}_{\text{depth}}$ denote the feature extracted from the frozen depth encoder, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ be a Gaussian noise vector normalized to match the statistics of the depth features. We introduce a control parameter $\delta \in [0, 1]$ (referred to as the "depth feature ratio" in the analysis) to construct the ablated feature representation $\tilde{\mathbf{f}}$:

$$\tilde{\mathbf{f}} = \delta \cdot \mathbf{f}\text{depth} + (1-\delta) \cdot \boldsymbol{\epsilon}. \tag{9}$$

These corrupted features $\tilde{\mathbf{f}}$ are then projected into keys KDepth and values V_{Depth} for the specific attention heads. By varying δ from 0 (pure noise) to 1.0 (clean depth signal), we quantitatively evaluate how the quality of 3D structural information impacts manipulation success.

Skill Head. To examine the causal effect of skill recognition on task success, we regulate the model's intent classification accuracy to specific target levels $\gamma \in \{0.25, 0.5, 0.75, 1.0\}$.We define the *soft accuracy* S as the mean predicted probability assigned to the ground-truth skill class y_i across a batch of size N:

$$S = \frac{1}{N} \sum_{i=1}^{N} \hat{p}i(y_i), \tag{10}$$

where $\hat{p}_i(y_i)$ denotes the probability of the correct label derived from the softmax distribution. To enforce convergence to the target accuracy γ, we introduce an auxiliary control loss $\mathcal{L}$ctrl derived from the Smooth L1 distance:

$$\mathcal{L}_{\text{ctrl}} = \begin{cases} \dfrac{0.5\,(S-\gamma)^2}{\beta}, & \text{if } |S-\gamma| < \beta, \\ |S-\gamma| - 0.5\,\beta, & \text{otherwise.} \end{cases} \tag{11}$$

with β set to 0.02. The final objective for the skill head during ablation is a weighted sum: $\mathcal{L}\text{total} = \mathcal{L}\text{skill} + \lambda\mathcal{L}\text{ctrl}$. By adjusting the weight λ, we empirically ensure the model's skill recognition performance converges to the designated target

To measure the intrinsic skill representation capability of the baseline π_0 (reported as 48.4%), we employ the identical skill head architecture and classification objective ($\mathcal{L}_{\text{skill}}$) as described above. Specifically, we attach the projection layer ($\mathbf{f}_h \rightarrow \hat{\mathbf{p}}_h$) to the pre-trained π_0. Distinct from the controlled ablation models, we freeze the entire backbone and exclusively optimize the projection parameters ($\boldsymbol{W}, \mathbf{b}$) using the classification loss. This setup effectively functions as a linear probe

TABLE III: **Head-wise real-robot results on factor-aligned tasks.** We report success rates for the base π_0 policy, three single-head diagnostic variants (Object-only / Depth-only / Skill-only), and the full GuidedVLA under three distribution shifts: positional (in-domain), scene, and lighting generalization. To isolate factor contributions, each single-head variant is evaluated *only* on its aligned tasks: Object→T1/T4, Depth→T2/T5, Skill→T3/T6; other entries are not evaluated and shown as "–".

Setting	Method	ALOHA AgileX			PSI-Bot			Avg.
		T1 (Object)	**T2 (Depth)**	**T3 (Skill)**	**T4 (Object)**	**T5 (Depth)**	**T6 (Skill)**	
In-Domain	π_0 (Base)	10/20	11/20	9/20	12/20	12/20	13/20	55.8
	Object-only	11/20	–	–	14/20	–	–	62.5
	Depth-only	–	13/20	–	–	14/20	–	67.5
	Skill-only	–	–	12/20	–	–	13/20	62.5
	GuidedVLA (Full)	14/20	15/20	14/20	16/20	17/20	15/20	75.8
Scene	π_0 (Base)	7/20	8/20	6/20	12/20	11/20	9/20	44.2
	Object-only	10/20	–	–	12/20	–	–	55
	Depth-only	–	11/20	–	–	13/20	–	60
	Skill-only	–	–	9/20	–	–	12/20	52.5
	GuidedVLA (Full)	13/20	12/20	11/20	15/20	16/20	14/20	67.5
Lighting	π_0 (Base)	11/20	9/20	10/20	14/20	12/20	13/20	57.5
	Object-only	11/20	–	–	15/20	–	–	65
	Depth-only	–	13/20	–	–	15/20	–	70
	Skill-only	–	–	12/20	–	–	14/20	65
	GuidedVLA (Full)	13/20	16/20	15/20	17/20	18/20	16/20	79.2

on the fixed action features to evaluate their linear separability regarding skill semantics.

Considering the Libero dataset consists of 3 distinct skill categories, a purely random guess would yield an accuracy of $\sim$ 33.3%. Consequently, the baseline's performance of 48.4% represents only a marginal improvement over chance. This indicates that without explicit temporal logic supervision, the representation of π_0 captures negligible high-level intent information, failing to effectively disentangle the long-horizon structure of tasks.

Cunxin: what does the result of π_0 (reported as 48.4%) mean? Generate a paragraph here. For example, you may claim pi0's feature contains no skill understanding, since 48% is only a little higher than random guessing.

C. Ablation on Head and Adapter Design Choices

1) Object Head: To improve the interpretability and spatial alignment of attention in action decoding, we supervise a dedicated set of heads $\mathcal{H}_{\text{obj}}$ to focus on semantically meaningful regions—such as the object to be grasped or its intended destination. We investigate two strategies for supervising these attention heads: a binary mask–based region loss, and a Gaussian prior–based KL divergence loss.

Object Region Supervision. To guide a subset of heads $\mathcal{H}_{\text{obj}}$ toward attending semantically meaningful areas such as the grasp object or destination region, we use direct supervision from binary masks $\boldsymbol{A}^{\text{GT}}$ annotated by foundation models. These masks indicate which patches are considered object-relevant. The supervision loss is defined as a masked negative log-likelihood over attention weights $\boldsymbol{A}$:

$$\mathcal{L}_{\text{object}} = -\frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} \boldsymbol{A}_a \log \boldsymbol{A}^{\text{GT}}. \quad (12)$$

Importantly, $\boldsymbol{A}^{\text{GT}}$ only sets non-object regions to 0 while leaving object regions unconstrained, allowing the model to flexibly distribute attention within those areas. This formulation does not penalize the model for exactly where it attends inside the object, but it does penalize any attention leaked outside the object region. As a result, the model learns to concentrate its attention inside the annotated object boundaries, without requiring precise spatial alignment. Empirically, this encourages more consistent and interpretable object-level grounding in attention maps.

Gaussian Prior Supervision. As an alternative, we evaluate a softer supervision strategy by replacing the binary mask with a 2D Gaussian prior centered at the mass centroid of the annotated object region. This provides a spatial bias encouraging attention to concentrate near the most representative region of the object. Specifically, we generate a normalized Gaussian heatmap $\boldsymbol{G} \in \mathbb{R}^{3\times 256}$ and compute the KL divergence between student attention and this distribution:

$$\mathcal{L}_{\text{KL}} = \frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}}\sum_{v=1}^{3}\sum_{p=1}^{256} \boldsymbol{G}_{v,p}\left(\log \boldsymbol{G}_{v,p} - \log \boldsymbol{A}_{a,v,p}\right). \quad (13)$$

Experiments and Results. We compare these two supervision strategies on the *RoboTwin 2.0* benchmark. As shown in Table IV, using object region supervision significantly outperforms the Gaussian prior approach, achieving an average success rate of 83.33% compared to 72.00%. This demonstrates that direct region supervision provides a stronger and more effective learning signal for grounding attention in semantically meaningful areas, leading to better task performance.

2) Depth Head: To better incorporate geometric information for 3D reasoning in manipulation tasks, we introduce a dedicated *Depth Head* that leverages depth embeddings from

Algorithm 2 Python Pseudocode of Applying Object Grounding Loss on Object Head

```
def object_attn_guidance(
    all_attn_states,
    teacher_attn_maps,          # {"object_maps": ..., "object_masks": ...}
    object_head_indices,
    att_mask,
    view_patch_indices,
    action_query_start_idx,
):
    layer_losses = []
    for layer_idx, attn_data in all_attn_states:
        Q, K = get_qk(attn_data)
        P = attention_probs(Q, K, att_mask)                 # [B, H, S, S]
        P = P[:, object_head_indices, action_query_start_idx:, :]

        P_img = index_select(P, dim=-1, index=view_patch_indices)
        P_img = reshape(P_img, [B, H, A_len, 3, 256])

        L = object_grounding_loss(
            P_img,
            teacher_attn_maps["object_maps"],
            teacher_attn_maps["object_masks"],
        )
        layer_losses.append(L)

    return mean(layer_losses) if len(layer_losses) > 0 else 0.0

def masked_mean(loss, valid, eps=1e-9):
    mask = valid[:, None].float().expand_as(loss) # [B, A_len]
    denom = mask.sum() + eps
    return (loss * mask).sum() / denom

def object_grounding_loss(P_img, object_maps, object_masks, eps=1e-6, delta=1e-6):
    S = mean(P_img, dim=1)                                  # [B, A_len, 3, 256]
    M_obj = (object_maps > eps).float()                     # [B, 3, 256]
    M_view = object_masks.float()                           # [B, 3]
    M = M_obj * broadcast(M_view)                           # [B, 3, 256]

    mass = sum_over_view_patch(S * broadcast(M))            # [B, A_len]
    loss = -log(clamp_min(mass, delta))                     # [B, A_len]

    valid = (sum_over_view_patch(M) > 0).float()            # [B]
    return masked_mean(loss, valid)
```

TABLE IV: **Ablation Study of Object Head Design on RoboTwin 2.0 tasks.** We compare two strategies for supervising attention heads in the Object Head: region-based supervision using binary masks from foundation models, and Gaussian prior–based KL divergence. The region-based method leads to significantly higher performance (83.33% average success rate), especially on precision-critical tasks such as *Click Bell*, confirming the advantage of providing explicit spatial constraints over soft priors when grounding object-level attention.

Model	Beat Hammer Block	Click Bell	Dump Bin BigBin	Avg
π_0 w/ gaussian	89%	40%	87%	72.00%
π_0 w/ object region **(Ours)**	**94%**	**62%**	**94%**	**83.33%**

Algorithm 3 Python Pseudocode of Depth KV Projector and Depth Head Attention

```python
class DepthKVProjector:
    def __init__(self, kv_projector):
        self.kv_projector = kv_projector

    @property
    def heads_to_modify(self):
        return self.kv_projector.heads_to_modify

    def project_group(self, depth_tokens, g, B, T_depth, H, D):
        # depth_tokens: [B, T_depth, hidden]
        depth_tokens = rmsnorm(depth_tokens)  # [B, T_depth, hidden]

        k = self.kv_projector.k_linear[g](depth_tokens)  # [B, T_depth, H*D]
        v = self.kv_projector.v_linear[g](depth_tokens)  # [B, T_depth, H*D]

        k = k.view(B, T_depth, H, D).transpose(1, 2)     # [B, H, T_depth, D]
        v = v.view(B, T_depth, H, D).transpose(1, 2)     # [B, H, T_depth, D]

        return {
            "depth_token_k": k,
            "depth_token_v": v,
            "heads_to_modify": self.heads_to_modify,
        }

    def build(self, depth_tokens_tuple, B, T_depth, H, D):
        # depth_tokens_tuple: tuple of length G, each [B, T_depth, hidden]
        return [
            self.project_group(depth_tokens, g, B, T_depth, H, D)
            for g, depth_tokens in enumerate(depth_tokens_tuple)
        ]

    def get_cfg(self, depth_tokens_tuple, depth_group_idx, B, T_depth, H, D):
        return self.build(depth_tokens_tuple, B, T_depth, H, D)[depth_group_idx]


def depth_modified_attention(
    Q, K, V, att_mask, scaling, dropout_p,
    depth_tokens_tuple=None,
    depth_kv_projector: DepthKVProjector = None,
    depth_cfg=None,
    depth_group_idx=0,
    B=None, T_depth=None, H=None, D=None,
):
    depth_cfg = depth_kv_projector.get_cfg(
        depth_tokens_tuple,
        depth_group_idx=depth_group_idx,
        B=B, T_depth=T_depth, H=H, D=D
    )

    heads_to_modify = depth_cfg["heads_to_modify"]
    Kd = depth_cfg["depth_token_k"]                             # [B, H, T_depth, D]
    Vd = depth_cfg["depth_token_v"]                             # [B, H, T_depth, D]

    std_heads = all_heads_except(H, heads_to_modify)
    mod_heads = heads_to_modify
    out = zeros_like(Q)

    if len(std_heads) > 0:
        out[:, std_heads] = sdpa(
            Q[:, std_heads], K[:, std_heads], V[:, std_heads],
            att_mask_for(K, std_heads), scaling, dropout_p
        )

    if len(mod_heads) > 0:
        out[:, mod_heads] = sdpa(
            Q[:, mod_heads], Kd[:, mod_heads], Vd[:, mod_heads],
            None, scaling, dropout_p
        )

    return out
```

Algorithm 4 Python Pseudocode of Applying Skill Head KL Loss

```python
def skill_guidance_loss(
    all_attn_states,
    observation,               # may contain skill_soft or skill_id
    skill_head,                # linear head: [d] -> [K]
    skill_num_classes,         # K
    action_query_start_idx,
    skill_use_control: bool,
    eps=1e-6,
):
    target_prob = build_skill_soft_label(observation, skill_num_classes, eps)
    if target_prob is None:
        return 0.0

    layer_feats = []
    for layer_idx, attn_data in all_attn_states:
        attn_out = select_skill_attn_out(attn_data, skill_use_control)
        skill_attn_out = attn_out[:, :, action_query_start_idx:, :]    # [B, H, A_len, d]
        feat = skill_attn_out.mean(dim=(1, 2))                         # [B, d]
        layer_feats.append(feat)

    if len(layer_feats) == 0:
        return 0.0

    feat = stack(layer_feats, dim=1).mean(dim=1)                 # [B, d]
    logits = skill_head(feat)                                    # [B, K]

    log_prob = log_softmax(logits, dim=-1)
    return kl_div_batchmean(log_prob, target_prob)


def select_skill_attn_out(attn_data, skill_use_control: bool):
    if (not skill_use_control) and ("skill_origin" in attn_data):
        return attn_data["skill_origin"]
    return attn_data["skill"]


def build_skill_soft_label(observation, K, eps=1e-6):
    if hasattr(observation, "skill_id") and observation.skill_id is not None:
        ids = observation.skill_id.long()
        if ids.ndim >= 2 and ids.shape[-1] == 1:
            ids = ids.squeeze(-1)

        if ids.ndim == 1:
            counts = one_hot(ids, K).float()                  # [B, K]
            T = 1
        else:
            ids_flat = ids.view(ids.shape[0], -1)             # [B, T]
            counts = one_hot(ids_flat, K).float().sum(dim=1)  # [B, K]
            T = ids_flat.shape[1]

        # y_k = (count_k + eps) / (T + K*eps)
        denom = float(T) + float(K) * eps
        return (counts + eps) / denom

    return None


def normalize_prob(p, eps=1e-6):
    p = p + eps
    return p / p.sum(dim=-1, keepdim=True)


def kl_div_batchmean(log_prob, target_prob, eps=1e-9):
    # log_prob:    [B, K]  (log softmax of student logits)
    # target_prob: [B, K]  (teacher/label distribution, sum=1)
    target_prob = target_prob.clamp_min(eps)
    return (target_prob * (target_prob.log() - log_prob)).sum(dim=-1).mean()
```

