

RAIN: Robotic Task Generalization with Region-Aware Interaction Networks

Anonymous Author(s)

Affiliation

Address

email

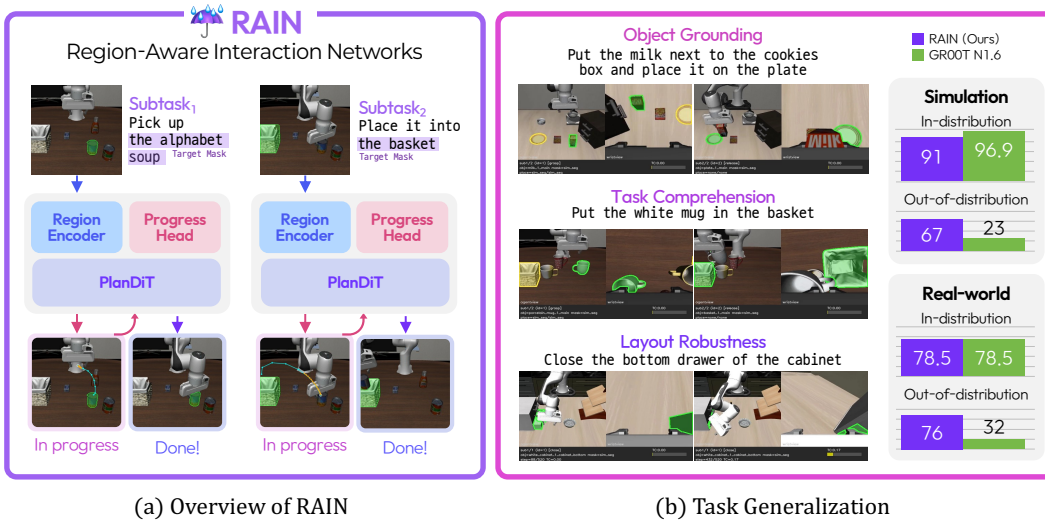


Figure 1: (a) We present **Region-Aware Interaction Networks (RAIN)** that decouples task reasoning from region-aware control by executing each subtask from an action type and target region. (b) Across object grounding, task comprehension, and layout robustness, RAIN shows stronger ID-to-OOD generalization than GR00T N1.6 [1] in simulation and real-world experiments.

1 **Abstract:** Existing vision-language-action (VLA) models benefit from broad
2 vision-language priors, but task-specific post-training often couples task reasoning
3 ing, visual grounding, and control within a single action policy. This coupling en-
4 courages policies to memorize execution patterns observed during post-training,
5 causing failures when the task objective remains unchanged but object locations,
6 subtask compositions, or scene layouts change. To address this problem, we pro-
7 pose **Region-Aware Interaction Networks (RAIN)**, an action network that sepa-
8 rates task interpretation from region-conditioned execution. Instead of directly
9 mapping a full task instruction and scene to actions, RAIN executes a specified
10 subtask action type on explicit target regions produced by external reasoning and
11 segmentation modules. It conditions multi-view visual features on the target re-
12 gion, exchanges target-aware context across views, predicts target-directed trajec-
13 tories and actions, and estimates subtask completion to enable automatic subtask
14 transitions. We also introduce a **Reference-based Retargeting Strategy (RRS)**,
15 which synthesizes trajectories for unseen target configurations by smoothly con-
16 necting arbitrary initial states to successful reference interactions. Experiments on
17 **LIBERO-EX**, our new LIBERO-derived benchmark for evaluating generalization
18 beyond training task settings, and real-world experiments demonstrate that RAIN
19 generalizes more robustly than existing VLA baselines.

20 1 Introduction

21 Robotic manipulation is a central capability for embodied intelligence, with applications spanning
22 industrial automation, household assistance, logistics, and general-purpose service robots. Recent
23 advances in Vision-Language-Models (VLMs) and embodied multimodal models [2, 3, 4, 5, 6],
24 large-scale robot datasets [7, 8, 9], and Vision-Language-Action (VLA) architectures [10, 11, 12,
25 13, 14, 1] have substantially expanded the range of language-conditioned tasks that robot policies
26 can execute from visual observations. By transferring semantic priors from large multimodal pre-
27 training and visuomotor priors from large robot action corpora, these systems have made general
28 robot policies an increasingly active research direction.

29 The dominant path toward such policies is to couple pretrained visual-language knowledge with
30 action learning, then post-train the resulting system on the target robot tasks [11, 12, 15, 16, 14,
31 17, 18, 19, 20]. While this strategy can yield strong performance on post-training tasks, it also
32 creates a critical failure mode: pretrained VLM representations, task semantics, visual grounding,
33 and action generation become optimized around a narrow set of demonstrations. Consequently, a
34 policy that appears competent on the trained tasks can fail under semantics-preserving changes,
35 such as object relocation, subtask recomposition, or scene-layout shifts. This brittleness suggests
36 task memorization rather than task understanding, and recent robustness studies show that high in-
37 distribution VLA performance can collapse under perturbations that preserve task meaning [21, 22].
38 A robot that truly understands a task should satisfy: (1) *object grounding*, executing the intended
39 interaction even when the referred object moves or a semantically similar unseen object appears; (2)
40 *task comprehension*, executing constituent subtasks of a learned long-horizon task and composing
41 independently learned subtasks into expanded instructions; and (3) *layout robustness*, maintaining
42 the learned interaction under changed scene arrangements. As illustrated in Fig. 1(a), existing post-
43 trained VLAs often degrade sharply when evaluation departs from the training scenario along these
44 axes.

45 We propose **Region-Aware Interaction Network (RAIN)**, a decoupled framework for robotic task
46 generalization. Given a task description and the current image, a general-purpose VLM indepen-
47 dently decomposes the instruction into ordered subtasks and identifies the target object and initial
48 target location for each subtask; A promptable segmentation model such as SAM 2 [23] then con-
49 verts these decisions into explicit target regions. RAIN learns how to interact with the specified re-
50 gion by combining Target-aware Layer Normalization (TarLN), which conditions each view’s visual
51 feature on the target region, a Cross-view Transformer Encoder (CTE), which exchanges informa-
52 tion across views and strengthens semantic alignment, and PlanDiT, which predicts a trajectory and
53 action chunk toward the target. A Progress Head further judges at each timestep whether the current
54 subtask has been completed, enabling automatic transition to the next subtask. We also introduce
55 a **Reference-based Retargeting Strategy (RRS)** that synthesizes novel trajectories from arbitrary
56 initial states and smoothly merges them into successful reference interactions. This allows RAIN to
57 learn region-conditioned interactions at target configurations not explicitly covered by demonstra-
58 tions. By decoupling task reasoning from action post-training, RAIN preserves the general reasoning
59 capability of pretrained VLMs, while improving execution beyond the narrow post-training distri-
60 bution.

61 We evaluate RAIN in both simulation and real-world manipulation. In simulation, we train on 40
62 tasks from LIBERO and construct **LIBERO-EX**, a benchmark designed to evaluate task general-
63 ization along object grounding, task comprehension, and layout robustness. Against strong VLA
64 baselines, including OpenVLA-OFT, π_0 , $\pi_{0.5}$, and GR00T, RAIN shows stronger generalization on
65 LIBERO-EX while maintaining competitive performance on the original LIBERO suites. We also
66 validate the same design in real-world tabletop manipulation, where RAIN generalizes to changed
67 objects, target locations, subtask combinations, and scene layouts. Our contributions are threefold:

- 68 - We propose **RAIN**, a region-aware action model for robotic task generalization that decouples
69 VLM-based task reasoning from post-trained visuomotor control.

- 70 - We introduce **RRS**, a reference-based retargeting strategy that synthesizes executable trajectories
71 toward arbitrary target configurations by adapting successful reference interactions.
- 72 - We present **LIBERO-EX** and real-world evaluations that test object grounding, task comprehen-
73 sion, and layout robustness, demonstrating that RAIN generalizes more robustly than existing
74 VLA baselines.

75 2 Related Works

76 **Vision-Language-Action Models.** Vision-language-action (VLA) models [10, 11, 13, 12] have
77 become a major paradigm for language-conditioned robot manipulation by transferring vision-
78 language priors into visuomotor policies. Early systems connected language-conditioned percep-
79 tion with action prediction for real-world control, while open-source generalist policies expanded
80 the setting to broader robot embodiments and task suites. Large robot datasets [7, 8, 9], flow-based
81 policies and generalist robot policies [14, 1], and hierarchical or reasoning-augmented architectures
82 [16, 24, 25, 26, 27, 19] have further extended this paradigm toward longer-horizon and more di-
83 verse manipulation tasks. Recent robustness benchmarks such as LIBERO-PRO [21] and LIBERO-
84 Plus [22] show that high in-distribution VLA success can hide memorization, evaluating perturba-
85 tions to objects, initial states, instructions, layouts, viewpoints, and visual conditions, but largely
86 preserve the original LIBERO task semantics and ask whether the same task remains solvable in a
87 modified but related setup. LIBERO-EX instead targets task-level generalization by evaluating ob-
88 ject grounding, task-scope expansion and reduction, and layout robustness on tasks derived from but
89 not identical to the learned LIBERO tasks.

90 **Visual Prompting for Robot Policies.** Visual and multimodal prompts provide explicit spatial or
91 procedural information for robot policies. VIMA [28] formulates robot manipulation with multi-
92 modal prompts that interleave language and visual tokens. Subsequent work has explored coordinate
93 or sketch-based visual instructions [29, 30], trajectory and trace prompts [17] for spatial-temporal
94 guidance, and pixel-level prompt-native VLA policies [18] that use dense visual cues such as points,
95 lines, boxes, or masks. Visual instruction tuning [3] and pixel grounding [31] provide general vision-
96 language foundations for handling visual prompts and grounded references.

97 3 Method

98 In this section, we present Region-Aware Interaction Networks (RAIN), an action network that ex-
99 ecutes a specified subtask and estimates when the subtask has been completed. As shown in Fig. 2,
100 each subtask is provided as an action type such as *grasp*, *release*, or *open*, together with target masks
101 in the available camera views. In real-world experiments, an external VLM decomposes the task in-
102 struction into subtasks before execution and identifies the target object or region for each subtask,
103 while SAM 2 [23] obtains and tracks the corresponding masks. In simulation, action types and tar-
104 get masks are obtained from predefined task annotations. RAIN therefore does not learn the full task
105 description inside the action policy. Instead, it learns how to interact with the specified region by
106 combining target-aware visual encoding, target-directed action generation, and progress estimation.

107 3.1 Region-Aware Interaction Networks

108 Although the formulation naturally extends to more camera views, we describe the method with two
109 views for clarity. At each timestep t , RAIN observes two RGB views $\mathcal{I}_t = \{\mathbf{I}_t^T, \mathbf{I}_t^W\}$ from a third-
110 person and wrist camera, the proprioceptive state $\mathbf{q}_t \in \mathbb{R}^7$, an action type c_t , and target masks $\mathcal{M}_t =$
111 $\{\mathbf{m}_t^T, \mathbf{m}_t^W\}$. The model aims to learn $p(\mathbf{A}_t, \mathbf{P}_t \mid \mathcal{I}_t, \mathbf{q}_t, c_t, \mathcal{M}_t)$, where $\mathbf{A}_t = [\mathbf{a}_t, \dots, \mathbf{a}_{t+H-1}] \in$
112 $\mathbb{R}^{H \times 7}$ is an action chunk of horizon H and $\mathbf{P}_t = \{\mathbf{p}_{t,1}, \dots, \mathbf{p}_{t,N}\} \in \mathbb{R}^{N \times 3}$ is an auxiliary plan
113 with N waypoints. The action chunk contains 3D position, 3D rotation, and gripper commands.
114 The plan represents target-directed 3D waypoints and guides the action decoder toward the region
115 specified by the masks.

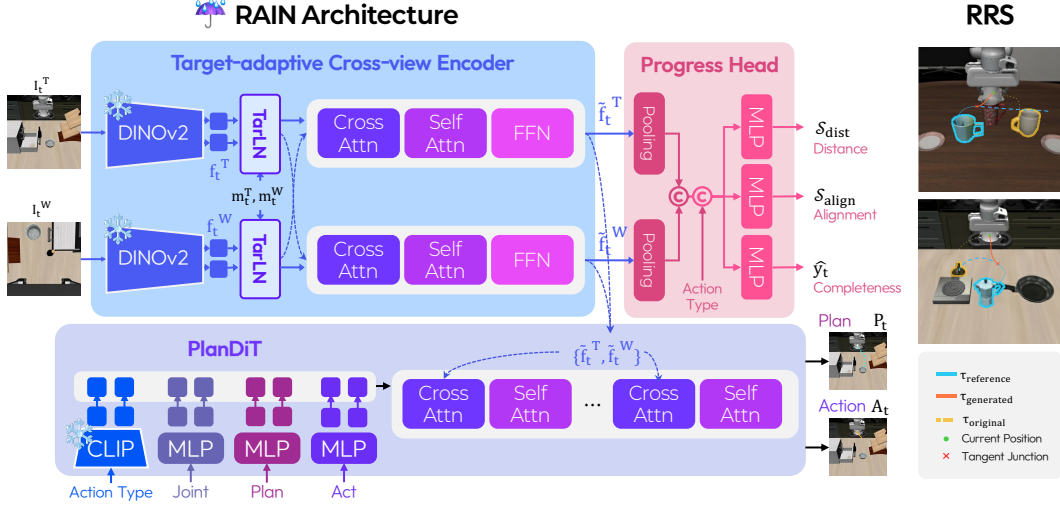


Figure 2: **Overall architecture of RAIN.** For each subtask, RAIN receives an action type and target masks. The **Target-adaptive Cross-view Encoder (TCE)** extracts target-aware features from the third-person and wrist views. The **PlanDiT** jointly predicts future waypoints and action chunks, while the **Progress Head** estimates interaction progress and subtask completion for autonomous transitions. **Reference-based Retargeting strategy (RRS)** augments training by retargeting recorded successful interactions to new target locations with smooth, executable trajectories.

116 RAIN consists of three main components. The Target-adaptive Cross-view Encoder (TCE) extracts
 117 target-aware multi-view features from images and masks, the Plan-guided Diffusion Transformer
 118 (PlanDiT) predicts the geometric plan and action chunk, and the Progress Head estimates whether
 119 the current subtask has been completed.

120 **Target-adaptive Cross-view Encoder.** The Target-adaptive Cross-view Encoder (TCE) extracts
 121 target-aware representations from both views. Given visual features $\mathbf{f}_t^T, \mathbf{f}_t^W \in \mathbb{R}^{h \times w \times d}$ from a
 122 pretrained image encoder, we first resize the target masks to the feature resolution. Target-adaptive
 123 Layer Normalization (TarLN) then injects mask information into each view:

$$\text{TarLN}(\mathbf{f}_t^v, \mathbf{m}_t^v) = \gamma^v(\mathbf{m}_t^v) \odot \text{LN}(\mathbf{f}_t^v) + \beta^v(\mathbf{m}_t^v), \quad v \in \{T, W\}. \quad (1)$$

124 Here, LN denotes layer normalization [32], and $\gamma^v(\cdot), \beta^v(\cdot) \in \mathbb{R}^{h \times w \times d}$ are pixel-wise modulation
 125 parameters produced by a spatial MLP, following the spirit of spatially-adaptive normalization [33].
 126 Since both the third-person and wrist views receive their own target masks, the target region is
 127 directly injected into each view before cross-view aggregation.

128 After TarLN, the two views exchange information through cross-view Transformer blocks. Let
 129 $\tilde{\mathbf{f}}_t^v = \text{TarLN}(\mathbf{f}_t^v, \mathbf{m}_t^v)$. Each block applies cross-attention between the two views, using the first
 130 argument as the query and the second as keys and values, followed by self-attention and a feed-
 131 forward network:

$$\tilde{\mathbf{f}}_t^T = \text{FFN}(\text{SelfAttn}(\text{CrossAttn}(\tilde{\mathbf{f}}_t^T, \tilde{\mathbf{f}}_t^W))), \quad \tilde{\mathbf{f}}_t^W = \text{FFN}(\text{SelfAttn}(\text{CrossAttn}(\tilde{\mathbf{f}}_t^W, \tilde{\mathbf{f}}_t^T))). \quad (2)$$

132 This design preserves the explicit target conditioning in each view while allowing the wrist and third-
 133 person views to share complementary geometric information. We denote the resulting target-aware
 134 visual context as $\phi_t = [\tilde{\mathbf{f}}_t^T; \tilde{\mathbf{f}}_t^W]$.

135 **Plan-guided Diffusion Transformer.** PlanDiT is a generative Transformer that predicts a target-
 136 directed geometric plan and an executable action chunk. The geometric plan \mathbf{P}_t consists of 3D way-
 137 points toward the target region, and the final waypoint $\mathbf{p}_{t,N}$ corresponds to the predicted interaction
 138 point. The action chunk \mathbf{A}_t contains the low-level commands executed by the robot controller.

139 PlanDiT forms an input token sequence $\mathbf{E}_t = [\mathbf{e}_c; \mathbf{e}_q; \mathbf{e}_p; \mathbf{e}_a]$, where \mathbf{e}_c is the action-type embed-
 140 ding, \mathbf{e}_q is the proprioception embedding, and $\mathbf{e}_p, \mathbf{e}_a$ are plan and action tokens embedded from the
 141 noisy sample used in flow matching. Transformer blocks process these tokens with cross-attention to
 142 the TCE context ϕ_t . To make the action prediction follow the geometric plan, we use an asymmetric
 143 self-attention mask: action tokens can attend to plan tokens, but plan tokens cannot attend to action
 144 tokens.

145 We train PlanDiT with Flow Matching [1]. Let $\mathbf{x}_t = [\mathbf{P}_t; \mathbf{A}_t]$ be the concatenated target trajectory.
 146 Given a flow step $\tau \in [0, 1]$ and Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we define $\mathbf{x}_t^\tau = \tau \mathbf{x}_t + (1 - \tau)\epsilon$ and
 147 optimize the vector field V_θ by

$$\mathcal{L}_{\text{fm}} = \mathbb{E}_{\tau, \epsilon, \mathbf{x}_t} \left[\|V_\theta(\mathbf{x}_t^\tau, \phi_t, \mathbf{q}_t, c_t) - (\mathbf{x}_t - \epsilon)\|_2^2 \right]. \quad (3)$$

148 We further encourage stable target grounding by enforcing consistency of the final waypoint across
 149 frames from the same subtask. Given K sampled frames and the frame t_K closest to completion, we
 150 minimize the consistency loss

$$\mathcal{L}_{\text{cons}} = \sum_{k=1}^{K-1} \text{SmoothL1}(\hat{\mathbf{p}}_{t_k, N} - \hat{\mathbf{p}}_{t_K, N}). \quad (4)$$

151 The total objective for PlanDiT is $\mathcal{L}_{\text{PlanDiT}} = \lambda_{\text{fm}} \mathcal{L}_{\text{fm}} + \lambda_{\text{cons}} \mathcal{L}_{\text{cons}}$, where λ_{fm} and λ_{cons} are loss
 152 weights.

153 **Progress Estimation.** To autonomously execute multiple subtasks, RAIN uses a Progress Head
 154 whose primary role is to decide whether the current subtask has been completed. To support this
 155 decision, the head also estimates how far the end-effector remains from the target object and how
 156 well the gripper is aligned with the target interaction pose. The head takes the pooled TCE features
 157 and the action-type embedding as input and predicts a completion probability \hat{y}_t together with a
 158 distance score $\hat{\mathcal{S}}_{\text{dist}}$ and an alignment score $\hat{\mathcal{S}}_{\text{align}}$. The target distance and alignment scores are
 159 defined as

$$\mathcal{S}_{\text{dist}} = \exp(-\alpha \|\mathbf{p}_t^{\text{ee}} - \mathbf{p}^g\|_2), \quad \mathcal{S}_{\text{align}} = \frac{1}{2} \left(1 + \frac{\text{tr}((\mathbf{R}_t^{\text{ee}})^\top \mathbf{R}^g) - 1}{2} \right), \quad (5)$$

160 where $\alpha > 0$ is a scaling constant, $(\mathbf{p}_t^{\text{ee}}, \mathbf{R}_t^{\text{ee}})$ is the current end-effector pose, and $(\mathbf{p}^g, \mathbf{R}^g)$ is
 161 the target interaction pose from the demonstration. The completion label y_t is obtained from the
 162 annotated subtask boundary. The Progress Head is trained with

$$\mathcal{L}_{\text{head}} = \lambda_{\text{dist}} \mathcal{L}_{\text{dist}} + \lambda_{\text{align}} \mathcal{L}_{\text{align}} + \lambda_{\text{comp}} \mathcal{L}_{\text{comp}}, \quad (6)$$

163 where $\mathcal{L}_{\text{dist}}$ and $\mathcal{L}_{\text{align}}$ are Smooth L1 losses for the auxiliary progress scores, $\mathcal{L}_{\text{comp}}$ is binary cross
 164 entropy between \hat{y}_t and y_t , and $\lambda_{\text{dist}}, \lambda_{\text{align}}, \lambda_{\text{comp}}$ are loss weights. During rollout, RAIN moves to
 165 the next subtask when \hat{y}_t exceeds a completion threshold.

166 3.2 Reference-based Retargeting Strategy

167 RRS expands region-aware interaction coverage by generating trajectories toward target configura-
 168 tions that are not directly covered by the original demonstrations. Given a recorded reference
 169 trajectory $\mathcal{R} = \{\mathbf{r}_0, \dots, \mathbf{r}_{M-1}\}$ and an arbitrary approach state $\mathbf{x}_0 \in \mathbb{R}^3$ denoting an end-effector
 170 position, RRS first identifies a tangent junction \mathbf{r}_{k^*} on the reference trajectory. For each reference
 171 point $k \in \{1, \dots, M-2\}$, we compute the unit tangent vector $\mathbf{t}_k = (\mathbf{r}_{k+1} - \mathbf{r}_{k-1}) / \|\mathbf{r}_{k+1} - \mathbf{r}_{k-1}\|_2$,
 172 the approach direction $\mathbf{v}_k = (\mathbf{r}_k - \mathbf{x}_0) / \|\mathbf{r}_k - \mathbf{x}_0\|_2$, and the distance $d_k = \|\mathbf{r}_k - \mathbf{x}_0\|_2$. The junction
 173 score is

$$S_k = (\mathbf{v}_k^\top \mathbf{t}_k) \left(1 - \frac{d_k}{d_{\text{max}}} \right) \left(1 - \frac{k}{M} \right), \quad (7)$$

174 and the junction index is selected as $k^* = \arg \max_k S_k$ under proximity and directional-alignment
 175 constraints, where d_{max} is a normalization radius determined by the workspace scale. The first term
 176 favors tangential merging into the reference motion, the second avoids long approach paths, and the
 177 third preserves as much of the successful reference interaction as possible.