Algorithm 5 Python Pseudocode of ControlNet-style Dual-Path Control Attention with Zero-Conv Fusion

```python
class ControlAttention:
    def __init__(self, original_attn, *, hidden_size, num_control_heads=2, use_headwise_gate=True):
        self.origin = original_attn
        self.branch = make_control_branch(
            original_attn,
            num_control_heads=num_control_heads,
            use_headwise_gate=use_headwise_gate
        )

        self.num_heads = original_attn.config.num_attention_heads
        self.head_dim  = original_attn.head_dim

        # zero-initialized projection (ControlNet design)
        self.zero_conv = zero_init_linear(hidden_size, hidden_size)

        # optional: expand control Q heads to match origin heads (e.g., 2 -> 8)
        self.has_q_expansion, self.q_expand = maybe_build_q_expansion(
            origin_heads=self.num_heads,
            control_heads=num_control_heads,
            head_dim=self.head_dim
        )

    def dual_path(self, hidden_states):
        B, T, _ = hidden_states.shape

        q0 = self.origin.q_proj(hidden_states)
        k0 = self.origin.k_proj(hidden_states)
        v0 = self.origin.v_proj(hidden_states)
        Q0, K0, V0 = reshape_to_heads(q0, k0, v0, H=self.num_heads, D=self.head_dim)  # [B,H,T,D]

        qc = self.branch.q_proj(hidden_states)  # may include extra dims for head-wise gates
        kc = self.branch.k_proj(hidden_states)
        vc = self.branch.v_proj(hidden_states)

        gate_h = None
        qc_query, qc_gate = maybe_split_query_and_gate(qc)  # qc_gate is optional
        if qc_gate is not None:
            gate_h = reshape_gate(qc_gate, H=self.num_heads)         # [B,H,T,1]
            Qc = reshape_query(qc_query, Hc=self.branch.num_heads)   # [B,Hc,T,D]
        else:
            Qc = reshape_query(qc, Hc=self.branch.num_heads)         # [B,Hc,T,D]

        Kc, Vc = reshape_to_heads(kc, vc, H=self.branch.num_heads, D=self.head_dim)  # [B,Hc,T,D]

        if self.has_q_expansion:
            Qc = expand_heads(Qc, q_expand=self.q_expand, target_H=self.num_heads)   # [B,H,T,D]

        return (Q0, K0, V0), (Qc, Kc, Vc), gate_h

    def fuse(self, origin_out, branch_out):
        # Zero Conv Fusion: y = y + ZeroConv(branch_out)
        return origin_out + self.zero_conv(branch_out)
```

pretrained models. Specifically, we study how different choices in depth feature extraction and processing affect performance, focusing on two aspects: (1) the backbone capacity of the depth encoder, and (2) whether to apply token downsampling to the depth tokens before fusion.

Depth Encoder Variants. We adopt the Depth Anything 3 model as our depth encoder and evaluate three of its released variants: `small`, `base`, and `large`. These differ in parameter count and representational capacity. All variants produce dense depth tokens which are then processed by our Depth Head module.

Downsampling Strategy. We also explore the impact of applying spatial downsampling to the output tokens of the depth encoder. The downsampling operation reduces the number of tokens before they are fed into the transformer layers of the Depth Head, potentially reducing noise and enhancing the focus on salient regions.

Experiments and Results. As summarized in Table V, using the `Depth Anything 3-small` encoder achieves the best average success rate (83.00%) across the RoboTwin 2.0 benchmark. This configuration also performs best or competitively across individual tasks. Notably, while the `large` variant attains strong performance (82.00%), it does not surpass the smaller model despite its increased complexity, suggesting diminishing returns from larger encoders. In contrast, the `base` variant lags behind (69.67%), indicating that encoder size alone does not guarantee better performance.

Moreover, removing token downsampling results in a notable performance drop (68.00%), especially on fine-grained tasks like *Click Bell*. This confirms that moderate spatial compression of depth tokens helps reduce redundancy and improve attention allocation in the Depth Head. Overall, these results highlight the importance of choosing a lightweight but expressive depth encoder and applying spatial abstraction to its outputs for robust depth-aware visuomotor learning.

3) Skill Head: To enhance the policy's ability to capture nuanced behaviors in manipulation tasks, we design dedicated *skill heads* that conditions the action generation on explicit skill representations. This module serves as a semantic interface between high-level task understanding and low-level action prediction. In this section, we study the effects of different label supervision strategies in training the Skill Head, focusing on: (1) the use of discrete hard labels, and (2) the alternative soft label formulation which allows richer skill supervision.

Hard Label Supervision. A straightforward approach is to assign a one-hot *hard label* to each skill instance based on expert annotation. These discrete identifiers are treated as categorical embeddings, enabling the Skill Head to specialize behavior per skill class. While this method is simple and interpretable, it may be limited in representing task ambiguity or overlapping skill boundaries.

Soft Label Supervision. To provide a more flexible representation, we experiment with using *soft labels*, where each skill instance is associated with a probability distribution over multiple skill classes. These distributions are derived from a pretrained skill classifier and reflect the uncertainty and compositionality in skill expressions. The Skill Head is trained to align with these soft distributions, encouraging smoother generalization and richer grounding. For the details of constructing soft labels, please refer to Eq. 18.

Experiments and Results. As shown in Table VI, the Skill Head trained with soft label supervision achieves the best overall performance, with an average success rate of 75.00% across three RoboTwin 2.0 tasks. It consistently outperforms the hard label variant, including on challenging tasks such as *Click Bell* (39% vs. 36%) and *Dump Bin BigBin* (94% vs. 82%), highlighting the benefit of using probabilistic skill distributions to capture diverse manipulation strategies. While the hard label variant performs well on more structured tasks like *Beat Hammer Block* (90%), its limited flexibility leads to lower generalization in tasks with more ambiguity.

4) Feature Fusion via ControlNet-style Adapter: To integrate auxiliary features into the *action decoder* without disrupting its core attention flow, we explore three fusion strategies: direct addition, gated modulation, and zero-initialized convolution (ours). These strategies modulate the main attention stream with external signals in different ways, balancing simplicity, control, and stability.

Direct Addition. A straightforward approach is to directly add the auxiliary attention features $\mathrm{Attn}_{\text{specified}}(\mathbf{x})$ to the main attention stream $\mathrm{Attn}_{\text{main}}(\mathbf{x})$:

$$\mathrm{Attn}(\mathbf{x}) = \mathrm{Attn}_{\text{main}}(\mathbf{x}) + \mathrm{Attn}_{\text{specified}}(\mathbf{x}). \tag{14}$$

While simple and computationally efficient, this method lacks any adaptive control over the fused signal. It may lead to feature interference or training instability, particularly in tasks requiring fine-grained control.

Gated Addition. To enable learnable modulation, we apply a nonlinear transformation to the auxiliary features using a $\tanh$ gate before fusion. This produces the final attention as:

$$\mathrm{Attn}(\mathbf{x}) = \mathrm{Attn}_{\text{main}}(\mathbf{x}) + \tanh\left(\mathrm{Attn}_{\text{specified}}(\mathbf{x})\right). \tag{15}$$

Unlike scalar gating, this formulation allows each element of the auxiliary feature map to be adaptively scaled within $(-1, 1)$, enabling smoother gradients and more expressive control. However, the unbounded nature of the fused signal can still introduce training instability in tasks requiring fine spatial precision.

Zero-initialized Convolution. Inspired by residual modulation in ControlNet, we propose applying a zero-initialized convolutional layer to the auxiliary features before fusion. This yields the final output as:

$$\mathrm{Attn}(\mathbf{x}) = \mathrm{Attn}_{\text{main}}(\mathbf{x}) + \mathrm{ZeroConv}\left(\mathrm{Attn}_{\text{specified}}(\mathbf{x})\right). \tag{16}$$

The zero initialization ensures that the fused path initially behaves as an identity function, preventing early training collapse. The network can then gradually learn to utilize auxiliary

TABLE V: **Ablation Study of Depth Head Design on RoboTwin 2.0 tasks.** We evaluate the impact of encoder capacity and token downsampling in our Depth Head. The best results are achieved using the `Depth Anything 3-small` variant (83.00% average success rate), which balances compactness and representational power. Larger encoders offer no significant benefit and may introduce redundancy. Removing token downsampling leads to a notable drop in performance (68.00%), especially on fine-grained tasks like *Click Bell*, supporting the need for moderate spatial abstraction in depth-guided attention.

Model	Beat Hammer Block	Click Bell	Dump Bin BigBin	Avg
π_0 w/ small encoder **(Ours)**	**96%**	63%	**87%**	**83.00%**
π_0 w/ base encoder	77%	49%	83%	69.67%
π_0 w/ large encoder	92%	**67%**	**87%**	82.00%
π_0 w/ small encoder wo/ downsample	78%	39%	**87%**	68.00%

TABLE VI: **Ablation Study on Skill Head Supervision.** We compare Skill Head variants trained with hard vs. soft label supervision on RoboTwin 2.0 tasks. The soft label variant (ours) achieves the highest average performance (75.00%), showing improved generalization on less structured tasks such as *Click Bell* and *Dump Bin BigBin*.

Model	Beat Hammer Block	Click Bell	Dump Bin BigBin	Avg
π_0 w/ hard labels	90%	36%	82%	69.33%
π_0 w/ soft labels **(Ours)**	**92%**	**39%**	**94%**	**75.00%**

cues in a stable and interpretable manner. Empirically, this design leads to more consistent performance gains across a range of tasks.

Experiments and Results. We evaluate all three fusion methods on the *RoboTwin 2.0* benchmark. As summarized in Table VII, our zero-initialized convolution strategy achieves the best average success rate of 83.33%, significantly outperforming both direct addition (64.00%) and gated addition (73.33%). Notably, the performance on the *Click Bell* task improves the most, demonstrating the benefit of stable and learnable modulation when precise spatial control is required.

D. Ablation on Overall Architecture

Guidance Layers. We study how the choice of guidance layers affects robustness by applying guidance to different subsets of transformer layers in π_0. Specifically, we compare guiding all layers with guiding only one of four layer quartiles, where layers are evenly divided from bottom to top. All settings use the same training protocol and evaluation benchmarks. Table VIII reports performance on LIBERO-Plus benchmark only the third quartile of layers achieves the best overall performance, with a total score of 75.4, outperforming guidance on all layers (74.1) as well as the first, second, and fourth quartiles (74.4, 74.3, and 73.8 respectively). This trend is consistent across multiple task categories and variation types. These results suggest that guidance is most effective when applied to mid-to-upper layers, which likely capture higher-level semantic and task-relevant representations. In contrast, guiding all layers or very early/late layers may dilute the effect of guidance or interfere with low-level feature learning.

Number of Guidance Query Attention Heads. The original π_0 model adopts a Multi-Query Attention (MQA) mechanism in its action decoder, with 8 query heads and a single shared key-value head. In our main experiments, we distill 2 query attention heads per factor (6 out of 8 total heads across all factors). This ablation study investigates the impact of distilling only 1 query attention head per factor (3 out of 8 total heads). As shown in Table IX, using 2 heads per factor yields better overall performance (75.3% vs. 74.3%), suggesting that allocating more dedicated attention heads per factor enhances the model's ability to capture diverse environmental variations.

E. Complete Model Architecture of GuidedVLA

We provide the complete model architecture of GuidedVLA in Table X, detailing every module used in both perception and control pathways. The model integrates a SigLIP vision tower for multi-view visual encoding, a PaliGemma language backbone for multimodal grounding, and a lightweight Gemma-based expert head for action prediction. Optional branches such as the Depth Head, Skill Head, and ControlAttention modules can be toggled depending on the task setup, enabling flexible scaling and specialization. This architecture supports the strong performance of GuidedVLA on diverse visuomotor benchmarks, including RoboTwin 2.0, by unifying visual, linguistic, and temporal modalities within a compact yet expressive framework.

F. Code Details

Our code and dataset will be open-sourced after acceptance.

We developed GuidedVLA based on the codebases of *openpi* (the official release of the π_0 model) and *RoboTwin 2.0*.

During development, we identified several limitations in these two codebases:

TABLE VII: **Ablation Study of Feature Fusion Strategies on RoboTwin 2.0.** We compare three strategies for incorporating auxiliary features into the action decoder: direct addition, elementwise tanh-gated addition, and zero-initialized convolution. The zero-initialized convolution achieves the highest average success rate (83.33%), particularly excelling in precision-sensitive tasks such as *Click Bell*, highlighting the benefit of stable and learnable feature modulation.

Model	Beat Hammer Block	Click Bell	Dump Bin BigBin	Avg
π_0 w/ direct add	67%	37%	88%	64.00%
π_0 w/ gate	87%	42%	91%	73.33%
π_0 w/ zero conv **(Ours)**	**94%**	**62%**	**94%**	**83.33%**

TABLE VIII: **Ablation Study of Guidance Layer Subsets on LIBERO-Plus.** We evaluate guidance on all layers and on four layer quartiles. The third quartile achieves the highest total score (75.4), higher than guiding on all layers (74.1) and the other quartiles (74.4/74.3/73.8), indicating that focusing guidance on a specific layer range improves robustness.