Table 1: **Simulation Results on LIBERO and LIBERO-EX.** We report success rates on the original LIBERO suites and our LIBERO-EX task-generalization benchmark. † denotes our reproduction using a single joint 4-suite fine-tuned model trained on all LIBERO suites.

Method	Size	Trainable Params		LIBERO-EX				LIBERO				
		Pre-train	Post-train	Object	Layout	Task	Avg.	Spatial	Object	Goal	Long	Avg.
OpenVLA-OFT [15] [RSS'25]	7B	7B	0.28B	1.6	0.0	0.0	0.5	97.6	98.4	97.9	94.5	97.1
π_0 [14] [RSS'25]	3B	3B	3B	38.8	37.2	6.0	27.3	96.8	98.8	95.8	85.2	94.2
π_0 – FAST [35] [RSS'25]	3B	3B	3B	0.4	12.8	7.2	6.8	96.4	96.8	88.6	60.2	85.5
$\pi_{0.5}$ [16] [CoRL'25]	3B	3B	3B	64.4	57.2	2.4	41.3	98.8	98.2	98.0	92.4	96.9
GR00T N1.6† [1] [arXiv'25]	3B	3B	3B	35.2	31.2	2.8	23.0	98.4	99.8	98.0	90.0	95.8
Action-Sketcher [18] [CVPR'26]	3B	3B	3B	47.6	27.6	3.2	26.1	97.2	99.6	94.8	94.8	96.9
Cosmos Policy [36] [ICLR'26]	2B	-	2B	40.8	48.4	5.6	31.6	98.1	100.0	98.2	97.6	98.5
SmolVLA [37] [arXiv'25]	0.45B	0.1B	0.1B	2.0	6.0	2.0	3.3	90.0	96.0	92.0	71.0	87.3
Dita† [38] [ICCV'25]	0.33B	0.22B	0.22B	29.6	26.0	9.6	21.7	82.0	83.5	80.0	68.5	78.5
RAIN-L (Ours)	0.62B	-	0.19B	76.0	72.0	53.2	67.0	86.8	96.0	96.0	85.2	91.0
RAIN-S (Ours)	0.25B	-	0.17B	53.6	54.0	50.4	52.7	87.8	91.4	89.8	78.4	86.9

Table 2: **Ablation studies on each component of our proposed method.**

(a) CTE		(b) PlanDiT		(c) Progress head			(d) Retarget strategy	
Method	LIBERO	Method	LIBERO	$\mathcal{L}_{\text{dist}}$	$\mathcal{L}_{\text{align}}$	LIBERO	Method	LIBERO-EX
Full model	91.0	Full model	91.0			89.7	no retarget	44.3
w/o cross-view	82.1	w/o plan token	89.7	✓		90.5	linear interp.	58.8
w/o third mask	86.6	w/o $\mathcal{L}_{\text{consis}}$	90.4		✓	89.9	RRS (ours)	67.0
w/o wrist mask	85.8	w/ bi-directional attn	89.0	✓	✓	91.0		

178 Once the junction is chosen, RRS connects \mathbf{x}_0 to \mathbf{r}_{k^*} using a cubic Hermite spline [34]. The variable
 179 $u \in [0, 1]$ denotes the normalized interpolation parameter:

$$\mathbf{C}(u) = \psi_{00}(u)\mathbf{x}_0 + \psi_{10}(u)\mathbf{D}_0 + \psi_{01}(u)\mathbf{r}_{k^*} + \psi_{11}(u)\mathbf{D}_1, \quad u \in [0, 1], \quad (8)$$

180 where $\mathbf{D}_0 = \mathbf{r}_{k^*} - \mathbf{x}_0$, $\mathbf{D}_1 = L\mathbf{t}_{k^*}$, and $L = \|\mathbf{r}_{k^*} - \mathbf{x}_0\|_2$. The basis functions are defined as
 181 follows:

$$\psi_{00} = 2u^3 - 3u^2 + 1, \quad \psi_{10} = u^3 - 2u^2 + u, \quad \psi_{01} = -2u^3 + 3u^2, \quad \psi_{11} = u^3 - u^2. \quad (9)$$

182 The approach segment is discretized with N_{app} steps using $u_i = i/N_{\text{app}}$ for $i = 0, \dots, N_{\text{app}}$. The
 183 final trajectory is obtained by concatenating the generated approach and the remaining reference
 184 interaction:

$$\mathcal{P} = [\mathbf{C}(u_0), \dots, \mathbf{C}(u_{N_{\text{app}}}), \mathbf{r}_{k^*+1}, \dots, \mathbf{r}_{M-1}]. \quad (10)$$

185 We convert the approach segment into delta-position actions and interpolate orientation toward the
 186 junction pose. After the junction, the recorded reference actions are reused, preserving contact-rich
 187 interaction dynamics while exposing RAIN to diverse target locations.

188 4 Experiments

189 We evaluate RAIN’s task generalization in both simulation and real-world manipulation. In simu-
 190 lation, we compare RAIN with recent state-of-the-art VLA policies on the original LIBERO suites
 191 and on LIBERO-EX, a benchmark designed to test generalization beyond the training task settings.
 192 In the real world, we evaluate whether the same region-aware action network remains robust under
 193 out-of-distribution object, target, and layout changes. Implementation details are provided in
 194 Appendix A, with additional simulation and real-world protocols in Appendix B and Appendix C.

195 4.1 Simulation Experiments

196 **Setting.** We train a single policy on the 40 tasks from the LIBERO Spatial, Object, Goal, and Long
 197 suites, and compare it with recent VLA models under the same joint-suite evaluation protocol. To

198 evaluate generalization beyond the training distribution, we introduce LIBERO-EX with three cat-
199 egories: object grounding, task comprehension, and layout robustness. Each category contains 25
200 tasks with 10 evaluation episodes per task. Detailed task definitions and selected initial states are
201 provided in Appendix B and Tables 8–12.

202 **Main results.** Tab. 1 compares RAIN with recent VLA models on both LIBERO and LIBERO-EX.
203 Large pretrained baselines often achieve high in-distribution success on LIBERO after post-training,
204 with several models exceeding 95 average success. However, their performance does not carry over
205 to LIBERO-EX. When the target object, task scope, or scene layout changes, existing VLA poli-
206 cies frequently fail to execute the intended task. The gap is most severe on task comprehension,
207 where all compared baselines remain below 10 success. This split requires policies to execute a
208 constituent subtask of a learned long-horizon task, or to compose learned single-step skills into
209 a new long-horizon task, directly testing whether the policy understands the task structure rather
210 than memorizing a demonstrated sequence. RAIN-L achieves the strongest LIBERO-EX average.
211 RAIN-S, despite using a DINOv2-Small vision encoder and only 0.25B parameters, also surpasses
212 all compared VLA baselines on LIBERO-EX. A per-task breakdown of LIBERO-EX is provided in
213 Tables 5–7.

214 **Ablation study.** Tab. 2 analyzes the main components of RAIN. For the Target-adaptive Cross-
215 view Encoder, removing cross-view exchange reduces LIBERO success from 91.0 to 82.1, showing
216 that information exchange around the target object is critical for aligning the third-person and wrist
217 views. Removing either target mask also hurts performance, since the model loses a view-specific
218 visual prompt and becomes more vulnerable to occlusion or viewpoint ambiguity. For PlanDiT, plan
219 tokens improve performance by 1.3 points by predicting a 3D path from the current state toward the
220 target interaction point. The consistency loss $\mathcal{L}_{\text{cons}}$ provides a smaller but consistent gain by aligning
221 the predicted goal of frames from the same subtask. Notably, bi-directional attention between plan
222 and action tokens decreases performance from 91.0 to 89.0, while our asymmetric design lets the
223 plan guide the action without being distorted by action-token noise. The Progress Head benefits from
224 both auxiliary distance and alignment supervision, which help the completion classifier decide when
225 the current subtask is finished. Finally, even without retargeting, RAIN reaches 44.3 on LIBERO-
226 EX, already 3.0 points higher than $\pi_{0.5}$. Linear interpolation from the current gripper state to the
227 goal raises performance to 58.8, while RRS further improves it to 67.0 by generating smoother
228 target-directed interactions.

229 4.2 Real-World Experiments

230 **Setting.** Our real-world platform uses a 6-DoF ROBOTIS OMY-F3M single-arm manipulator with
231 a two-finger gripper. Policies operate through a 7D joint-space action interface and observe robot
232 joint states with RGB images from wrist-mounted, third-person, and top-down cameras. Demonstra-
233 tions are recorded with a ROS-based system and converted to LeRobot [39]. We evaluate two task
234 families, Potato-to-Plate and Tape-to-Clay. All policies are trained on the same demonstra-
235 tions and evaluated under matched robot, camera, action-space, initial-state, and success-criterion
236 settings. We compare with GROOT N1.6 [1], X-VLA [40], and MolmoAct2 [41], using their offi-
237 cial codebases with only the interface wrappers required for our robot setup. Detailed setup, split
238 definitions, and baseline training protocols are provided in Appendix C.1–C.4.

239 **Results.** As shown in Fig. 3, RAIN maintains stronger out-of-distribution performance across real-
240 world task families. In Potato-to-Plate, all methods are trained to place a potato on the middle
241 plate. GROOT N1.6 and MolmoAct2 remain effective when the middle plate changes from round to
242 square, but their success drops sharply when the target plate moves away from the middle position,
243 by 60 and 30 points respectively. RAIN shows little degradation under the same target-location shift,
244 indicating that it follows the specified target region rather than a memorized placement trajectory.
245 In Tape-to-Clay, RAIN also achieves the best OOD average, showing robustness to reordered
246 source objects and changed targets. These results support the simulation findings: separating task
247 interpretation from region-conditioned interaction improves generalization when the task semantics
248 are preserved but the visual and spatial context changes.



Figure 3: **Real-World Generalization Results.** OOD success rates on (a) Potato-to-Plate and (b) Tape-to-Clay task suites, with (c) a qualitative zero-shot rollout of RAIN successfully executing the unseen Lemon-to-Plate task.

249 5 Conclusion

250 We presented **Region-Aware Interaction Networks (RAIN)**, an action network for robotic task
 251 generalization that separates task-level interpretation from region-conditioned execution. Instead of
 252 absorbing reasoning, grounding, and control into a single post-trained VLA policy, RAIN receives
 253 a subtask action type and an explicit target region from external vision-language reasoning and
 254 segmentation models, then learns how to execute the specified interaction. Its target-conditioned
 255 multi-view representation, target-directed trajectory and action prediction, and progress estimation
 256 enable autonomous subtask transitions while keeping the action model focused on region-grounded
 257 execution. We further introduced **Reference-based Retargeting Strategy (RRS)**, which connects
 258 arbitrary target-directed approaches to reference trajectories through tangent junctions, expanding
 259 the diversity of region-aware interactions available during training. Experiments on LIBERO-EX
 260 and in the real world show that this design improves generalization beyond the post-training task
 261 distribution, especially under changes in object placement, task scope, and scene layout.

262 6 Limitations

263 RAIN improves task generalization by conditioning action generation on explicit target regions, but
 264 this design also makes the policy dependent on reliable visual prompts. When the relevant target
 265 region is invisible from all available views, for example due to severe occlusion, poor camera place-
 266 ment, or a target that only becomes visible after substantial scene rearrangement, RAIN may fail
 267 because the action model no longer receives a grounded region to interact with. The model can
 268 also still fail under extreme out-of-distribution shifts that require interaction geometry or object dy-
 269 namics far beyond those covered by demonstrations and RRS-generated trajectories. Nevertheless,
 270 our results show that separating task interpretation from region-conditioned interaction substantially
 271 improves generalization over existing VLA baselines across object grounding, task comprehension,
 272 layout robustness, and real-world OOD settings.

References

- 273
- 274 [1] NVIDIA, J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox,
275 F. Hu, S. Huang, et al. GR00T N1: An open foundation model for generalist humanoid robots.
276 *arXiv preprint arXiv:2503.14734*, 2025.
- 277 [2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell,
278 P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervi-
279 sion. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139
280 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021.
- 281 [3] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In A. Oh, T. Naumann,
282 A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information*
283 *Processing Systems*, volume 36, pages 34892–34916. Curran Associates, Inc., 2023. URL
284 <https://proceedings.neurips.cc>.
- 285 [4] S. Karamcheti, S. Nair, A. Balakrishna, P. Liang, T. Kollar, and D. Sadigh. Prismatic VLMs:
286 Investigating the design space of visually-conditioned language models. In *Proceedings of the*
287 *41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine*
288 *Learning Research*, pages 23123–23144. PMLR, 2024.
- 289 [5] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdul-
290 mohsin, M. Tschannen, E. Bugliarello, et al. PaliGemma: A versatile 3b VLM for transfer.
291 *arXiv preprint arXiv:2407.07726*, 2024.
- 292 [6] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson,
293 Q. Vuong, T. Yu, et al. PaLM-E: An embodied multimodal language model. In *Proceedings*
294 *of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of*
295 *Machine Learning Research*, pages 8469–8488. PMLR, 2023.
- 296 [7] A. O’Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta,
297 A. Mandlekar, A. Jain, et al. Open X-Embodiment: Robotic learning datasets and RT-X mod-
298 els. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–
299 6903. IEEE, 2024.
- 300 [8] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. My-
301 ers, M. J. Kim, M. Du, A. Lee, K. Fang, C. Finn, and S. Levine. Bridgedata v2: A dataset for
302 robot learning at scale. In *Proceedings of The 7th Conference on Robot Learning*, volume 229
303 of *Proceedings of Machine Learning Research*, pages 1723–1736. PMLR, 2023.
- 304 [9] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany,
305 M. K. Srirama, L. Y. Chen, K. Ellis, et al. DROID: A large-scale in-the-wild robot manip-
306 ulation dataset. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July
307 2024. doi:10.15607/RSS.2024.XX.120. URL [https://www.roboticsproceedings.org/](https://www.roboticsproceedings.org/rss20/p120.html)
308 [rss20/p120.html](https://www.roboticsproceedings.org/rss20/p120.html).
- 309 [10] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Haus-
310 man, A. Herzog, J. Hsu, et al. RT-1: Robotics transformer for real-world control at scale. In
311 *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi:
312 10.15607/RSS.2023.XIX.025. URL [https://www.roboticsproceedings.org/rss19/](https://www.roboticsproceedings.org/rss19/p025.html)
313 [p025.html](https://www.roboticsproceedings.org/rss19/p025.html).
- 314 [11] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid,
315 et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In
316 *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine*
317 *Learning Research*, pages 2165–2183. PMLR, 2023.

- 318 [12] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster,
319 G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine,
320 P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. In *Conference*
321 *on Robot Learning*, 2024. URL <https://openreview.net/forum?id=ZMnD6QZAE6>.
- 322 [13] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna,
323 C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh,
324 C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proceedings of*
325 *Robotics: Science and Systems*, Delft, Netherlands, 2024.
- 326 [14] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman,
327 B. Ichter, et al. π_0 : A vision-language-action flow model for general robot control. In *Proceed-*
328 *ings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi:10.15607/RSS.
329 2025.XXI.010. URL <https://www.roboticsproceedings.org/rss21/p010.html>.
- 330 [15] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing
331 speed and success. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA,
332 June 2025. doi:10.15607/RSS.2025.XXI.017.
- 333 [16] Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail,
334 M. Equi, C. Finn, N. Fusai, et al. $\pi_0.5$: a vision-language-action model with open-world
335 generalization. *arXiv preprint arXiv:2504.16054*, 2025.
- 336 [17] R. Zheng, Y. Liang, S. Huang, J. Gao, H. Daumé III, A. Kolobov, F. Huang, and J. Yang.
337 Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic
338 policies. *arXiv preprint arXiv:2412.10345*, 2024.
- 339 [18] W. Liang, G. Sun, Y. He, J. Dong, S. Dai, I. Laptev, S. Khan, and Y. Cong. Pixelvla: Ad-
340 vancing pixel-level understanding in vision-language-action model. In *International Con-*
341 *ference on Learning Representations*, 2026. URL [https://openreview.net/forum?id=](https://openreview.net/forum?id=7M6ryCAB1c)
342 [7M6ryCAB1c](https://openreview.net/forum?id=7M6ryCAB1c).
- 343 [19] C.-P. Huang, Y.-H. Wu, M.-H. Chen, Y.-C. F. Wang, and F.-E. Yang. Thinkact:
344 Vision-language-action reasoning via reinforced visual latent planning. *arXiv preprint*
345 *arXiv:2507.16815*, 2025.
- 346 [20] G.-C. Kang, J. Kim, K. Shim, J. K. Lee, and B.-T. Zhang. Clip-rt: Learning
347 language-conditioned robotic policies from natural language supervision. *arXiv preprint*
348 *arXiv:2411.00508*, 2024.
- 349 [21] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu, P. Zhou, and L. Sun. Libero-pro: To-
350 wards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv*
351 *preprint arXiv:2510.03827*, 2025.
- 352 [22] S. Fei, S. Wang, J. Shi, Z. Dai, J. Cai, P. Qian, L. Ji, X. He, S. Zhang, Z. Fei, et al.
353 Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint*
354 *arXiv:2510.13626*, 2025.
- 355 [23] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland,
356 L. Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint*
357 *arXiv:2408.00714*, 2024.
- 358 [24] L. X. Shi, B. Ichter, M. Equi, L. Ke, K. Pertsch, Q. Vuong, J. Tanner, A. Walling, H. Wang,
359 N. Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-
360 action models. *arXiv preprint arXiv:2502.19417*, 2025.
- 361 [25] Y. Li, Y. Deng, J. Zhang, J. Jang, M. Memmel, R. Yu, C. R. Garrett, F. Ramos, D. Fox, A. Li,
362 et al. Hamster: Hierarchical action models for open-world robot manipulation. *arXiv preprint*
363 *arXiv:2502.05485*, 2025.

- 364 [26] D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, Z. Wang, J. Gu, B. Zhao, D. Wang, et al.
365 Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint*
366 *arXiv:2501.15830*, 2025.
- 367 [27] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via
368 embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.
- 369 [28] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu,
370 and L. Fan. Vima: General robot manipulation with multimodal prompts. In *Fortieth Interna-*
371 *tional Conference on Machine Learning*, 2023.
- 372 [29] X. Li, C. Mata, J. Park, K. Kahatapitiya, Y. S. Jang, J. Shang, K. Ranasinghe, R. Burgert,
373 M. Cai, Y. J. Lee, et al. Llara: Supercharging robot learning data for vision-language policy.
374 *arXiv preprint arXiv:2406.20095*, 2024.
- 375 [30] Y. Li, Z. Gong, H. Li, X. Huang, H. Kang, G. Bai, and X. Ma. Robotic visual instruction. In
376 *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 12155–12165,
377 2025.
- 378 [31] H. Rasheed, M. Maaz, S. Shaji, A. Shaker, S. Khan, H. Cholakkal, R. M. Anwer, E. Xing, M.-
379 H. Yang, and F. S. Khan. Glamm: Pixel grounding large multimodal model. *The IEEE/CVF*
380 *Conference on Computer Vision and Pattern Recognition*, 2024.
- 381 [32] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*,
382 2016.
- 383 [33] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-
384 adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and*
385 *pattern recognition*, pages 2337–2346, 2019.
- 386 [34] D. H. Kochanek and R. H. Bartels. Interpolating splines with local tension, continuity, and bias
387 control. In *Proceedings of the 11th annual conference on Computer graphics and interactive*
388 *techniques*, pages 33–41, 1984.
- 389 [35] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and
390 S. Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint*
391 *arXiv:2501.09747*, 2025.
- 392 [36] M. J. Kim, Y. Gao, T.-Y. Lin, Y.-C. Lin, Y. Ge, G. Lam, P. Liang, S. Song, M.-Y. Liu, C. Finn,
393 and J. Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning.
394 *arXiv preprint arXiv:2601.16163*, 2026.
- 395 [37] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi,
396 C. Pascal, M. Russi, A. Marafioti, et al. Smolvla: A vision-language-action model for afford-
397 able and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.
- 398 [38] Z. Hou, T. Zhang, Y. Xiong, H. Duan, H. Pu, R. Tong, C. Zhao, X. Zhu, Y. Qiao, J. Dai, et al.
399 Dita: Scaling diffusion transformer for generalist vision-language-action policy. In *Proceed-*
400 *ings of the IEEE/CVF International Conference on Computer Vision*, pages 7686–7697, 2025.
- 401 [39] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, S. Palma, P. Kooijmans, M. Ar-
402 actingi, M. Shukor, D. Aubakirova, M. Russi, F. Capuano, C. Pascal, J. Choghari, J. Moss,
403 and T. Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch.
404 <https://github.com/huggingface/lerobot>, 2024.
- 405 [40] J. Zheng, J. Li, Z. Wang, D. Liu, X. Kang, Y. Feng, Y. Zheng, J. Zou, Y. Chen, J. Zeng, et al. X-
406 vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model.
407 *arXiv preprint arXiv:2510.10274*, 2025.

- 408 [41] H. Fang, J. Duan, D. Clay, S. Wang, S. Liu, W. Huang, X. Fan, W.-C. Tsai, S. Chen, Y. R.
409 Wang, et al. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint*
410 *arXiv:2605.02881*, 2026.
- 411 [42] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haz-
412 iza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision.
413 *arXiv preprint arXiv:2304.07193*, 2023.
- 414 [43] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint*
415 *arXiv:1711.05101*, 2017.
- 416 [44] NVIDIA. Cosmos-reason2: Physical ai common sense and embodied reasoning models.
417 <https://huggingface.co/nvidia/Cosmos-Reason2-2B>, 2026. Model card.

419 **A Implementation Details**

Table 3: **Backbone and model configurations.** We use frozen DINOv2 image encoders and frozen CLIP text encoders for RAIN-L and RAIN-S.

Item	RAIN-L	RAIN-S
DINO backbone	DINOv2-L/14	DINOv2-S/14
CLIP text encoder	ViT-L/14	ViT-B/32
DINO token dim	1024	384
CLIP token dim	768	512
Hidden dim	1024	768
DiT blocks / heads	12 / 12	12 / 12

420 **Model architecture.** RAIN uses DINOv2 [42] as the vision encoder and CLIP [2] as the text
 421 encoder, both frozen during training. RAIN-L uses DINOv2 ViT-L/14 and CLIP ViT-L/14, while
 422 RAIN-S uses DINOv2 ViT-S/14 and CLIP ViT-B/32; detailed dimensions are summarized in Tab. 3.
 423 The Target-adaptive Cross-view Encoder (TCE) uses a single cross-view Transformer block com-
 424 posed of cross-attention, self-attention, and a feed-forward network. When three views are available,
 425 each view is used as the query and the concatenation of the other two views is used as the key and
 426 value for cross-attention. PlanDiT consists of 12 Transformer layers. We set the action chunk size
 427 to 16 and the plan token length to 8 during training. During rollout, PlanDiT uses 4 diffusion steps
 428 and the controller executes the first 8 actions from each predicted chunk.