	Camera	Robot	Language	Light	Background	Noise	Layout	Total
π_0 guided on all layers								
Spatial	81.6	61.7	66.2	96.2	95.7	91.7	92.7	82.7
Object	79.5	43.0	81.6	96.0	96.0	91.9	74.7	78.9
Goal	71.6	45.2	42.2	95.0	85.8	83.4	66.1	67.7
Long	51.6	43.5	65.3	83.9	76.8	80.8	79.8	67.5
Avg	70.7	47.9	63.1	92.9	88.1	86.7	77.9	74.1
π_0 guided on first quartile of layers								
Spatial	83.5	59.1	63.1	97.3	95.0	92.6	91.4	82.1
Object	85.1	47.0	85.6	97.0	91.5	96.2	73.2	81.1
Goal	72.1	44.7	40.0	92.1	87.9	83.4	61.6	66.5
Long	50.4	45.5	64.2	88.7	74.4	82.2	83.7	68.5
Avg	72.3	48.8	62.4	93.9	86.8	88.5	76.7	74.4
π_0 guided on second quartile of layers								
Spatial	83.0	58.9	64.9	96.6	95.3	91.2	93.5	82.4
Object	80.3	45.5	83.3	94.3	94.0	94.8	74.4	79.7
Goal	72.3	44.0	39.0	93.9	89.7	87.1	65.4	67.8
Long	55.8	41.2	64.8	86.9	78.9	73.1	86.2	67.8
Avg	72.5	47.0	62.2	93.0	89.1	86.1	79.1	74.3
π_0 guided on third quartile of layers								
Spatial	84.8	58.0	65.4	99.0	95.7	88.6	91.7	82.3
Object	84.6	48.7	82.2	97.3	91.9	91.0	72.5	79.9
Goal	69.6	49.4	52.9	93.5	90.4	88.4	68.7	71.2
Long	46.3	42.5	68.9	81.0	75.8	88.9	82.7	68.4
Avg	70.8	49.4	66.8	92.9	88.1	89.3	78.4	75.4
π_0 guided on fourth quartile of layers								
Spatial	83.8	54.3	65.4	97.6	94.6	89.2	93.2	81.7
Object	83.8	44.5	79.1	93.9	92.7	90.5	75.9	78.9
Goal	71.8	47.7	43.9	92.5	86.5	83.6	66.1	68.2
Long	45.8	44.0	64.5	86.1	76.1	78.1	86.2	67.0
Avg	70.8	47.4	62.6	92.6	87.1	85.1	79.6	73.8

TABLE IX: **Ablation Study on the Number of Guidance Attention Heads.** Comparing the performance of distilling 1 vs. 2 query attention heads per factor across various environment variation types. Distilling 2 heads per factor leads to improved average performance (75.3% vs. 74.3%), indicating that more dedicated attention capacity per factor improves robustness to diverse variations.

	Camera	Robot	Language	Light	Background	Noise	Layout	Total
π_0 w/ one head per factor guided								
Spatial	85.2	59.7	66.7	97.6	95.3	92.8	92.7	83.4
Object	77.3	45.7	80.5	92.9	94.0	93.8	74.9	78.6
Goal	73.0	45.2	41.7	92.8	89.3	85.0	66.1	68.2
Long	48.4	43.0	69.2	86.1	76.8	76.5	84.3	67.5
Avg	70.5	48.0	63.8	92.5	88.5	86.7	78.9	74.3
π_0 w/ two heads per factor guided (Ours)								
Spatial	84.8	58.0	65.4	99.0	95.7	88.6	91.7	82.3
Object	84.6	48.7	82.2	97.3	91.9	91.0	72.5	79.9
Goal	69.6	49.4	52.9	93.5	90.4	88.4	68.7	71.2
Long	46.3	42.5	68.9	81.0	75.8	88.9	82.7	68.4
Avg	70.8	49.4	66.8	92.9	88.1	89.3	78.4	75.3

Data Format Conversion. Both the official LIBERO dataset from *openpi* and our custom-collected RoboTwin 2.0 dataset were originally stored in the LeRobot 2.0 format, which suffers from a critical data loading bottleneck (see Pull Request #2408). LeRobot 3.0 resolves this issue with improved I/O efficiency. To enable faster training and evaluation, we therefore converted both public and private datasets into the LeRobot 3.0 format.

Training Speed Optimization. The default PyTorch training pipeline provided by *openpi* is significantly slower than its JAX counterpart. To address this, we applied `torch.compile` to wrap the model, which led to a noticeable speedup in training efficiency without impacting performance.

Framework and Precision Sensitivity. When training with full `float32` precision, we observe that the model π_0 achieves equivalent performance (90) across both JAX and PyTorch implementations, suggesting that the choice of train/test framework is not a limiting factor. However, when switching to full `bfloat16` training precision, performance degrades significantly (e.g., down to 10) in our setting. This issue is eliminated by using either full `float32` or mixed precision training. We therefore adopt mixed precision by default, which provides a good balance between speed and stability while matching full `float32` performance. Details are reported in Table XI.

Batch Size and Gradient Accumulation. To increase the effective batch size, we implemented gradient accumulation in the training loop. However, this modification did not lead to meaningful performance improvements and in some cases slightly degraded the results. As such, gradient accumulation is disabled by default in our final setup.

Validation Strategy. The official codebase does not include a validation set or evaluation pipeline during training. However, we find that monitoring the convergence of auxiliary objectives—such as object grounding loss and skill prediction loss—is critical to ensuring effective learning. We thus split each dataset into training and validation subsets using a 93:7 ratio, and incorporated open-loop validation loss tracking throughout training. This allows us to verify that auxiliary heads are making meaningful progress, even in the absence of closed-loop task rollouts.

G. *Complete Results for All Datasets*

For LIBERO-plus [27] benchmark, we provide the complete results in Table XII.

For Robotwin 2.0 [15] benchmark, we provide the complete results in Table XIII.

For Real robot experiments, we provide the complete results across all 6 tasks in Table III.

H. *Dataset Construction*

*1) **Object Masks**:* To provide the spatial targets required by Eq. 2, we construct *stage-aware* object masks via a semi-automatic, human-in-the-loop pipeline. Each episode is first partitioned into a sequence of temporal stages, where each stage corresponds to a specific task-relevant object.

We automate the initialization process using Qwen3-VL [3]. For the start frame of each stage, we query Qwen3-VL with the stage description to detect the target object and generate candidate foreground point prompts. Given these VLM-proposed points, we invoke the video tracking capability of SAM2 [71] to propagate the object mask across frames within the stage interval.To ensure high-quality supervision, we implement a final human verification step. This hybrid workflow combines the efficiency of VLM-based auto-labeling

TABLE X: **Model Architecture of GuidedVLA.** This table lists the detailed layer-wise composition of our visuomotor agent, including the vision encoder, language backbone, action decoder, and optional modules such as the Depth Head, Skill Head, and ControlAttention layers. Our design uses a multi-view SigLIP transformer for image encoding, a PaliGemma (Gemma-2B) backbone for multimodal reasoning, and a compact Gemma-300M expert for action prediction. The modular architecture allows for easy integration of spatial and semantic grounding signals, contributing to the strong results achieved by GuidedVLA across manipulation tasks.

Module	Layer Type	Layer Num	Input Shape	Output Shape
		SigLIP Vision Tower (per view, V=3)		
SiglipVisionTransformer	SiglipVisionEmbeddings	1	$(B, 3, 224, 224)$	$(B, 256, 768)$
	SiglipEncoderLayer	12	$(B, 256, 768)$	$(B, 256, 768)$
	LayerNorm	1	$(B, 256, 768)$	$(B, 256, 768)$
		PaliGemma Multi-Modal Projector (per view)		
MultiModalProjector	Linear	1	$(B, 256, 768)$	$(B, 256, 2048)$
		PaliGemma Language Model (Gemma-2B)		
Embed Tokens	Embedding	1	$(B, 48)$	$(B, 48, 2048)$
Language Transformer	GemmaDecoderLayer	18	$(B, 816, 2048)$	$(B, 816, 2048)$
Norm	RMSNorm	1	$(B, 816, 2048)$	$(B, 816, 2048)$
		Action/State/Time Embedding		
State Projection	Linear	1	$(B, 32)$	$(B, 1, 1024)$
Action In Projection	Linear	1	$(B, 50, 32)$	$(B, 50, 1024)$
Time Embedding	Sin/Cos	1	(B)	$(B, 1024)$
Action-Time MLP	Linear + SiLU + Linear	1	$(B, 50, 2048)$	$(B, 50, 1024)$
		Gemma Action Expert (Gemma-300M)		
Expert Transformer	GemmaDecoderLayer	18	$(B, 51, 1024)$	$(B, 51, 1024)$
Norm	RMSNorm	1	$(B, 51, 1024)$	$(B, 51, 1024)$
Action Out Projection	Linear	1	$(B, 50, 1024)$	$(B, 50, 32)$
		Depth Branch (Optional)		
DepthEncoder	DepthAnything + TokenMerging2D	1	$(B, 3, 224, 224)$	$4 \times (B, 16, 1024)$
DepthTokenKVProjector	Linear (K/V)	4	$(B, 16, 1024)$	$(B, 8, 16, 256)$ (K/V)
		Skill Head (Optional)		
Skill Head	Linear	1	$(B, 256)$	$(B, 8)$
		ControlAttention (Optional, 8 Attention Heads for ControlNet-style Adapter)		
PaliGemma Self-Attn	ControlAwareAttention	18	$(B, 816, 2048)$	$(B, 816, 2048)$
Expert Self-Attn	ControlAwareAttention	18	$(B, 51, 1024)$	$(B, 51, 1024)$

with the precision of human oversight, yielding supervision that is both *temporally localized* and *object specific*.

For training, we convert each per-frame binary mask to patch-level targets aligned with the 16×16 image-token grid. Specifically, for each patch $p \in \mathcal{P}$ we average pool the mask pixels inside the patch to obtain a foreground-coverage score $s_p \in [0, 1]$, and then threshold it to obtain a binary patch indicator:

$$m_p = \mathbb{I}[s_p \geq \tau], \qquad p \in \mathcal{P}. \tag{17}$$

Frames outside any annotated stage interval are treated as unlabeled for object supervision. In addition, if the propagated mask is empty for a given view/frame (equivalently, $\sum_{p \in \mathcal{P}} m_p = 0$, typically because the stage-specific object is not visible), we also mark that view/frame as unlabeled and exclude it from Eq. 2.

*2) **Skill Labels**:* To support the semantic intent objective in the Skill Head (Eq. 4), we derive a *soft* target distribution from the stage-wise skill annotations. Each stage is assigned a discrete skill identifier, and the stage label is applied to all timesteps within its interval. Given a segment of T timesteps with skill ids $\{s_t\}_{t=1}^{T}$, we compute a histogram over K skills and normalize it into a probability vector $\mathbf{y} \in \mathbb{R}^K$:

TABLE XI: **Precision and framework ablation for π_0.** Performance remains stable (90) across training/testing frameworks (JAX vs. Torch) when using full `float32` precision. In contrast, full `bfloat16` training leads to a significant drop (10), consistent with LIBERO-Plus reproducibility issues. Mixed-precision training serves as an efficient alternative, achieving the same performance as full `float32`.

Model	Pretrain Ckpt Precision	Train Framework	Test Framework	Training Precision Policy	Performance
π_0	float32	JAX	JAX	float32 (full)	90
π_0	float32	JAX	Torch	float32 (full)	90
π_0	bfloat16	Torch	Torch	bfloat16 (full)	10
π_0	float32	Torch	Torch	float32 (full)	90
π_0	float32	Torch	Torch	mixed precision	90

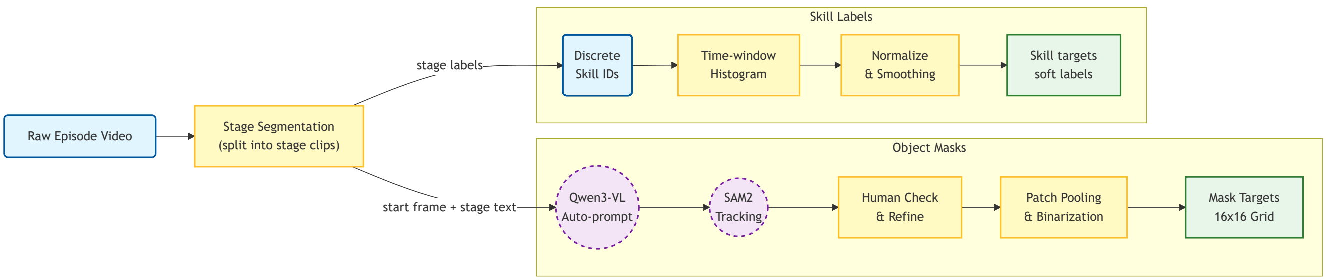


Fig. 10: Overview of the dataset construction pipeline. The raw episode video is segmented into stages, which are then processed to generate both discrete skill labels (top) and stage-aware object masks (bottom) using a VLM-assisted semi-automatic workflow.

$$y_k = \frac{\sum_{t=1}^{T} \mathbb{I}[s_t = k] + \epsilon}{\sum_{j=0}^{K-1} \sum_{t=1}^{T} \mathbb{I}[s_t = j] + K\epsilon}, \qquad k = 0, \ldots, K-1, \tag{18}$$

where ϵ is a small constant for numerical stability. When only one skill appears in the segment, $\mathbf{y}$ reduces to a smoothed one-hot target; when multiple skills occur, $\mathbf{y}$ reflects their relative prevalence. This construction is directly matched to the KL-divergence loss in Eq. 4, providing stable supervision that encourages each designated skill head to encode trajectory-level intent rather than purely step-wise cues.

I. Experiment Details

LIBERO-Plus. To evaluate robustness, we train all models solely on the official LIBERO dataset, annotated via the pipeline described in Section H. Evaluation is performed zero-shot on the full *LIBERO-Plus* benchmark to assess generalization. We adopt a two-stage training strategy: (1) we first fine-tune the pretrained π_0 on the *LIBERO* dataset without any guidance; (2) we then continue training with auxiliary guidance using object and skill losses, weighted by $w_{\text{obj}} = 0.1$ and $w_{\text{skill}} = 0.1$.

For both stages, we use the AdamW optimizer with a cosine learning rate schedule: 1,000 warmup steps, peak learning rate 2.5×10^{-5}, decaying to 2.5×10^{-6}. Training uses a global batch size of 64 and an action chunk size of 50 across 8 NVIDIA H200 GPUs, with mixed precision enabled. Evaluation is conducted on a single NVIDIA RTX 4090 GPU, with the VLM backbone in `bfloat16` precision and the action decoder in `float32`. All models, including the π_0 baseline, are trained and evaluated under identical settings for fair comparison. For ablations on training hyperparameters, refer to Section J.

RoboTwin 2.0. We evaluate on 8 representative tasks from *RoboTwin 2.0*: *Adjust Bottle*, *Beat Hammer Block*, *Click Bell*, *Dump Bin BigBin*, *Moving PlayingCard away*, *Lift Pot*, *Place Burger Fries*, and *Place Can Basket*. For each task, we collect 1,000 demonstration trajectories in randomized environments. Training follows the same optimizer, auxiliary weights, and precision setup as above. Each model is trained for 30k steps using 4 NVIDIA H200 GPUs, with a global batch size of 32 and an action chunk size of 10. Evaluation is done on a single RTX 4090 GPU using the same precision settings. Success rates are computed over 100 rollouts per task.

Real-World. We consider six real-world tasks, each with approximately 50 human demonstration episodes. Each episode is automatically annotated with object, skill, and geometry signals using our developed labeling tool. Models are trained on two NVIDIA H200 GPUs and evaluated on an RTX 4090. We follow standard training and inference procedures to ensure a fair and reproducible comparison.

Training Curves. Figure **??** illustrates the training and validation loss curves throughout our two-stage training pipeline for LIBERO-Plus benchmark In **Stage-1**, we fine-tune the

TABLE XII: Full Results on LIBERO-Plus Benchmark.