429 **Training.** All models are trained with a batch size of 1024 for 100k iterations using AdamW [43]
 430 with a base learning rate of 1×10^{-4} . We use linear warmup for the first 2,000 steps followed
 431 by a cosine decay schedule. For stable optimization, we use mixed-precision training and gradient
 432 clipping with threshold 1.0. Training follows a two-stage procedure: we first train all modules except
 433 the Progress Head, then freeze all other parameters and train the Progress Head. The loss weights
 434 are set empirically to $\lambda_{\text{fm}} = 1.0$, $\lambda_{\text{cons}} = 0.1$, $\lambda_{\text{dist}} = 0.3$, $\lambda_{\text{align}} = 0.3$, and $\lambda_{\text{comp}} = 1.2$. All
 435 experiments are run on 4 NVIDIA B200 GPUs.

436 **Retarget augmentation.** Our retarget augmentation reconstructs future interactions from the same
 437 demonstration episode at the current observation timestep. Importantly, it does not create arbitrary
 438 cross-episode combinations; candidates are restricted to future subtasks that can be executed imme-
 439 diately from the current state. We only sample subtasks that occur after the current subtask, never
 440 past subtasks or subtasks from other episodes, and do not use reverse augmentation in our experi-
 441 ments. We estimate the current hand state and filter candidates by action type: an empty hand permits
 442 *grasp* and *close*, while an object-holding hand permits *release* and *close*. This prevents physically in-
 443 valid retargets such as releasing without a held object or grasping a new object while already holding
 444 one. To avoid trivial continuation of the original task, we also exclude the original task-completion
 445 edge. In particular, release retargeting does not reuse the current subtask’s original place target. For
 446 example, *grasp mug* followed by *place mug in microwave* is treated as the original task progression,
 447 whereas *holding mug* followed by *close microwave door* is a valid retarget candidate.

448 Candidates are further filtered by visual and geometric validity. A future target must have a valid
 449 target mask, its mask must not overlap too heavily with the current object mask, and the target
 450 interaction point must be sufficiently far from the current end-effector pose. We use a minimum
 451 distance threshold of 3 cm to avoid near-duplicate state copies. For release retargeting, we do not
 452 copy the source object of the future subtask; instead, the current held object is paired with the future
 453 place target. This defines release retargets as interactions between the held object and a feasible
 454 future placement region, enabling combinations such as placing a held white mug on a future right
 455 plate.

Table 4: **Dita LIBERO fine-tuning and reproduction details.** The official Dita recipe reports LIBERO suite results from the released method, while Dita[†] denotes our single joint 4-suite fine-tuning reproduction. Both use the standard LIBERO suites (Spatial, Object, Goal, Long) and exclude LIBERO-90. – denotes not applicable / not reported.

Setting	Dita official recipe	Dita [†] reproduction
<i>Base & shared settings</i>		
Base model	Dita-0.33B	Dita-0.33B
Parameters	334M total / 221M trainable	334M total / 221M trainable
Text encoder	Frozen CLIP text encoder	Frozen CLIP text encoder
Vision backbone	DINOv2, 224×224	DINOv2, 224×224
Visual reducer	Q-Former, depth 4, 32 tokens	Q-Former, depth 4, 32 tokens
Policy backbone	LLaMA2-style Transformer, 12 layers, hidden 768	LLaMA2-style Transformer, 12 layers, hidden 768
Action decoder	DDPM diffusion head	DDPM diffusion head
DDPM train steps	100	100
Inference steps	10	10
Action chunk	10	10
Action dim.	7	7
Optimizer	AdamW	AdamW
Image augmentation	yes	yes
<i>Training recipe</i>		
Fine-tuning scheme	Suite-specific FT	Single joint 4-suite FT
Training suites	Spatial/Object/Goal/Long	Spatial, Object, Goal, Long
Excluded suite	LIBERO-90	LIBERO-90
Dataset name	libero_{spatial,object,goal,10}_no_noops	libero_all_no_noops
Dataset mixture	–	Four no-noops RLDS datasets, weight 1.0 each
Base checkpoint	dit_policy_checkpoint.pth	dit_policy_checkpoint.pth
Finetune scope	Full FT except CLIP text encoder	Full FT except CLIP text encoder
# GPUs	8	8
Batch size / GPU	64	16
Effective batch size	512	128
Learning rate	1e−4; Long uses 5e−4	1e−4

456 **Observation and deployment setup.** In LIBERO simulation, images are resized to 224×224 and
457 RAIN uses two views: agent view and wrist view. We use the object IDs and segmentation masks
458 provided by the LIBERO simulator for both views. In real-world experiments, images are resized
459 to 336×336 and RAIN uses three views: top view, third-person view, and wrist view. A Cosmos-
460 Reason2-2B model [44] decomposes each task into subtasks and predicts the initial target bounding
461 box; SAM 2 [23] then tracks real-time masks for the top and third-person views, while the wrist view
462 is used as an RGB observation without a mask. During multi-subtask execution, the Progress Head
463 marks the current subtask as completed when its completion score exceeds 0.7 for two consecutive
464 timesteps, after which the policy switches to the next subtask.

465 B Simulation Experiment Details

466 B.1 Finetuning of Baseline Model on LIBERO

467 Standard LIBERO recipes are typically optimized for suite-specific performance, reporting results
468 using separate fine-tuned checkpoints for each individual suite (i.e., LIBERO-Spatial, Object, Goal,
469 and Long). To evaluate a model’s capacity to absorb the entire distribution of standard tasks within
470 a single policy, we reproduce representative baseline methods under a joint 4-suite fine-tuning pro-
471 tocol. Our reproduction strictly adheres to the officially released training codes and configurations,
472 with the sole modification being the dataset mixture to train a single checkpoint on the union of
473 all four standard suites. Specifically, for Dita [38] and GR00T N1.6 [1], we strictly follow the hy-
474 perparameter settings and optimization pipelines provided in their respective official repositories,
475 ensuring a faithful reproduction under the combined multi-suite data distribution.

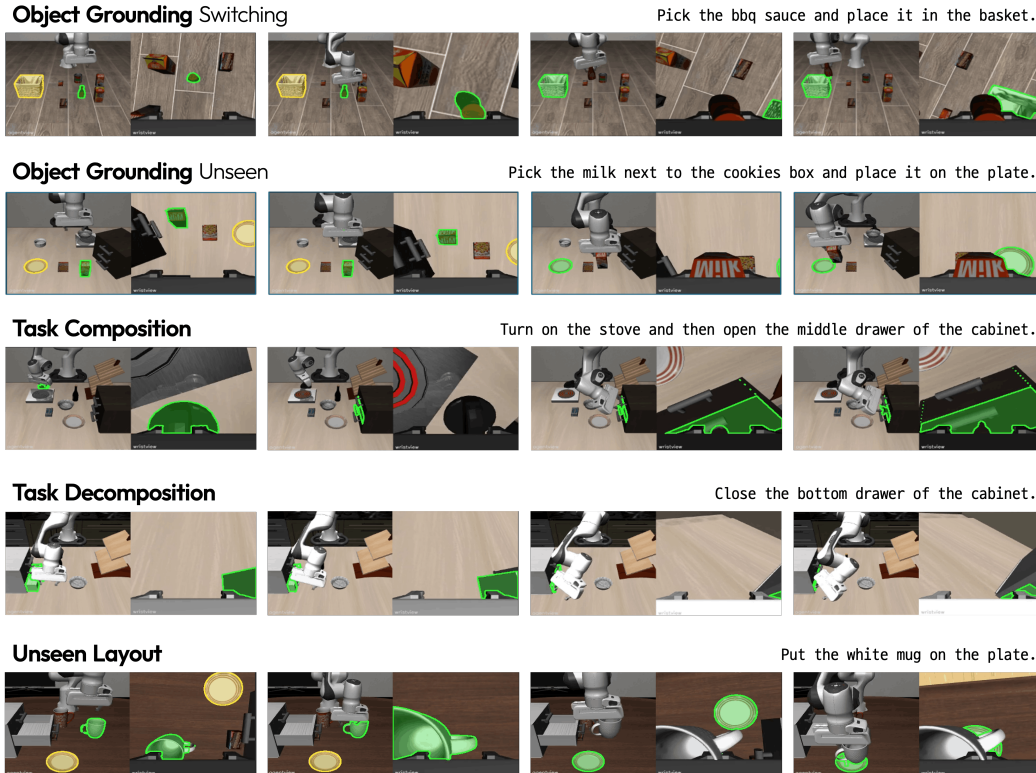


Figure 4: **Qualitative results on LIBERO-EX.**

476 B.2 LIBERO-EX

477 LIBERO-EX is derived from LIBERO to evaluate whether a policy trained on the original LIBERO
 478 tasks can generalize beyond the exact task settings seen during post-training. We organize LIBERO-
 479 EX into three splits, *object*, *layout*, and *task*, corresponding to object grounding, layout robustness,
 480 and task comprehension. Each split contains 25 tasks and each task is evaluated over 10 episodes.
 481 The task definitions and selected initial observations are shown in Tab. 8–Tab. 12, and detailed per-
 482 task success rates corresponding to Tab. 1 are reported in Tab. 5–Tab. 7. We also provide qualitative
 483 results of RAIN in Fig. 4.

484 **Object grounding.** The object split tests whether a policy can ground the intended object rather than
 485 replaying a scene-specific trajectory. It contains two types of tasks. *Unseen object* tasks preserve the
 486 learned task structure but require interaction with objects that did not appear in the corresponding
 487 training scene. *Object switching* tasks use familiar scenes but swap the positions of multiple objects,
 488 forcing the policy to select the instruction-relevant object under changed spatial assignments. As
 489 shown in Tab. 5, RAIN-L achieves 76.0 average success, outperforming the strongest baseline $\pi_{0.5}$
 490 by 11.6 points.

491 **Layout robustness.** The layout split changes the scene composition of learned tasks by rearranging
 492 objects and adding distractors. Compared with object grounding, this split introduces more aggres-
 493 sive visual and geometric changes because the relevant object must be identified and manipulated in
 494 a scene whose overall arrangement differs from training. Several baselines struggle under this shift:
 495 Action-Sketcher drops from 47.6 on object grounding to 27.6 on layout robustness, GROOT N1.6
 496 reaches only 31.2, and Dita reaches 26.0. In contrast, RAIN-L maintains similar performance across
 497 the two splits, achieving 76.0 on object grounding and 72.0 on layout robustness. Per-task results
 498 are provided in Tab. 6.

499 **Task comprehension.** The task split evaluates whether a policy understands the structure of a task
500 rather than memorizing the demonstrated sequence. It consists of *task decomposition*, where a pol-
501 icy trained on a long-horizon task must execute only the requested constituent subtask, and *task*
502 *composition*, where independently learned single-step tasks are combined into a new long-horizon
503 instruction. For decomposition tasks, completing the original full long-horizon task instead of the re-
504 quested subtask is counted as failure. As shown in Tab. 7, most VLA baselines fail on this split, with
505 all compared baselines below 10 average success. In particular, task composition is rarely solved,
506 indicating that existing policies heavily rely on memorized post-training trajectories rather than com-
507 positional task understanding. RAIN-L and RAIN-S substantially improve this split, achieving 53.2
508 and 50.4 average success, respectively.

Table 5: **Detailed LIBERO-EX Object Grounding Results.** Each column is a LIBERO-EX task index from the task-definition tables above. Success rates are reported in % over 10 evaluation episodes, with the category average in the last column.

Model	Task ID																									Avg.
	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015	016	017	018	019	020	021	022	023	024	025	Avg.
OpenVLA-OFT	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.6
π_0	10	20	10	0	40	70	90	0	90	0	0	0	0	60	100	0	0	0	0	100	100	90	100	0	90	38.8
π_0 -FAST	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.4
$\pi_{0.5}$	60	80	60	100	100	100	90	20	100	0	40	40	0	100	100	10	10	0	0	100	100	100	100	100	100	64.4
GR00T N1.6	20	30	50	40	0	10	40	0	80	0	0	0	0	80	70	0	0	0	0	70	70	80	80	80	80	35.2
Action-Sketcher	20	40	60	40	80	70	90	0	100	0	0	0	0	90	100	0	0	0	0	100	100	100	100	0	100	47.6
Cosmos Policy	0	0	10	60	30	40	90	0	100	0	0	0	0	100	100	0	0	0	0	100	100	100	100	0	90	40.8
SmolVLA	0	0	0	0	30	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	2.0
Dita	0	0	0	0	20	50	10	0	100	0	0	0	0	80	20	0	0	0	0	90	90	50	100	40	90	29.6
RAIN-L	70	80	60	80	60	70	70	50	100	100	50	80	100	100	100	60	40	40	50	100	70	80	100	90	100	76.0
RAIN-S	0	60	60	30	20	0	50	20	80	90	100	90	100	100	100	10	80	10	0	100	60	70	70	30	10	53.6

Table 6: **Detailed LIBERO-EX Layout Robustness Results.** Each column is a LIBERO-EX task index from the task-definition tables above. Success rates are reported in % over 10 evaluation episodes, with the category average in the last column.

Model	Task ID																									Avg.
	026	027	028	029	030	031	032	033	034	035	036	037	038	039	040	041	042	043	044	045	046	047	048	049	050	Avg.
OpenVLA-OFT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0
π_0	20	50	60	70	40	60	0	50	90	80	60	0	100	0	30	0	10	30	0	100	20	40	0	10	10	37.2
π_0 -FAST	10	0	10	0	0	0	0	0	0	0	0	10	0	10	30	20	50	40	20	20	30	60	10	0	0	12.8
$\pi_{0.5}$	70	60	100	100	100	80	40	100	100	90	90	0	100	0	90	0	100	100	0	100	0	10	0	0	0	57.2
GR00T N1.6	0	0	50	40	70	10	0	70	20	70	40	0	80	0	0	10	80	30	50	40	30	40	30	10	10	31.2
Action-Sketcher	10	70	50	90	40	40	0	90	10	100	20	0	30	0	40	10	10	0	0	40	0	30	0	10	0	27.6
Cosmos Policy	10	10	40	100	80	90	10	100	70	40	60	0	100	0	90	0	100	100	0	100	0	60	0	0	50	48.4
SmolVLA	0	0	0	0	0	0	0	0	0	0	0	10	20	0	0	0	10	60	10	20	20	0	0	0	0	6.0
Dita	10	0	0	90	80	90	0	70	30	70	40	0	40	0	10	0	0	40	0	60	0	20	0	0	0	26.0
RAIN-L	80	70	90	100	90	100	10	80	90	100	90	100	90	10	80	20	100	100	70	90	60	100	20	50	10	72.0
RAIN-S	20	60	80	90	80	60	30	80	20	40	50	60	70	0	70	0	100	100	50	100	50	90	10	0	40	54.0

Table 7: **Detailed LIBERO-EX Task Comprehension Results.** Each column is a LIBERO-EX task index from the task-definition tables above. Success rates are reported in % over 10 evaluation episodes, with the category average in the last column.

Model	Task ID																									Avg.
	051	052	053	054	055	056	057	058	059	060	061	062	063	064	065	066	067	068	069	070	071	072	073	074	075	Avg.
OpenVLA-OFT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0
π_0	0	0	0	0	0	0	0	0	0	0	10	0	20	30	20	0	40	0	10	0	10	0	0	10	0	6.0
π_0 -FAST	0	0	0	0	0	0	0	0	0	0	0	20	40	0	0	20	40	0	0	20	10	0	20	0	10	7.2
$\pi_{0.5}$	0	0	0	0	0	0	20	30	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	2.4
GR00T N1.6	0	0	0	0	0	0	0	0	0	0	20	10	0	0	0	10	20	0	0	0	10	0	0	0	0	2.8
Action-Sketcher	0	0	0	0	0	0	0	0	0	0	0	0	0	20	10	0	20	10	0	10	10	10	0	0	0	3.2
Cosmos Policy	0	0	0	0	0	0	0	0	0	0	10	0	0	30	0	0	10	0	0	0	0	50	0	40	0	5.6
SmolVLA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	10	0	0	0	0	10	2.0
Dita	0	0	0	0	0	0	0	0	0	0	20	10	50	10	40	0	20	0	50	0	40	0	0	0	0	9.6
RAIN-L	10	0	0	20	20	10	20	30	20	20	90	100	100	100	70	90	100	20	50	70	100	70	100	100	20	53.2
RAIN-S	10	0	10	0	30	0	0	40	20	0	80	100	60	90	60	70	100	10	90	100	80	100	100	90	20	50.4

Table 8: **Object grounding tasks.** Selected initial images for the object grounding category, part 1 of 2.

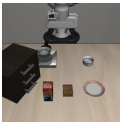




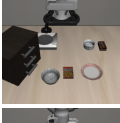







Image	Task ID	Image Type	Description
	001	Unseen object	Pick the milk next to the cookies box and place it on the plate.
	002	Unseen object	Pick the alphabet soup next to the plate and place it on the plate.
	003	Unseen object	Pick the tomato sauce next to the plate and place it on the plate.
	004	Unseen object	Put the white mug on the stove.
	005	Unseen object	Pick the tomato sauce between the plate and the ramekin and place it on the plate.
	006	Unseen object	Pick the chocolate pudding next to the ramekin and place it on the plate.
	007	Unseen object	Pick the cream cheese next to the cookies box and place it on the plate.
	008	Object switching	Pick the salad dressing and place it in the basket.
	009	Object switching	Pick the milk and place it in the basket.
	010	Object switching	Pick the tomato sauce and place it in the basket.
	011	Object switching	Pick the alphabet soup and place it in the basket.
	012	Object switching	Pick the butter and place it in the basket.
	013	Object switching	Pick the ketchup and place it in the basket.

Table 9: **Object grounding tasks.** Selected initial images for the object grounding category, part 2 of 2.













Image	Task ID	Image Type	Description
	014	Object switching	Pick the tomato sauce and place it in the basket.
	015	Object switching	Pick the cream cheese and place it in the basket.
	016	Object switching	Pick the bbq sauce and place it in the basket.
	017	Object switching	Pick the ketchup and place it in the basket.
	018	Object switching	Pick the tomato sauce and place it in the basket.
	019	Object switching	Pick the orange juice and place it in the basket.
	020	Object switching	Pick the chocolate pudding and place it in the basket.
	021	Object switching	Pick the bbq sauce and place it in the basket.
	022	Object switching	Pick the orange juice and place it in the basket.
	023	Object switching	Pick the milk and place it in the basket.
	024	Object switching	Pick the cream cheese and place it in the basket.
	025	Object switching	Pick the alphabet soup and place it in the basket.

Table 10: **Layout robustness tasks.** Selected initial images for the layout robustness category.

Image	ID	Description	Image	ID	Description
	026	Put the white mug on the left plate.		027	Put the white mug on the left plate.
	028	Put the white mug in the basket.		029	Put the white mug on the plate.
	030	Put the white mug on the plate.		031	Put the white mug on the plate.
	032	Put the chocolate pudding in the basket.		033	Put the white mug on the plate.
	034	Put the white mug on the left plate.		035	Put the white mug on the left plate.
	036	Put the yellow and white mug in the microwave.		037	Close the microwave.
	038	Put the yellow and white mug in the microwave.		039	Close the microwave.
	040	Close the microwave.		041	Close the middle drawer of the white cabinet.
	042	Turn on the stove.		043	Turn on the stove.
	044	Close the middle drawer of the cabinet.		045	Put the black bowl in the middle drawer of the cabinet.
	046	Close the middle drawer of the cabinet.		047	Close the top drawer of the cabinet.
	048	Close the bottom drawer of the cabinet.		049	Put the yellow and white mug in the microwave.
	050	Turn on the stove.			

Table 11: **Task comprehension tasks.** Selected initial images for the task comprehension category, part 1 of 2.






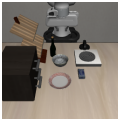
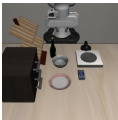
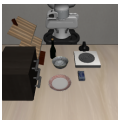

















Image	Task ID	Image Type	Description
	051	Task composition	Put the bowl on the plate, and then open the middle drawer of the cabinet.
	052	Task composition	Put the wine bottle on the rack, and then open the top drawer and put the bowl inside.
	053	Task composition	Put the wine bottle on the rack, and then turn on the stove.
	054	Task composition	Open the top drawer and put the bowl inside, and then put the wine bottle on top of the cabinet.
	055	Task composition	Open the top drawer and put the bowl inside, and then turn on the stove.
	056	Task composition	Put the cream cheese in the bowl, and then put the wine bottle on top of the cabinet.
	057	Task composition	Put the cream cheese in the bowl, and then push the plate to the front of the stove.
	058	Task composition	Put the cream cheese in the bowl, and then turn on the stove.
	059	Task composition	Push the plate to the front of the stove, and then turn on the stove.
	060	Task composition	Turn on the stove, and then open the middle drawer of the cabinet.
	061	Task decomposition	Put the white mug on the left plate.
	062	Task decomposition	Put the yellow and white mug on the right plate.
	063	Task decomposition	Put the white mug on the plate.

Table 12: **Task comprehension tasks.** Selected initial images for the task comprehension category, part 2 of 2.

Image	Task ID	Image Type	Description
	064	Task decomposition	Put the chocolate pudding to the right of the plate.
	065	Task decomposition	Put the yellow and white mug in the microwave.
	066	Task decomposition	Close the microwave.
	067	Task decomposition	Turn on the stove.
	068	Task decomposition	Put the moka pot on the stove.
	069	Task decomposition	Put the alphabet soup in the basket.
	070	Task decomposition	Put the cream cheese box in the basket.
	071	Task decomposition	Put the alphabet soup in the basket.
	072	Task decomposition	Put the tomato sauce in the basket.
	073	Task decomposition	Put the cream cheese box in the basket.
	074	Task decomposition	Put the butter in the basket.
	075	Task decomposition	Close the bottom drawer of the cabinet.

509 **C Real-World Experiment Details**

510 In this section, we provide additional details for the real-world robot experiments, including the
511 robot setup, dataset construction, evaluation splits, and baseline training details.

512 **C.1 Common Setup**

513 **Robot platform.** Our real-world setup is shown in Fig. 5. We use a ROBOTIS OMY-F3M robot
514 equipped with a 6-DoF manipulator driven by high-precision DYNAMIXEL-Y actuators and a 1-
515 DoF two-finger gripper. For perception, we use three RGB camera viewpoints: an Intel RealSense
516 D405 wrist camera mounted near the end-effector, and two Intel RealSense D435 cameras for the
517 third-person and top-down views.