	Camera	Robot	Language	Light	Background	Noise	Layout	Total
DreamVLA								
Spatial	79.3	46.0	64.4	96.9	96.0	93.1	90.1	79.7
Object	85.4	38.4	80.2	94.3	93.5	91.9	77.9	79.0
Goal	58.5	40.8	39.7	82.7	80.8	84.4	59.3	61.7
Long	39.2	38.9	72.7	67.5	63.3	72.6	69.2	59.8
Avg	65.0	40.9	63.5	85.7	82.7	85.0	74.0	69.9
AdaMoE								
Spatial	55.1	12.0	20.8	72.9	76.4	62.4	69.1	51.0
Object	73.7	14.5	29.6	85.5	83.9	62.2	69.5	57.9
Goal	65.9	25.4	12.9	81.7	81.9	58.6	64.9	53.3
Long	22.1	17.3	20.5	53.6	55.0	52.1	58.0	38.1
Avg	53.8	17.5	20.6	73.7	73.8	58.6	65.8	50.1
π_0								
Spatial	71.3	52.3	68.2	92.8	91.9	87.2	87.3	77.7
Object	76.3	33.2	79.1	92.9	87.9	87.7	71.2	74.1
Goal	63.7	40.1	45.1	79.2	82.6	81.5	51.5	61.4
Long	39.6	35.1	62.3	78.1	70.6	74.5	70.5	60.1
Avg	62.3	39.8	63.1	86.0	82.8	82.4	69.6	68.2
π_0 **w/ object head**								
Spatial	73.7	46.9	63.3	94.9	92.2	87.7	90.4	77.4
Object	89.4	35.7	77.7	97.0	90.3	92.2	77.4	78.8
Goal	69.9	43.3	42.7	93.9	92.9	84.7	62.8	67.5
Long	41.5	34.6	67.1	79.2	74.4	76.3	76.6	62.7
Avg	68.2	40.0	62.1	91.4	87.2	85.0	76.5	71.5
π_0 **w/ skill head**								
Spatial	80.9	48.0	62.6	95.9	95.9	89.2	93.2	79.8
Object	88.6	33.2	79.7	94.9	94.9	95.5	73.0	78.6
Goal	69.1	44.3	44.9	91.8	91.8	83.6	58.1	66.6
Long	40.8	37.4	68.3	77.4	77.4	74.9	80.4	63.6
Avg	69.3	40.5	63.2	90.2	87.6	85.5	75.5	71.8
π_0 **w/ depth head**								
Spatial	81.6	54.0	69.5	96.2	92.8	88.9	92.2	81.4
Object	84.1	44.7	79.7	92.9	92.5	92.9	73.9	79.0
Goal	70.6	44.0	45.1	89.2	86.4	84.7	53.6	65.4
Long	38.4	34.1	71.3	83.9	64.1	76.9	73.4	61.8
Avg	68.1	43.9	65.8	90.7	83.4	85.6	72.8	71.7
GuidedVLA (Ours)								
Spatial	84.8	58.0	65.4	99.0	95.7	88.6	91.7	82.3
Object	84.6	48.7	82.2	97.3	91.9	91.0	72.5	79.9
Goal	69.6	49.4	52.9	93.5	90.4	88.4	68.7	71.2
Long	46.3	42.5	68.9	81.0	75.8	88.9	82.7	68.4
Avg	70.8	49.4	66.8	92.9	88.1	89.3	78.4	75.4

TABLE XIII: **RoboTwin 2.0 Benchmark Full Results.**

Model	Adjust Bottle Moving PlayingCard away	Beat Hammer Block Lift Pot	Click Bell Place Burger Fries	Dump Bin BigBin Place Can Basket	Avg
π_0	97% 79%	78% 92%	35% 85%	89% 64%	77.38%
π_0 w/ object head	**99%** 91%	94% 95%	62% 93%	**94%** 66%	86.75%
π_0 w/ skill head	98% 93%	92% 96%	39% 95%	**94%** 73%	85.00%
π_0 w/ depth head	98% 97%	**96%** 82%	63% 94%	87% 74%	86.38%
π_0 w/ all heads **(Ours)**	**99%** **98%**	95% **99%**	**65%** **98%**	93% **78%**	**90.63%**

pretrained policy π_0 solely on the LIBERO dataset without auxiliary guidance. Both training and validation losses steadily decrease and converge, confirming that the base policy adapts well to the core supervision. In **Stage-2**, we introduce auxiliary supervision in the form of attention and skill losses, weighted equally ($w_{\text{obj}} = w_{\text{skill}} = 0.1$). We observe that all losses—including the main, attention, and skill objectives—exhibit stable convergence. This demonstrates that the auxiliary losses are both learnable and helpful, without introducing instability. Moreover, validation losses (especially for skill and attention heads) follow a similar downward trend, indicating effective generalization beyond the training set. Overall, the training dynamics validate the robustness and consistency of our two-stage optimization strategy.

J. Training Hyperparameter Ablation.

We conduct an extensive ablation study on key training hyperparameters, including supervision weights, number of control heads, batch size, training steps, and whether skill supervision is used. As shown in Table XIV, reducing both supervision weights to $(0.01, 0.01)$ consistently improves performance across all perturbation tracks. The optimal configuration—8 control heads, batch size 64, and low supervision weights—achieves the best average score of 85.6.

Increasing training steps from 30k to 60k yields only marginal benefit (Avg=83.9), indicating early convergence. We also observe that scaling up either control heads or batch size independently brings moderate gains, but the best performance emerges when both are increased jointly. Asymmetric weighting, such as lowering only object loss to $(0.01, 0.001)$, results in a clear drop in performance (Avg=81.4), highlighting the importance of balanced supervision. Furthermore, removing skill supervision altogether leads to notable degradation, confirming its necessity for robust generalization.

K. Real-World Experiments: Deployment & Evaluation

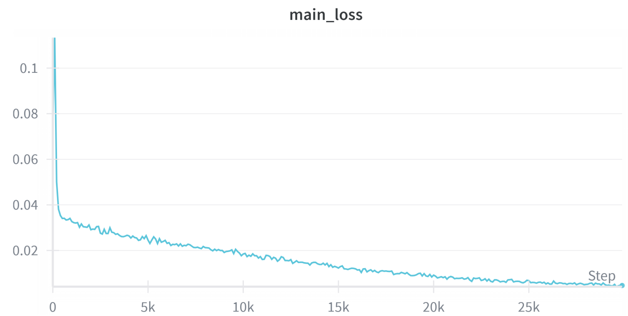
1) Real-world Deployment Setup: We deploy GuidedVLA for real-world inference on a single NVIDIA RTX 4090 GPU. At each inference cycle, given RGB observations from the robot-mounted cameras, GuidedVLA outputs a **50-step** action chunk. The chunk is parameterized at an effective rate of **20 Hz**. On the client side, an executor upsamples the 20 Hz keyframes via linear interpolation to produce smooth trajectories, and streams commands to the low-level controller at a **50 Hz** control rate. To ensure stable execution, after publishing each chunk we wait for joint-position convergence with a **4.0 s** timeout; if the timeout is reached, we proceed to the next inference cycle. Camera extrinsics are set to match the training distribution. The third-person camera is mounted at an elevation angle of approximately **45°**, at roughly **60 cm** above the workspace center (Figure 4). All test objects are placed within a **50 cm** $\times$ **60 cm** $\times$ **30 cm** workspace in front of the robot. During deployment, depth-aware inference is performed by a **frozen Depth Anything V3** encoder (small variant) integrated into the model. Its depth features are spatially downsampled to match the token resolution, and then injected into a dedicated **Depth Head** within the attention pathway.

2) Detailed Evaluation Setup: **Task Definitions & Success Criteria:** We evaluate three household manipulation tasks on the ALOHA AgileX dual-arm platform (T1–T3) and three laboratory manipulation tasks on the PSI-Bot dual-arm platform (T4–T6). Each evaluation trial lasts for at most **120 s** and terminates early once the success condition is met. Unless otherwise specified, we require a **1 s dwell time**: the relevant objects must remain stable in the target configuration for at least 1 s without human intervention.

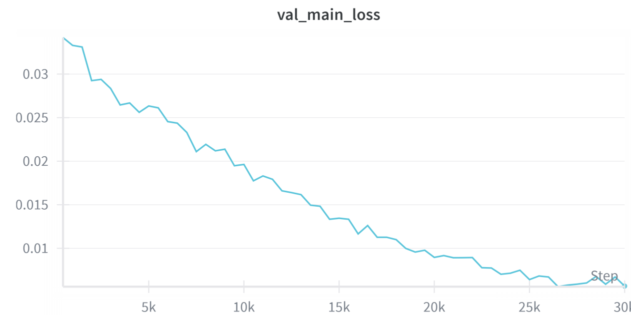
ALOHA household tasks:

(T1) Pick up fruits and vegetables. The robot must place the green pepper and carrot onto the plate, and place the strawberry into the bowl. A trial is successful if the pepper and carrot are both inside the plate region and the strawberry is inside the bowl region.

(T2) Stack bowls and place on the first shelf. The robot stacks two bowls and places the stacked bowls onto the first shelf. A trial is successful if the bowls form a stable stacked configuration and the stack is placed within the designated shelf region.



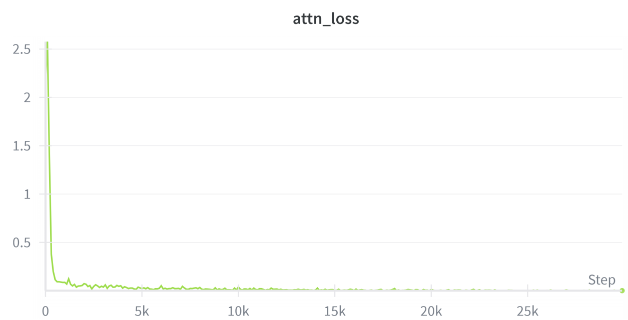
(a) Stage-1 (no guidance): Train main loss.



(b) Stage-1 (no guidance): Val main loss.

(c) Stage-2: Train main loss.

(d) Stage-2: Val main loss.



(e) Stage-2: Train attention loss ($w_{\text{obj}} = 0.1$).

(f) Stage-2: Val attention loss ($w_{\text{obj}} = 0.1$).

(g) Stage-2: Train skill loss ($w_{\text{skill}} = 0.1$).

(h) Stage-2: Val skill loss ($w_{\text{skill}} = 0.1$).

Fig. 11: **Training curves during two-stage optimization on LIBERO.** Each row shows train (left) and validation (right) curves for different objectives. **Row 1**: Stage-1 training without guidance, optimizing only the main loss. **Row 2**: Stage-2 training with auxiliary guidance, where the main loss continues to converge. **Row 3-4**: Auxiliary attention and skill losses (with $w_{\text{obj}} = 0.1$, $w_{\text{skill}} = 0.1$) also converge stably, showing their learnability and effectiveness. All objectives—including auxiliary heads—achieve low validation error, suggesting strong generalization across tasks in the LIBERO-Plus benchmark.

TABLE XIV: **Ablation study on training hyperparameters.** We evaluate training hyperparameters on the Light, Background, and Layout tracks of the LIBERO-Plus benchmark. The baseline (2 heads, batch size 32, $(w_{\text{skill}}, w_{\text{obj}}) = (0.1, 0.1)$) achieves Avg=79.6. Reducing supervision weights to $(0.01, 0.01)$ consistently improves performance. Further gains are achieved by scaling up control heads and/or batch size, with the best configuration (8 heads, batch size 64, (0.01, 0.01)) reaching Avg=85.6. Notably, increasing training steps from 30k to 60k provides only marginal benefit. Asymmetric weighting (e.g., $(0.01, 0.001)$) underperforms the symmetric ($(0.1, 0.1)$) setting. Removing skill supervision degrades performance. We also observe that larger batch size helps more than deeper control heads in isolation, but combining both is best.

Steps(k)	Control heads	Batch size	w_{skill}	w_{obj}	Light	Background	Layout	Avg
30	2	32	0.1	0.1	86.0	82.8	72.6	79.6
30	2	32	0.0	0.0	87.5	84.9	68.9	79.2
30	2	32	0.1	0.1	91.2	87.1	76.0	83.8
60	8	64	0.01	0.01	**91.6**	**88.0**	75.2	83.9
30	8	64	0.01	0.01	**92.9**	**88.1**	**78.4**	**85.6**
30	8	64	0.01	0.001	89.3	83.6	73.9	81.4
30	2	64	0.01	0.01	92.1	86.5	74.3	83.2
30	2	64	0.0	0.01	87.9	82.3	73.7	80.5
30	8	32	0.01	0.01	89.6	85.3	72.2	81.3
30	16	32	0.01	0.01	89.9	87.7	74.6	83.0
30	16	64	0.01	0.01	89.4	86.0	74.4	82.3
30	8	64	0.01	0.01	88.2	84.4	71.8	80.4

(T3) Clean the tabletop (sweep → dustpan → pour → return). The robot sweeps trash into the dustpan with a broom, pours the trash from the dustpan into the tray, and returns both the broom and dustpan back to the table. A trial is successful if the robot completes the pouring action over the tray and returns the tools to the table.

PSI-Bot laboratory tasks:

(T4) Pick up the beaker into the heating mantle. The robot picks up a beaker and places it into the heating mantle. A trial is successful if the beaker bottom is seated inside the mantle opening (i.e., inserted into the cavity).

(T5) Stack small beakers inside a large beaker. The robot places small beakers into a large beaker. A trial is successful if the small beakers are contained within the large beaker.

(T6) Heat the beaker (place the asbestos mesh, then place the beaker on it). The robot first places the asbestos mesh on the lower level of the iron stand, and then places the beaker on top of the mesh. A trial is successful if the mesh is placed on the designated lower support ring and the beaker is stably placed on the mesh.

L. Real-World Generalization Settings

We evaluate three real-robot generalization regimes: **in-domain (positional)**, **scene**, and **lighting**. Each trial is first reset to a canonical task layout and then randomized according to *exactly one* regime; we do not combine multiple shifts within a single trial.

a) In-domain (positional) generalization.: We perturb the initial object placement within the training distribution by sampling from a 3×3 grid of 9 discrete anchors centered at the nominal pose. Adjacent anchors are spaced by 1–2 cm (approximately within ± 2 cm per axis relative to the nominal position), while keeping task semantics unchanged.

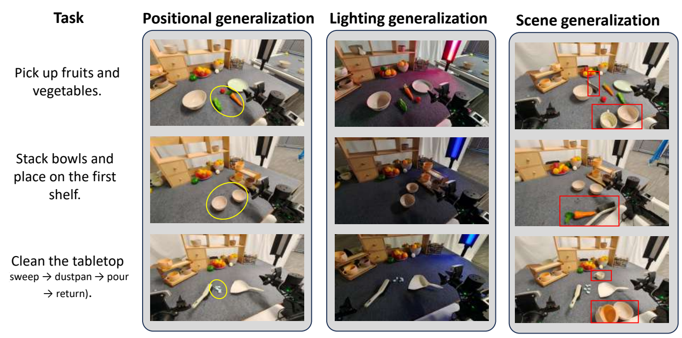


Fig. 12: **ALOHA real-world generalization settings (T1–T3).** From left to right: **in-domain (positional)** perturbations using a 3×3 anchor grid, **lighting** shifts with colored illumination, and **scene** shifts by adding distractor objects.

b) Scene generalization.: We introduce clutter by adding 3–5 distractor objects per trial, sampled from the same domain as the task (household items for ALOHA tasks, lab items for PSI-Bot tasks). Distractors are placed to avoid occluding target objects and to keep the nominal manipulation corridor feasible, thereby inducing appearance/context shifts without altering the intended task.

c) Lighting generalization.: We change illumination using colored decorative lighting with three color settings. Lighting is kept constant within each trial and constrained not to render target objects visually ambiguous, inducing appearance shifts while preserving task observability.

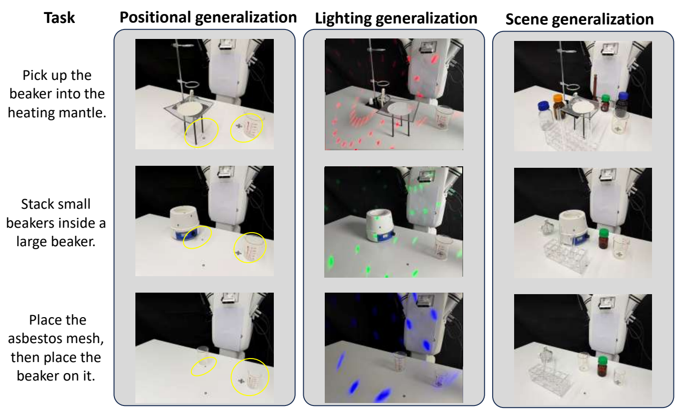


Fig. 13: **PSI-Bot real-world generalization settings (T4–T6).** From left to right: **in-domain (positional)** perturbations using a 3×3 anchor grid, **lighting** shifts with colored illumination, and **scene** shifts by adding distractor objects.

M. Visualization of LIBERO-Plus Results

To complement the quantitative success rates on simulation-based benchmarks in Table XII, we visualize representative *successful* policy rollouts from the LIBERO-Plus benchmark, covering its four task suites: **spatial**, **object**, **goal**, and **long**. Each visualization shows a sequence of 7 keyframes sampled from a successful episode, covering the stages of approach, interaction, and completion. From top to bottom, the rows visualize: RGB image, object head attention, predicted depth map, and depth head attention.

1) Object attention is stage-aware: The object-specialized head dynamically shifts its attention across the episode. In early stages, the attention is concentrated on the object of interest (e.g., a bowl or a book), enabling precise targeting for grasping. As the robot transitions toward goal states, the attention gradually shifts to the target container or placement area, clearly demonstrating a *stage-wise* semantic understanding of task progression.

2) Depth attention captures geometric awareness: The depth head consistently highlights meaningful spatial regions, including both the robot arm and task-relevant objects. This behavior enables *geometry-aware reasoning*, particularly useful for tasks involving occlusion, stacking, or motion planning. For example, in stacking tasks or when reaching into a container, the depth attention map exhibits strong focus on object contours and their relative positioning, helping the policy plan precise and feasible motions.

N. Visualization of Real-World Result

1) Visualization of Real-World Tasks Rollouts: To complement the quantitative success rates in Table II, we visualize representative *successful* real-robot rollouts under the three distribution shifts defined in Sec. L, following the evaluation protocol and success criteria in Sec. K2. For each task, we show a 3-row keyframe grid with **7** manually selected stages (approach, grasp, transport, placement, and completion). Rows correspond to **in-domain (positional)**, **lighting**, and **scene** shifts (top to bottom).

2) Head-wise Mechanism Visualization on Real-World Robots: To further substantiate that our decoupled supervision indeed induces factor-specific behaviors, we provide head-wise diagnostics on real robots from both quantitative and qualitative perspectives. We align each specialized head with the tasks where its factor is most critical: **Object** head $\rightarrow$ **T1** (pick up fruits and vegetables) and **T4** (pick up the beaker), **Depth/Geometry** head $\rightarrow$ **T2** (stack bowls) and **T5** (stack beakers), and **Skill/Temporal** head $\rightarrow$ **T3** (clean the tabletop) and **T6** (Heat the beaker). This alignment allows us to isolate each head's contribution without conflating unrelated failure modes.

a) Quantitative attribution: Table ???III reports success rates for the base $0\pi_0 policy, threesingle - headvariants, andthefullGuidedVLAunderthesamethreedistributio$ $domain, scene, andlighting).Single - headvariantsareevaluatedonlyontheiralignedtasks(otherentriesarer$ $-'')andserveasdiagnosticablationsratherthanstandalonegeneral - purposepolicies.$

b) Qualitative mechanism evidence: To complement the head-wise success rates in Table ???III, we also provide qualitative evidence that each specialized head exhibits the intended factor-specific behavior on its aligned tasks.

Specifically, Fig. ???28 visualizes **object grounding** by overlaying the attention from the object-specialized head on RGB frames at matched key stages. Fig. ???29 visualizes **depth/geometry reasoning** by showing depth cues together with attention overlays from the depth/geometry-specialized head. Fig. ???30 visualizes **skill progression** on a multi-stage task, where $0\pi_0 mayskiprequiredsub - stepswhileGuidedVLAcompletestheintendedsequence.Allexamplesf$

O. Failure Case Analysis (Tasks 1–6)

We analyze representative failure modes of the baseline π_0 on two real-robot platforms: (i) **ALOHA AgileX** for household tasks (T1–T3) and (ii) **PSI-Bot** (*Realman RM63 + DexHand2 Pro*, dual Intel D435) for chemical-lab tasks (T4–T6). Across both domains, failures consistently cluster into three manipulation-critical factors—**object grounding**, **metric geometry/clearance**, and **temporal skill progression**. Figs. 31 and 32 visualize representative failures (panels (a)–(c) correspond to Tasks 1–3 and 4–6, respectively). Unless noted otherwise, examples are under in-domain conditions with nominal object placement.

1) Object grounding failures: The policy executes *phantom grasps* by approaching empty space near the target, or grasps with an *offset* that causes slippage at lift-off. This is most evident for small objects in household scenes (Fig. 31a) and becomes more severe for transparent glassware in the lab due to refraction/specularities (Fig. 32a, top).

π_0 relies on incidental appearance cues (contrast/highlights) rather than invariant target identity and precise spatial alignment, making grounding brittle under appearance changes.

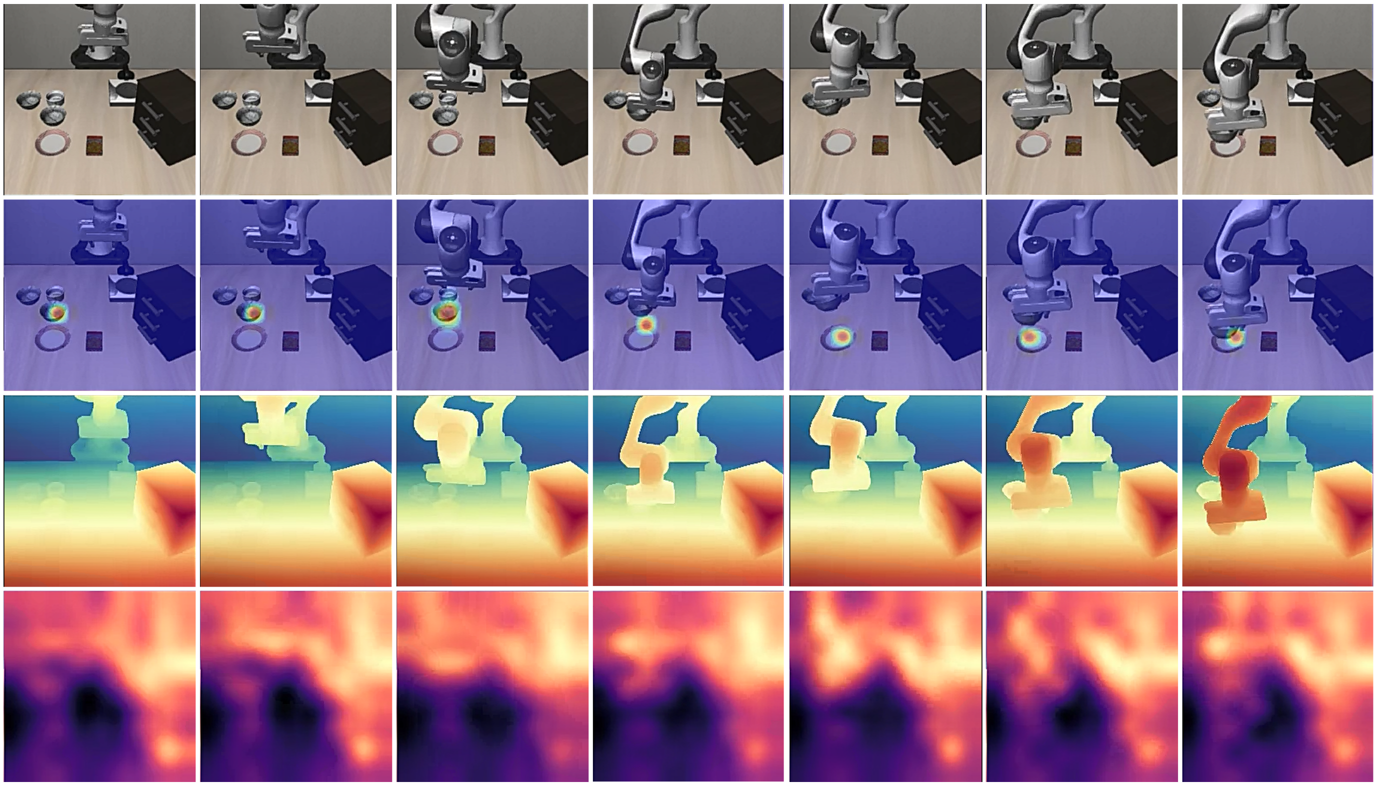


Fig. 14: **LIBERO-Plus rollout visualization (spatial task suite of LIBERO-Plus).** Each column corresponds to one stage in the whole episode, with 7 stages in total.

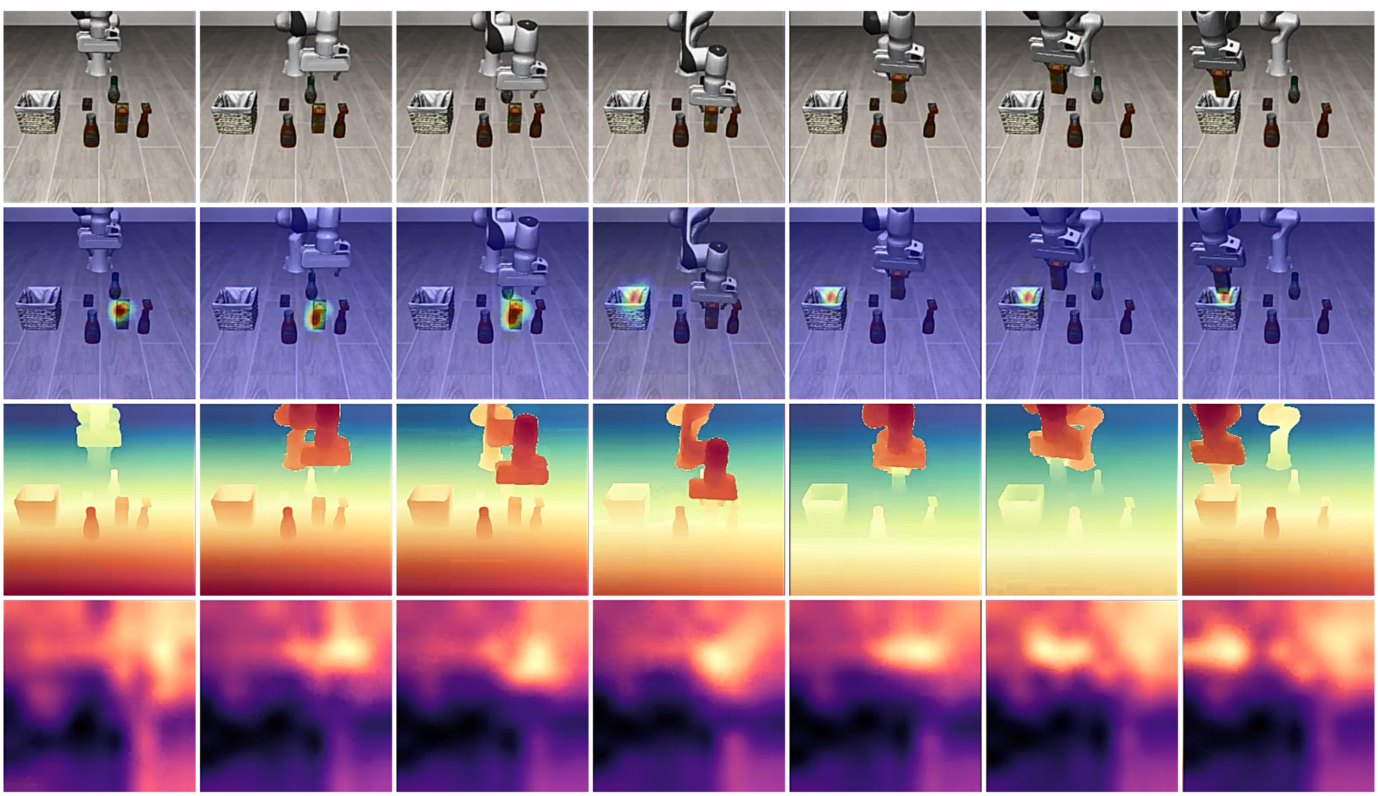


Fig. 15: **LIBERO-Plus rollout visualization (object task suite of LIBERO-Plus).** Each column corresponds to one stage in the whole episode, with 7 stages in total.