518 **Control interface.** All methods use the same 7-dimensional joint-space policy interface. The robot
519 state consists of six arm joint positions and one gripper joint value, and the policy action consists of
520 six absolute target joint angles and one gripper command. We use absolute joint-pose control instead
521 of end-effector delta control for real-world execution, so the predicted actions can be directly sent
522 to the OMY-F3M joint trajectory controller. The robot client executes joint commands at 30 Hz.

523 **Deployment infrastructure.** Robot control is managed by a ROS-based rollout client, while each
524 policy runs as a separate model server on an NVIDIA RTX PRO 6000 workstation. The rollout
525 client reads live camera frames and robot joint states, sends the assembled observation to the se-
526 lected model server, receives an action chunk, and publishes the predicted joint targets to the robot
527 controller. The client queries the model server asynchronously while executing the current action
528 chunk, allowing policy inference and robot control to run concurrently.

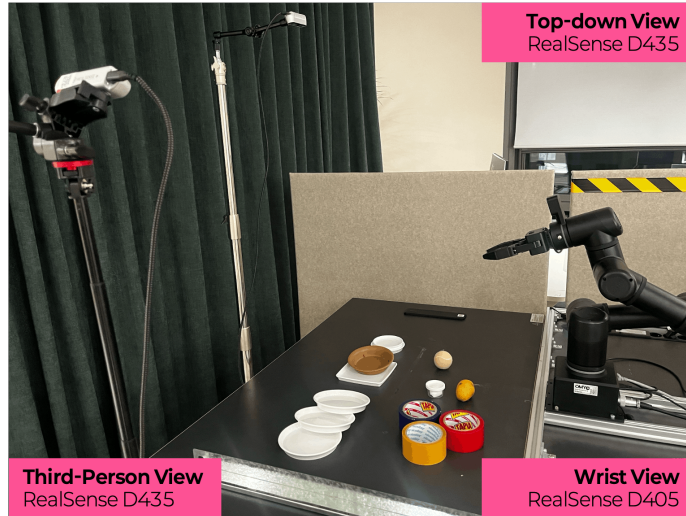


Figure 5: **Real-world experimental setup.** The OMY-F3M robot is observed by wrist, third-person, and top-down RGB cameras and controlled through a 7D joint-absolute action interface.

529 **Real-world dataset.** We collect OMY-F3M demonstrations using a ROS-based recorder and store
530 them in the LeRobot format [39]. Each trajectory contains synchronized RGB observations from the
531 wrist, third-person, and top-down cameras, robot joint states, actions, and end-effector poses at 30
532 Hz. We construct an aggregated real-world dataset from the task-specific datasets. All tasks share
533 the same embodiment, camera setup, state representation, and 7D joint-absolute action space.

534 **C.2 Real-World Task Suites**

535 Rather than benchmarking on standard, isolated scenarios, our real-world evaluation protocol is ex-
536 plicitly constructed to heavily prioritize out-of-distribution (OOD) generalization within visually



Figure 6: **Qualitative samples from the Potato-to-Plate task.** All 40 collected episodes are shown from the top camera, with one representative frame sampled from each trajectory. The grid is ordered row-major by episode index: the horizontal direction advances through consecutive episodes, and the vertical direction continues the episode sequence after each row.

537 and spatially cluttered settings. All methods are trained on identical, data-efficient sets of real-robot
 538 demonstrations within each family and evaluated using the same physical platform, camera config-
 539 uration, action interface, initial-state protocol, and success criteria.

540 While In-Distribution (ID) trials establish baseline performance, our primary focus lies in these
 541 comprehensive OOD evaluation splits. A central objective of these splits is to expose policies to
 542 dense, cluttered environments where multiple objects are scattered and reshuffled simultaneously
 543 across the workspace. By heavily stacking these variations—spanning permuted relative layouts,
 544 completely unseen target categories, and displaced target positions—we force the models to oper-
 545 ate in scenes packed with visual distractors. This rigorous, OOD-centric protocol explicitly tests
 546 whether a policy possesses the fine-grained visual-spatial reasoning required to segment and ground
 547 instruction-relevant regions out of a cluttered scene, or if it collapses under superficial shortcuts
 548 memorized from the clean, structured training demonstrations.

549 To qualitatively summarize the real-world data collection process, we visualize representative frames
 550 sampled from the collected demonstrations in Figures 6–7. These visualizations show the range of
 551 object layouts, camera viewpoints, and trajectory-level variation used to construct the real-world
 552 training datasets.

553 For visualization, we therefore project object-pose distributions onto the actual camera images and
 554 additionally show ROI-focused views that highlight the small manipulation region in which success-
 555 ful actions must be grounded.

556 C.3 RAIN Real-World Deployment Architecture

557 To execute long-horizon natural language instructions without control-loop bottlenecks, we deploy
 558 our framework using a decoupled serving stack engineered specifically to maximize the through-
 559 put of the **Region-Aware Interaction Network (RAIN)**. As schematically illustrated in Figure 9,
 560 the system architecture segregates the unconstrained, low-frequency visual-symbolic grounding of
 561 general-purpose Vision-Language Models (VLMs) from RAIN’s high-frequency visuomotor action
 562 generation loop.

563 A central design paradigm of this deployment is that the VLM does not function as a per-frame
 564 closed-loop controller, a live completion judge, or an online task planner. Instead, the symbolic

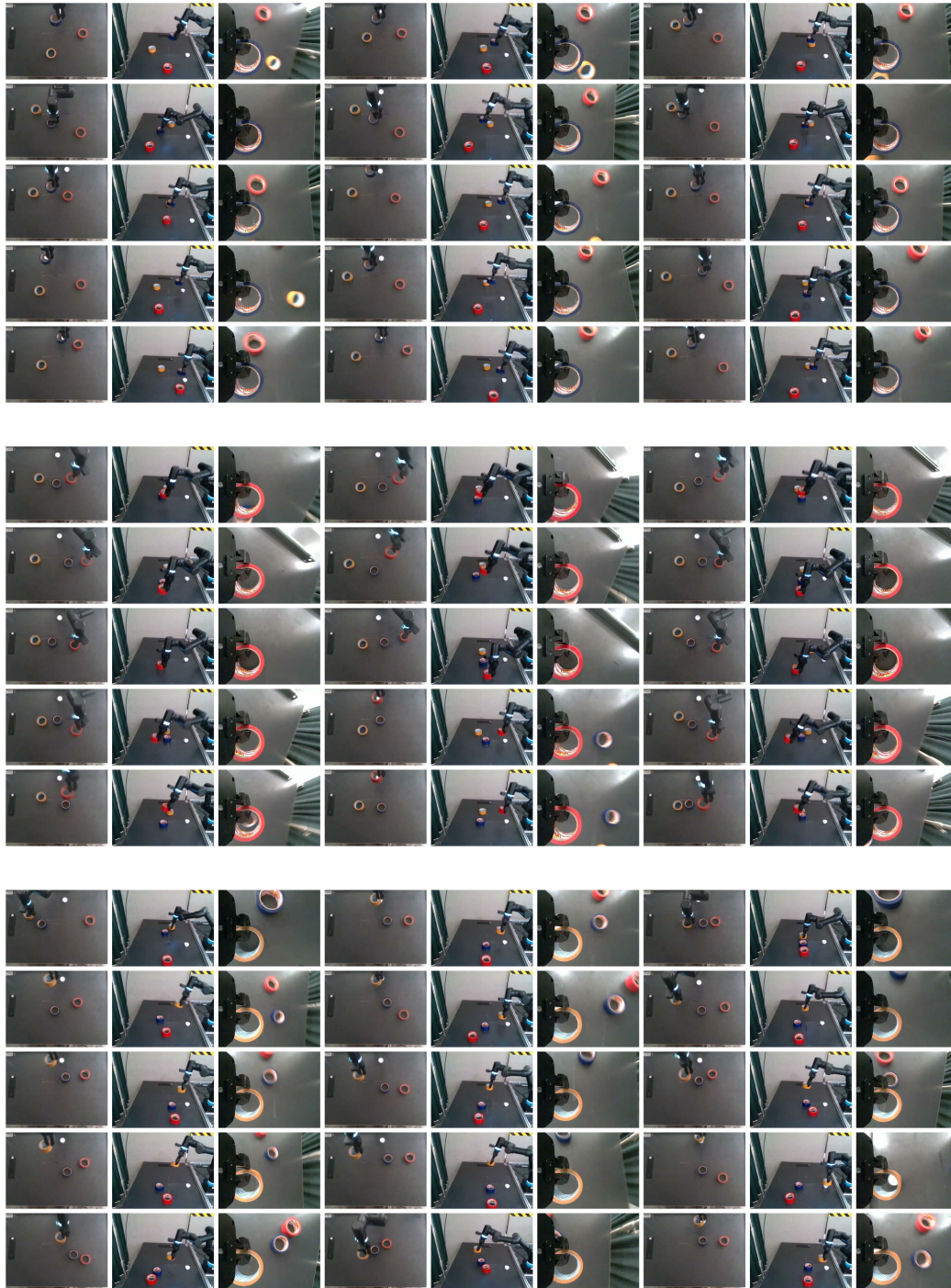


Figure 7: **Qualitative samples from the Tape-to-Clay task suite.** The three bands show blue, red, and yellow tape variants, respectively. For each variant, all 15 collected episodes are shown, with columns repeating top, third-person, and wrist camera views. The horizontal direction is organized as episode-wise camera triplets (top/third-person/wrist) for consecutive episodes, while the vertical direction continues the remaining episode groups. The vertical stacking across bands corresponds to the target tape color rather than a physical workspace coordinate.

565 instruction decomposition is offloaded onto a structured task registry that specifies the ordered sub-
566 tasks, target linguistic phrases, and low-level action primitives.

567 At the temporal boundary of each subtask transition, the live mask server invokes the VLM
568 (Cosmos-Reason2-2B) exactly once. The VLM processes the current multi-view observation to
569 ground the active target phrase (e.g., “*the potato*”) into a localized 2D bounding box, which serves
570 as a zero-shot spatial anchor to initialize a high-speed SAM 2 tracking thread. Once initialized, SAM
571 2 handles token-free mask propagation across subsequent frames, completely removing the heavy
572 VLM inference overhead from the active control loop.

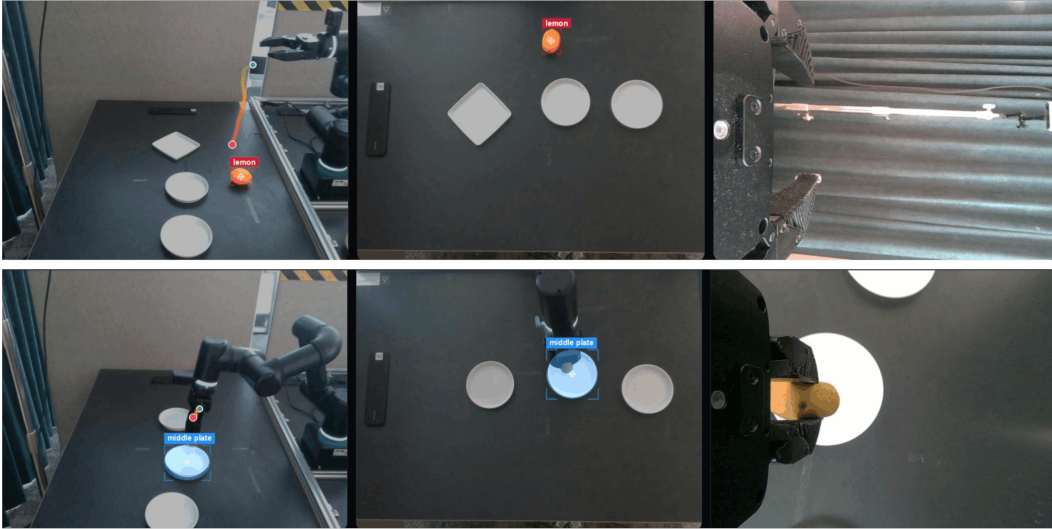


Figure 8: **Real-World Multi-View Setup and Target Grounding Visualization.** Examples of syn-
synchronized multi-view observations used in our real-world experiments: third-person, top-down, and
wrist camera views. The perception pipeline dynamically initializes a zero-shot 2D bounding box
via the VLM (e.g., grounding “middle plate”) and propagates the region mask to guide the RAIN
motor policy for robust manipulation.