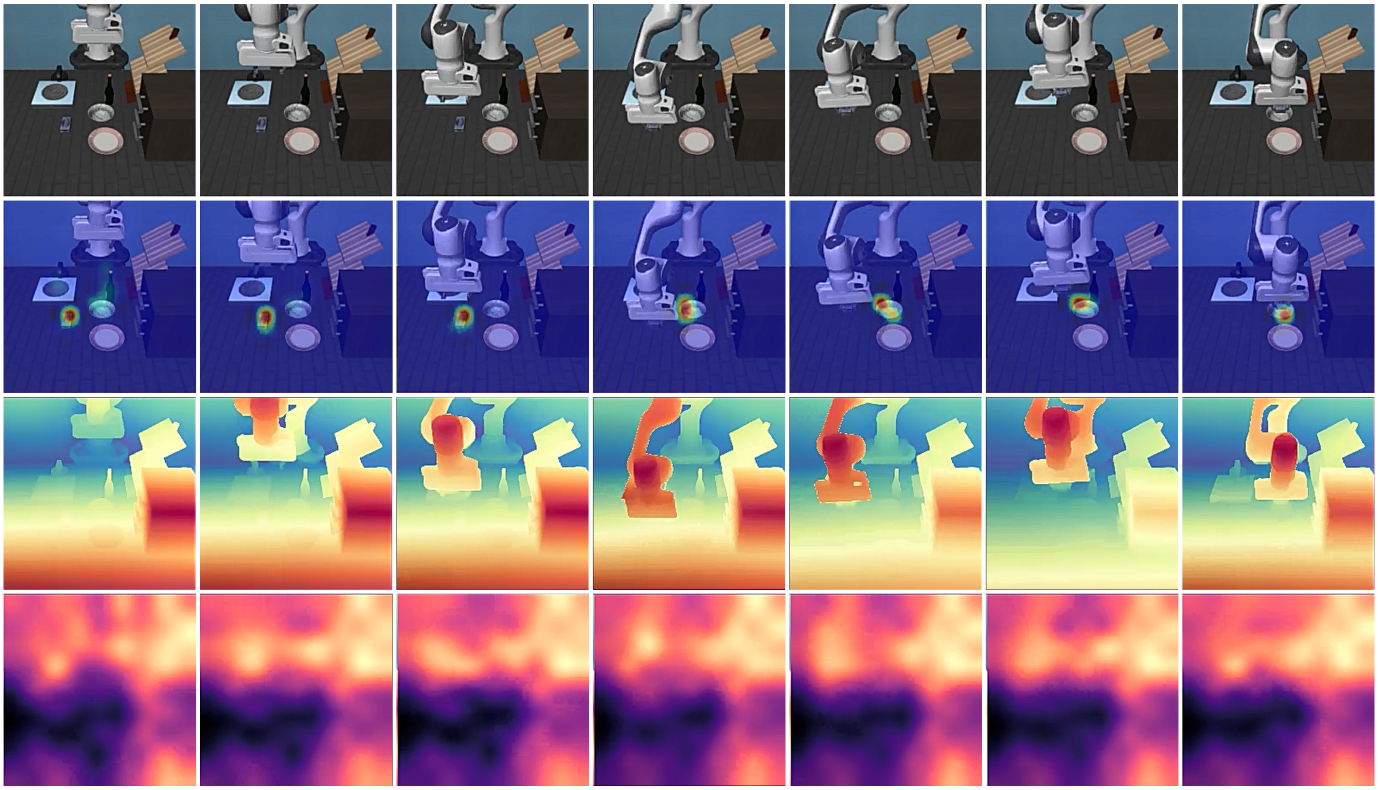


Fig. 16: **LIBERO-Plus rollout visualization (goal task suite of LIBERO-Plus).** Each column corresponds to one stage in the whole episode, with 7 stages in total.

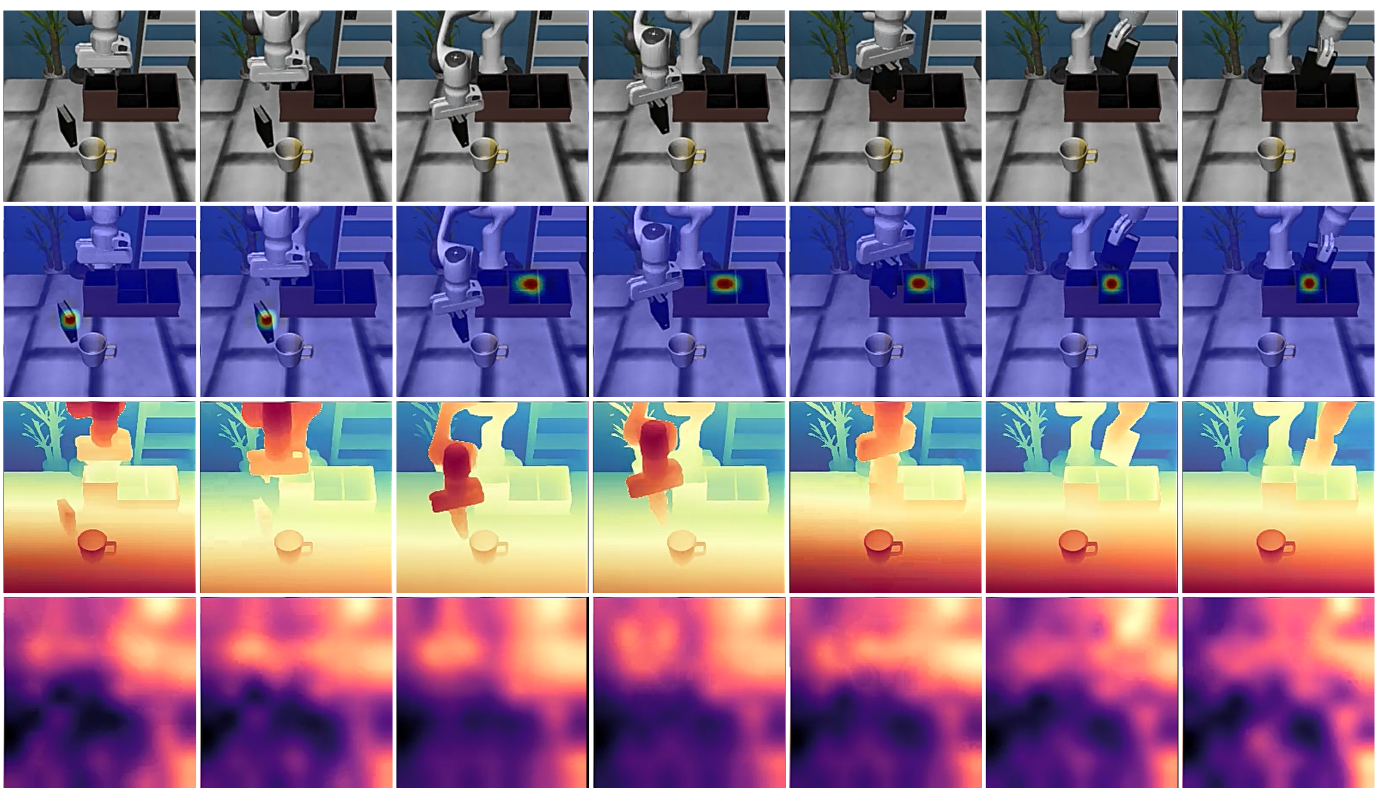


Fig. 17: **LIBERO-Plus rollout visualization (long task suite of LIBERO-Plus).** Each column corresponds to one stage in the whole episode, with 7 stages in total.

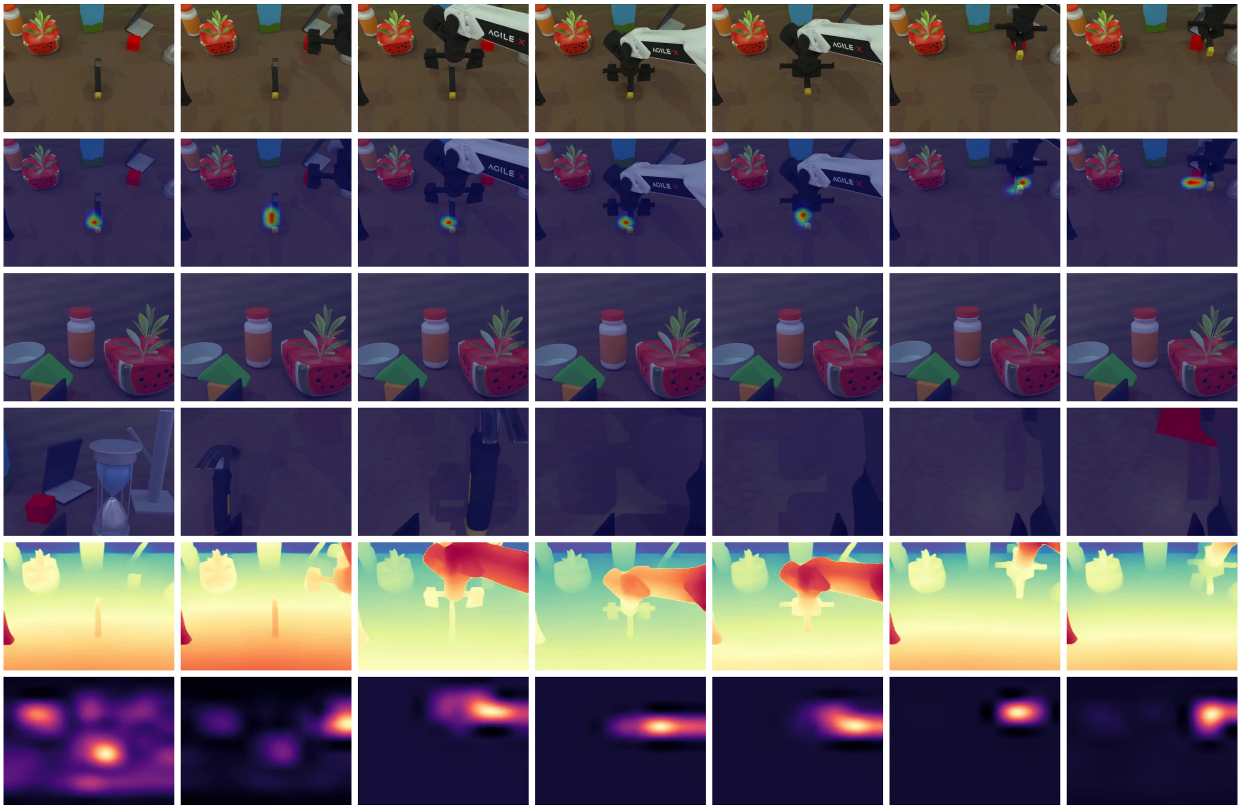


Fig. 18: **Robotwin 2.0 rollout visualization (beat block hammer).** Each column corresponds to one stage in the whole episode, with 7 stages in total.

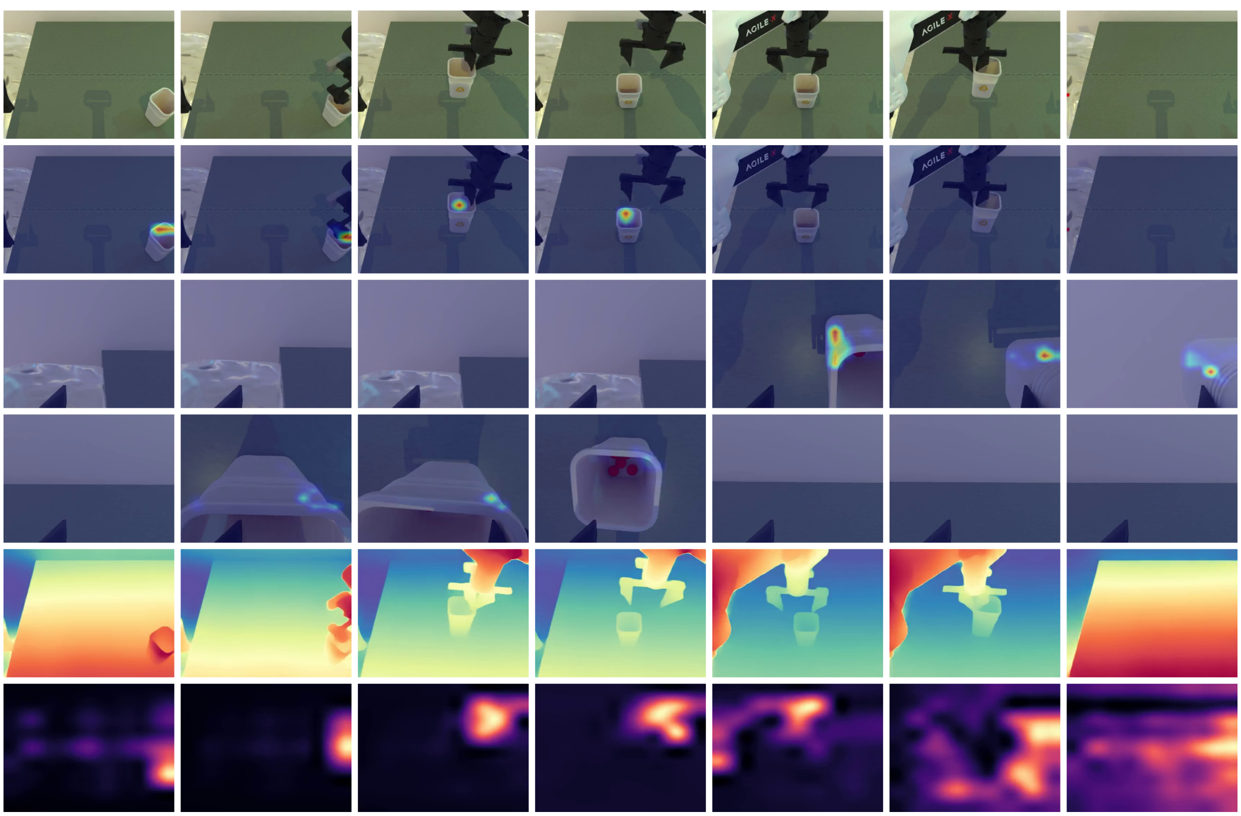


Fig. 19: **Robotwin 2.0 rollout visualization (dump bin bigbin).** Each column corresponds to one stage in the whole episode, with 7 stages in total.

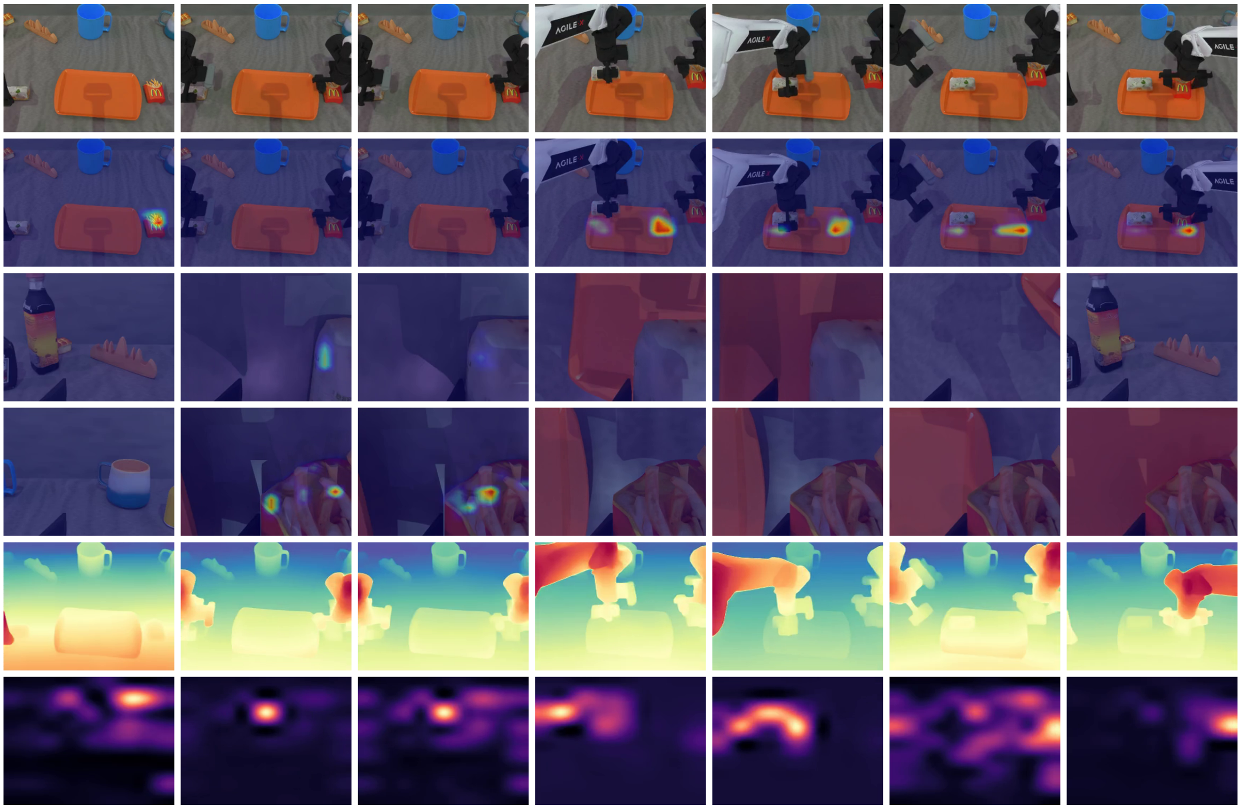


Fig. 20: **Robotwin 2.0 rollout visualization (place burger fries).** Each column corresponds to one stage in the whole episode, with 7 stages in total.

Fig. 21: **Robotwin 2.0 rollout visualization (place can basket).** Each column corresponds to one stage in the whole episode, with 7 stages in total.

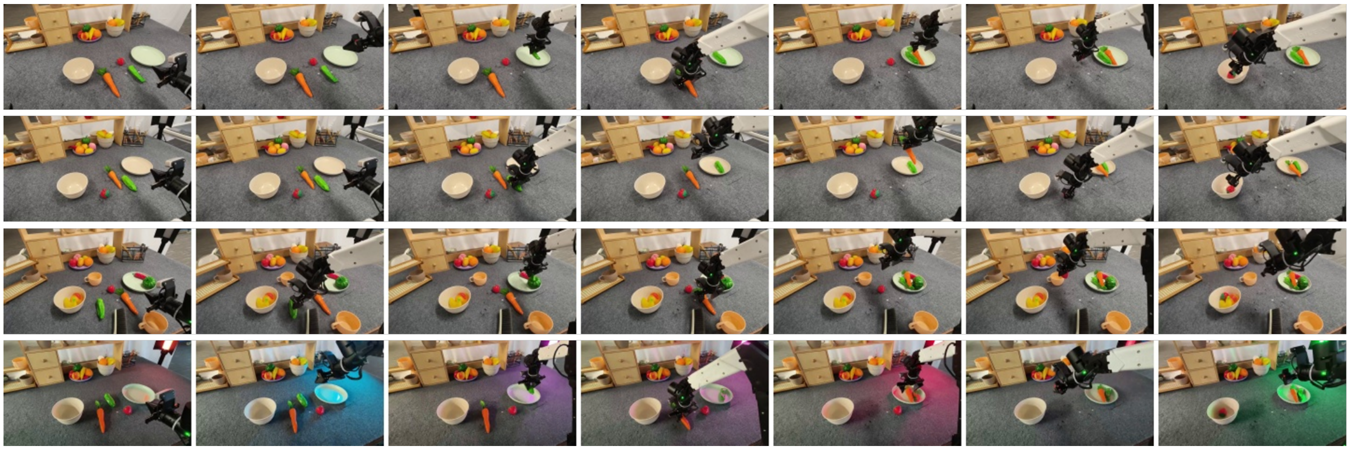


Fig. 22: **Real-robot rollout visualization (ALOHA, T1) under distribution shifts.** Rows: in-domain (positional) / lighting / scene (top to bottom). Columns show 7 key stages of a representative successful trajectory.

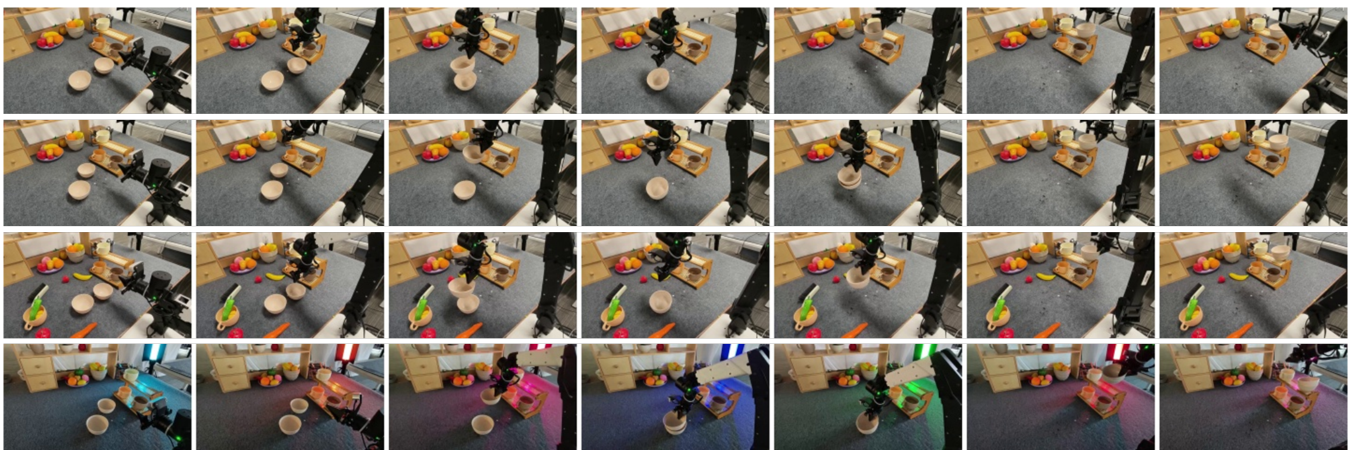


Fig. 23: **Real-robot rollout visualization (ALOHA, T2) under distribution shifts.**Rows: in-domain (positional) / lighting / scene (top to bottom). Columns show 7 key stages of a representative successful trajectory.

2) Metric geometry and clearance failures: The policy fails when millimeter-level depth and clearance are required: *half-grasp* on nested bowls due to insufficient insertion depth (Fig. 31b), rim collisions during heating-mantle insertion (Fig. 32a, bottom), beaker–beaker collisions during nesting under clutter (Fig. 32b, bottom), and collisions with the ring structure from inaccurate stand geometry localization (Fig. 32c, top). Implicit geometric cues from RGB are insufficient for precise insertion/stacking with tight clearances, especially under clutter and reflective materials.

3) Temporal skill collapse in multi-stage execution (T3/T6): The policy completes a visually salient subgoal but skips required subsequent stages, e.g., pouring succeeds but the tool-return phase is omitted in tabletop cleaning (Fig. 31c), and premature release before stabilization causes roll-off in ring-stand assembly (Fig. 32c, bottom). Without explicit supervision for stage awareness, the action decoder can collapse to a short-horizon mode and fail to maintain long-horizon intent.

P. Limitations and Future Work

Our method requires manual selection of task-relevant factors, which can be domain-dependent. Automating factor discovery, exploring additional factors (e.g., force/torque reasoning), and investigating more general head specialization strategies are promising directions for future research.

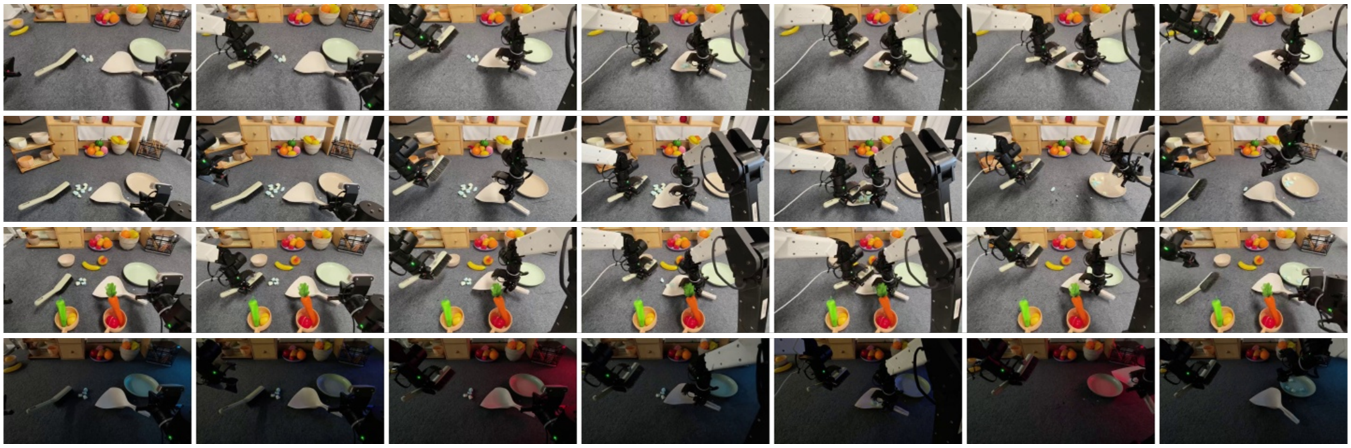


Fig. 24: **Real-robot rollout visualization (ALOHA, T3) under distribution shifts.** Rows: in-domain (positional) / lighting / scene (top to bottom). Columns show 7 key stages of a representative successful trajectory.

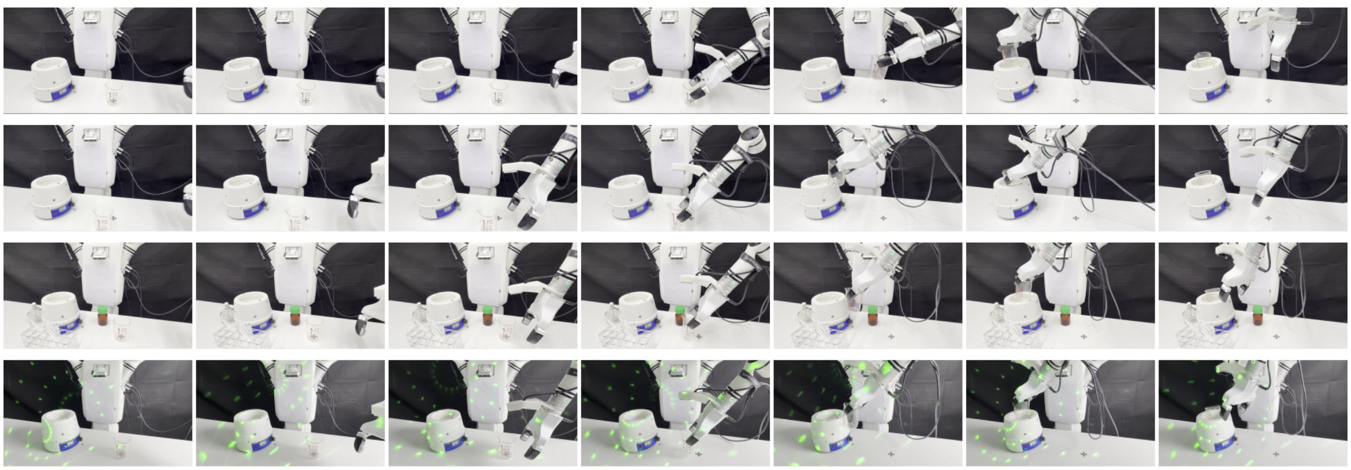


Fig. 25: **Real-robot rollout visualization (PSI-Bot, T4) under distribution shifts.** Rows: in-domain (positional) / lighting / scene (top to bottom). Columns show 7 key stages of a representative successful trajectory.

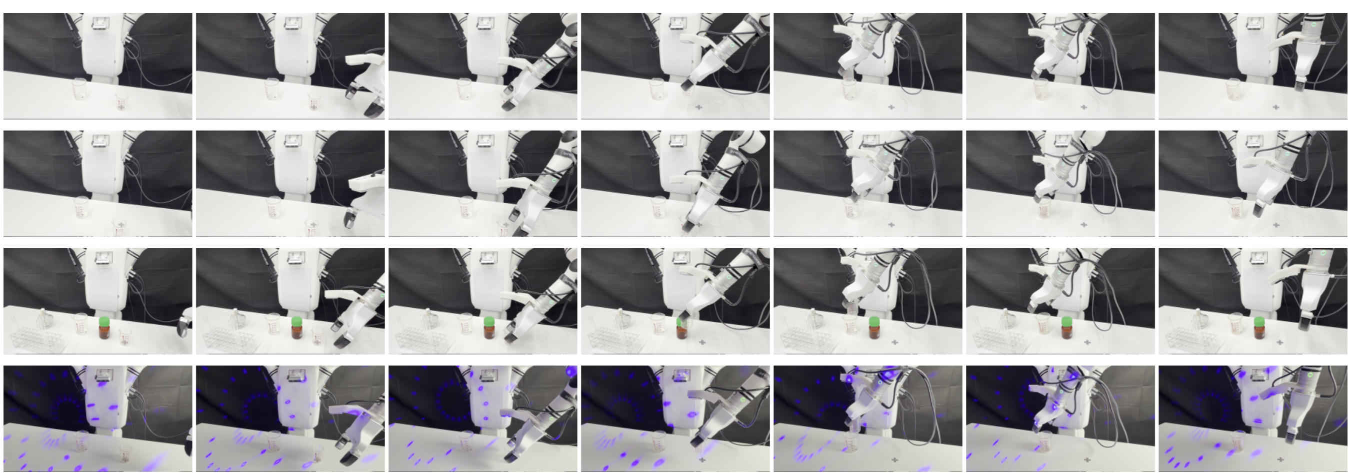


Fig. 26: **Real-robot rollout visualization (PSI-Bot, T5) under distribution shifts.** Rows: in-domain (positional) / lighting / scene (top to bottom). Columns show 7 key stages of a representative successful trajectory.