573 Downstream low-level control is entirely executed by the RAIN motor server at 30 Hz. The RAIN
574 policy consumes multi-view RGB observations, the tracked SAM 2 region masks, and the 7-DoF
575 robot state configured under a strict joint-position and joint-absolute action space contract. By ex-
576 plicitly injecting the tracked masks into RAIN’s target-aware visual layers, the policy jointly pre-
577 dicted target-directed action chunks ($H_{\text{action}} = 16$, $H_{\text{execute}} = 10$) and an auxiliary task-completion
578 (TC) score. Sequential subtask switching is triggered autonomously by RAIN’s internal progress-
579 estimation head ($tc \geq 0.7$ over consecutive frames) or manually overridden via the operator dash-
580 board. Upon a transition signal, the client advances the registry pointer, dynamically flashing the
581 next target phrase to re-initialize the VLM bounding box grounding cycle.

582 **Interactive mask retargeting.** To further verify that RAIN uses the region mask as an actionable
583 conditioning signal, we evaluate an interactive click-to-act mode. In this mode, the operator selects a
584 target region directly from the live dashboard by clicking on the camera stream. The click initializes
585 a SAM 2 point-prompt mask for the selected connected component, and the resulting mask is sent to
586 RAIN through the exact same interface used during autonomous operations. The policy checkpoint,
587 robot state interface, action space, and active action type remain completely unchanged; only the
588 target region mask is dynamically altered.

589 This setup provides a rigorous causal test of region conditioning. For example, during a placement
590 subtask, the operator can click different candidate plates or target objects within the same scene.
591 When the clicked mask changes from one target region to another, RAIN dynamically retargets
592 the predicted end-effector trajectory and the executed placement behavior toward the newly selected

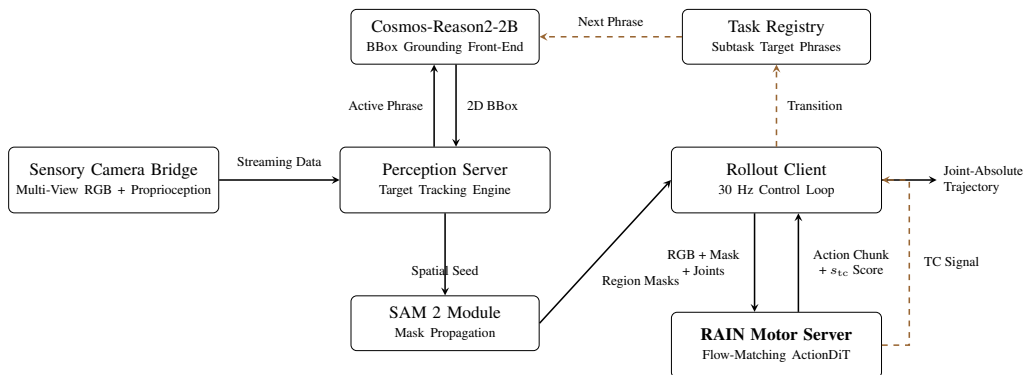


Figure 9: **RAIN Real-World Serving Pipeline**. The architecture explicitly separates sparse vision-language target initialization from closed-loop control. At each subtask boundary, the VLM grounds the registry phrase into a 2D bounding box to seed SAM 2 tracking. The resulting region masks are directly consumed by the **RAIN motor server** alongside multi-view RGB observations to generate joint-absolute action chunks and drive autonomous progress estimation.

593 region in real-time. Similarly, during a grasp subtask, clicking a different valid source object changes
 594 the grasp target while preserving the underlying low-level grasp primitive.

595 **Protocol.** We keep the scene, robot checkpoint, task family, and action interface entirely fixed
 596 across all interactive evaluations. For each trial, we manually select the active subtask from the
 597 dashboard, e.g., `grasp` for source-object selection or `place` for placement-target selection. We then
 598 click a target region in the live camera stream to initialize a SAM 2 mask. The rollout client re-
 599 ceives this clicked mask exactly as it receives an autonomous VLM-grounded mask. We compare
 600 the emergent trajectories across different clicked regions while keeping all other inputs fixed. A suc-
 601 cessful retargeting trial is counted when the robot correctly executes the intended primitive toward
 602 the clicked region, such as grasping the specific clicked object or placing the held object onto the
 603 newly designated target.

604 C.4 Real-World Baseline Training Details

605 We fine-tune all baseline policies on the same real-world demonstrations used by RAIN. For each
 606 method, we use the released pretrained checkpoint and official fine-tuning code when available, and
 607 adapt only the input/output interface to the OMY-F3M setting. In particular, each baseline receives
 608 the same three RGB camera views, language instruction, and 7D robot state, and outputs actions
 609 through the same 7D joint-absolute action interface. RAIN additionally receives online masks for
 610 the third-person and top-down views; baselines do not receive these mask-conditioning inputs. The
 611 detailed training configurations and hyperparameter settings for all evaluated models are summa-
 612 rized in Table 13.

613 C.4.1 MolmoAct2

614 We adapt MolmoAct2 [41] using the released LeRobot-style policy implementation, initialized from
 615 the public MolmoAct2 checkpoint.¹ We use this checkpoint because it is the closest released Mol-
 616 moAct2 initialization to our real-world setting: SO-100/101 and OMY-F3M are both tabletop single-
 617 arm manipulation embodiments controlled with continuous joint-space action trajectories, although
 618 their kinematics and action dimensions differ.

619 For OMY-F3M, we adapt the embodiment metadata, dataset interface, state/action dimensionality,
 620 and rollout adapter to the 6-DoF arm plus 1-DoF gripper. The observation contains the same three

¹Allen AI MolmoAct2. https://huggingface.co/allenai/MolmoAct2-S0100_101

Table 13: **Hyperparameters for OMY-F3M real-world finetuning.** All methods are trained on the same OMY-F3M teleoperation trajectories, utilizing 3 camera views (third-person, top-down, wrist) and 7 DoF absolute joint-pose control.

Setting	GR00T N1.6	X-VLA	MolmoAct2	RAIN (Ours)
<i>Base & shared settings</i>				
Base model	GR00T-N1.6-3B	X-VLA-0.9B	MolmoAct2-SO100	DINOv2 ViT-L/14 + DiT
# GPUs	1	1	1	2
Learning rate	1e-4	1e-4	1e-5/5e-5*	1e-4
LR schedule	cosine (wu 0.05)	wu 2k	cosine (wu 0.05)	wu 2k
Total batch size	8	8	8	32
Grad. accum.	1	4	1	1
Weight decay	1e-5	-	1e-5	0.01
Optimizer	AdamW	AdamW	AdamW	AdamW
Precision	bf16	bf16	bf16	bf16
Image input	3 views	3 views	3 views (480×640)	3 views (DINO 448)
Use proprio	yes	yes	yes	yes
Action chunk	16	32	16	16
<i>Method-specific settings</i>				
Finetune scope	proj.+VLLN+DiT+top-4 LLM	soft-prompt PEFT	VLM + action expert	full FT (FiLM)
% trainable param	1.62B (49.3%)	~1%	partial	full
Denosing steps	4	10	8	4
Extra conditioning	-	32-dim domain prompt	-	SAM2 mask + 8 plan wp

621 RGB camera views, language instruction, and 7D proprioceptive state used by the other real-world
622 methods.

623 During fine-tuning, we freeze the Molmo2-derived vision-language backbone, including the vi-
624 sion encoder, vision-language connector, language model, and token embeddings. We train only
625 the continuous action expert, adapting the SO-100/101 action distribution to the OMY-F3M 7D
626 joint-absolute action space. This keeps the visual-language representation fixed and limits train-
627 able capacity for the small real-robot datasets. The adapted policy predicts a 16-step action chunk
628 $a_{t:t+15} \in \mathbb{R}^{16 \times 7}$, where each action consists of six absolute arm-joint targets and one gripper com-
629 mand. The shared rollout client executes these joint-absolute commands at 30 Hz.

630 C.4.2 X-VLA

631 We fine-tune X-VLA [40] from the public X-VLA checkpoint² The model is adapted to the same
632 OMY-F3M LeRobot demonstrations used by all real-world methods, with three RGB views, a lan-
633 guage instruction, a 7D robot state, and a 7D joint-absolute action.

634 X-VLA uses a padded cross-embodiment action interface. We set `action_mode=auto`,
635 `real_action_dim=7`, and `max_action_dim=20`. The first seven dimensions correspond to the
636 OMY-F3M action space, consisting of six arm joint targets and one gripper command, while the
637 remaining dimensions are dummy padded channels. The training loss is applied only to the valid 7D
638 action dimensions, and inference outputs are trimmed back to 7D before execution.

639 We fine-tune X-VLA with PEFT/LoRA adapters using rank 8 and alpha 16. The base pretrained
640 weights remain frozen, while the LoRA adapters, domain soft prompt, and action encoder/decoder
641 components are saved in the adapter checkpoint. We train with AdamW, learning rate 1×10^{-4} , 2k
642 warmup steps, bfloat16 precision, global batch size 8, and gradient accumulation 4.

643 At deployment, X-VLA is served through the same ZMQ rollout interface as the other baselines.
644 The policy receives the current multi-view observation, proprioceptive state, and instruction, then
645 returns a 7D joint-absolute action chunk to the 30 Hz rollout client.

²X-VLA Official Checkpoints. <https://huggingface.co/collections/2toINF/x-vla>,

646 C.4.3 GR00T N1.6

647 We fine-tune GR00T N1.6 [1] using the released Isaac GR00T post-training codebase, initialized
648 from the public GR00T-N1.6-3B checkpoint³. We adapt its embodiment metadata and modality
649 schema to OMY-F3M. The observation contains three RGB views (`cam_wrist`, `cam_third`, and
650 `cam_top`), a language instruction, and a 7D robot state. The state and action are grouped into a
651 6D `single_arm` joint group and a 1D `gripper` group. Both are registered as joint-space, absolute-
652 action modalities, matching the real-robot joint-position controller.

653 During fine-tuning, we freeze the visual encoder and do not fully fine-tune the language backbone.
654 We train the action-head projector, the VL-to-latent module, the diffusion action model, and the top
655 four language-model layers, corresponding to approximately 1.62B trainable parameters (49.29%).
656 The action head is trained with conditional flow matching and uses four integration steps at infer-
657 ence.

658 We train with AdamW, learning rate 1×10^{-4} , cosine decay with warmup ratio 0.05, weight decay
659 1×10^{-5} , bfloat16 precision, and global batch size 8. The adapted policy predicts action chunks
660 of shape 16×7 , consisting of six absolute arm-joint targets and one gripper command. The shared
661 rollout client executes a prefix of each chunk at 30 Hz and then re-queries the policy for closed-loop
662 replanning.

³NVIDIA GR00T-N1.6-3B. <https://huggingface.co/nvidia/GR00T-N1.6-3B>