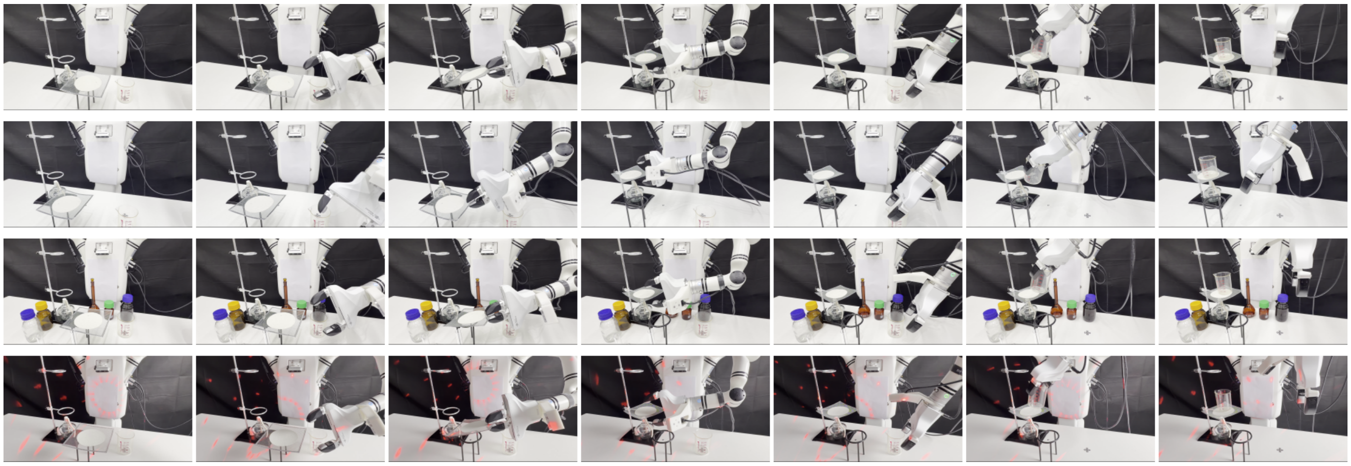
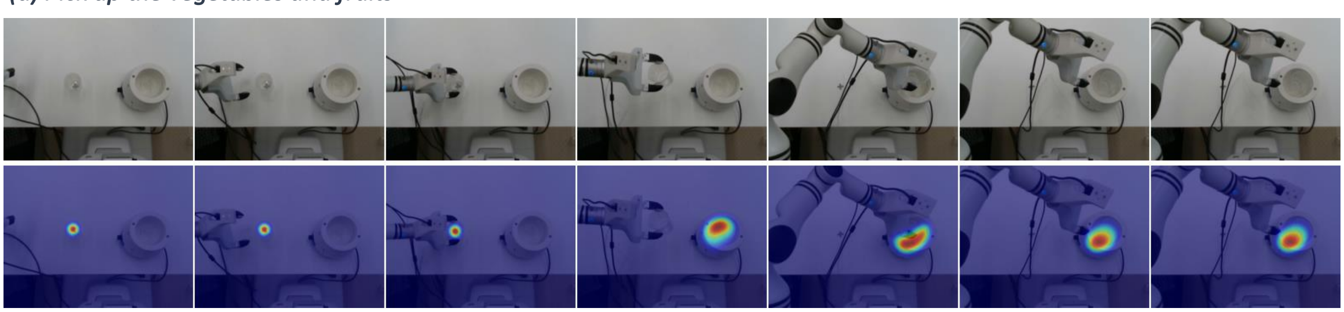


Fig. 27: **Real-robot rollout visualization (PSI-Bot, T6) under distribution shifts.** Rows: in-domain (positional) / lighting / scene (top to bottom). Columns show 7 key stages of a representative successful trajectory.

(a) Pick up the vegetables and fruits



(b) Pick up the beaker

Fig. 28: **Object-head attention on real robots (aligned tasks: T1/T4).** For each task, columns show 7 matched key stages of a representative successful rollout (left to right). Top: raw RGB observations. Bottom: normalized attention heatmaps from the **object-specialized head** overlaid on RGB (warmer colors indicate higher attention).

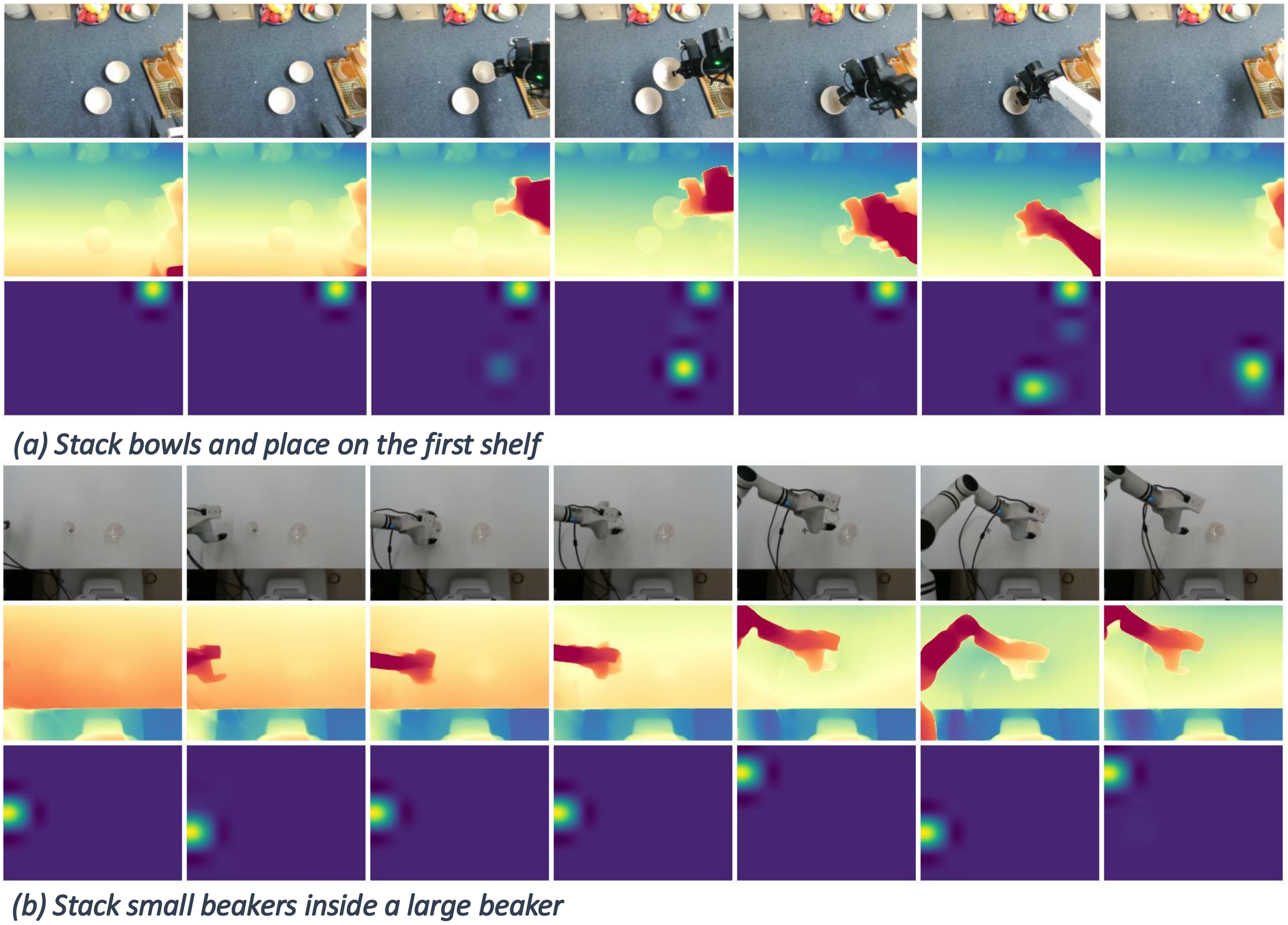


Fig. 29: **Depth/geometry-head diagnostics on real robots (aligned tasks: T2/T5).** Columns show 7 matched key stages of a representative successful rollout (left to right). Top: RGB observations. Middle: depth predictions (Depth Anything V3, small variant). Bottom: normalized attention heatmaps from the **depth/geometry-specialized head** (warmer colors indicate higher attention).

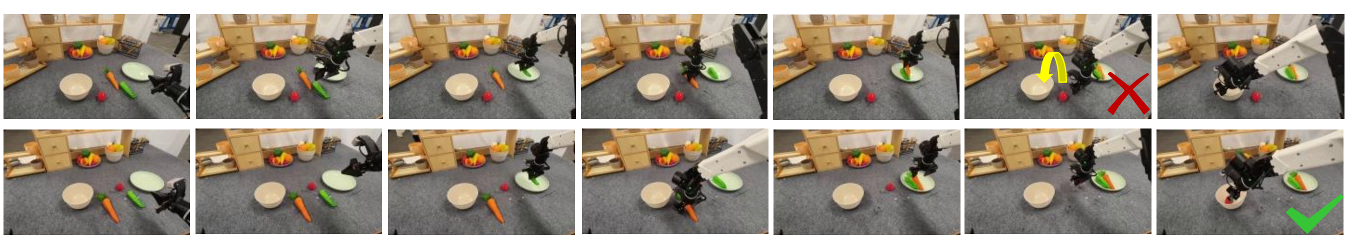


Fig. 30: **Skill/temporal diagnostics on a multi-stage real-robot task.** Columns show key stages of the tabletop-cleaning sequence. Top: $0\pi_0 exhibitsincorrecttemporalprogression(e.g., prematureterminationormissingrequiredsub - steps; markedwithredx).Bottom: GuidedVLAcompletestherequiredsub{-}taskorder, consistentwithskill/temporalsupervision.$

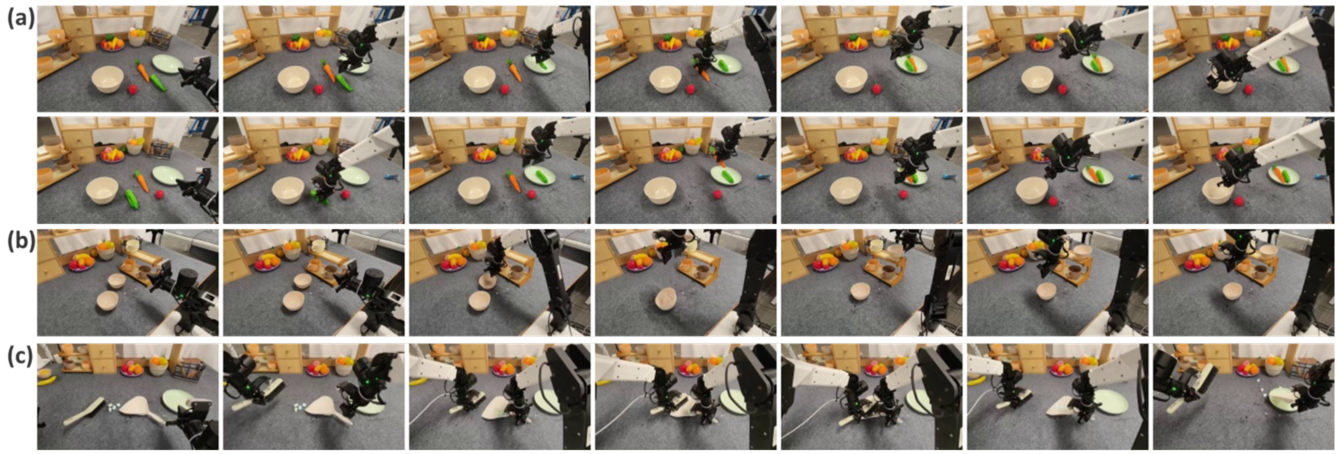


Fig. 31: **Representative failure cases of baseline π_0 on household manipulation tasks (T1–T3, ALOHA).** (a) T1: *phantom grasp* (top) and *grasp offset/slip* (bottom) when grasping the small strawberry. (b) T2: *half-grasp* on nested bowls due to insufficient insertion depth, failing to lift both bowls together. (c) T3: *stage-skipping*—pouring succeeds but the required tool-return stage is omitted. Examples are under in-domain conditions with nominal object placement.

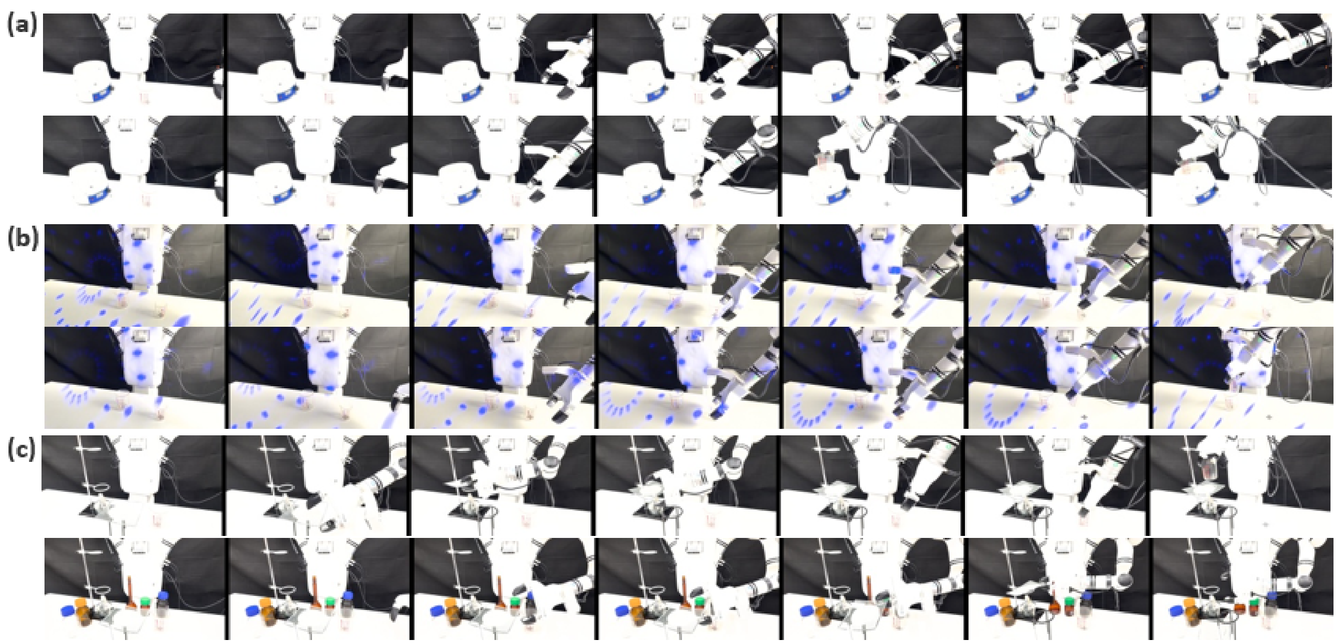


Fig. 32: **Representative failure cases of baseline π_0 on chemical-lab manipulation tasks (T4–T6, PSI-Bot).** (a) T4: transparent beaker induces *phantom grasp* (top) and *rim collision* during mantle insertion from clearance misestimation (bottom). (b) T5: *miss-grasp* under lighting/specular highlights (top) and *beaker–beaker collision* during nesting under clutter (bottom). (c) T6: collision with the ring structure from geometry mislocalization (top) and premature release before stabilization causing gauze roll-off (bottom). Lab conditions amplify grounding/geometry/temporal weaknesses due to transparent materials and millimeter-level tolerances.