ACT WITH INTENT: DISTILLING BEHAVIOR INTENT FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Action decoders in vision-language-action (VLA) models are commonly trained by behavior cloning to reproduce demonstrated action sequences. To enrich this supervision, prior work adds training signals based on future observations and motion. These signals describe executions and outcomes, but do not explicitly identify the local objective an action segment serves under the instruction. We propose Intention Distillation (INDI), which makes this behavior-level intent a supervised intermediate state that action generation is trained to use. A frozen teacher vision-language model (VLM) provides multimodal targets from executed demonstration segments under the task instruction. The student learns to predict these targets at an intermediate decoder layer using only its current observations, instruction, and robot state. Context routing and an intent-mismatch objective encourage downstream predictions to depend on the recovered intent, while visual-outcome and textual-purpose targets provide complementary supervision. The teacher and target-generation modules are removed at deployment. INDI improves GR00T-N1.7 from 64.3% to 84.7% on SimplerEnv-Bridge and from 64.1% to 70.3% on RoboCasa Kitchen, with improvements on $\pi_{0.5}$ across both benchmarks. Across standard scenes, held-out objects, and distractor conditions, average real-world success increases from 62.0% to 68.7%. Representation probes and interventions further show that the recovered states encode task and progress information and influence downstream decoding. Anonymous project page: https://indiproject.github.io/

1 INTRODUCTION

Recent vision-language-action (VLA) models and robot foundation models have advanced rapidly by scaling vision-language backbones, action decoders, and robot demonstrations (Zitkovich et al., 2023; Kim et al., 2025; Black et al., 2025b;a; Physical Intelligence et al., 2026; NVIDIA et al., 2025; NVIDIA, 2026; Gemini Robotics Team et al., 2025; Li et al., 2024). These models transfer Internet-scale visual and linguistic knowledge to robot control, enabling specialized action decoders to turn multimodal context into continuous motor commands. Behavior cloning trains these decoders to reproduce demonstrated action sequences from the current observation and instruction. This objective supervises which actions to execute, while leaving each segment's local objective implicit.

To enrich action-level supervision, a growing line of work introduces targets describing future scenes and motion. These include generated frames, subgoal images, and latent observations (Wu et al., 2024; Black et al., 2024; Zhao et al., 2025; Hu et al., 2025; Zheng et al., 2025b), as well as motion fields, point tracks, and structured world dynamics (Xu et al., 2025; Bharadhwaj et al., 2024; Zhang et al., 2025). These targets describe anticipated states and movements. A complementary question is what local objective an action segment serves under the instruction. For example, a placement task includes segments that secure an object for transport and segments that align it for release at the destination. We call the local objective of a segment its *behavior-level intent*. We seek to make this role an explicit source of supervision for the intermediate states used to generate actions.

We propose Intention Distillation (INDI) to supervise behavior-level intent inside the action decoder (Figure 1). Our key idea is to couple purpose-conditioned supervision of an intermediate decoder state with training that promotes its use in action generation.

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

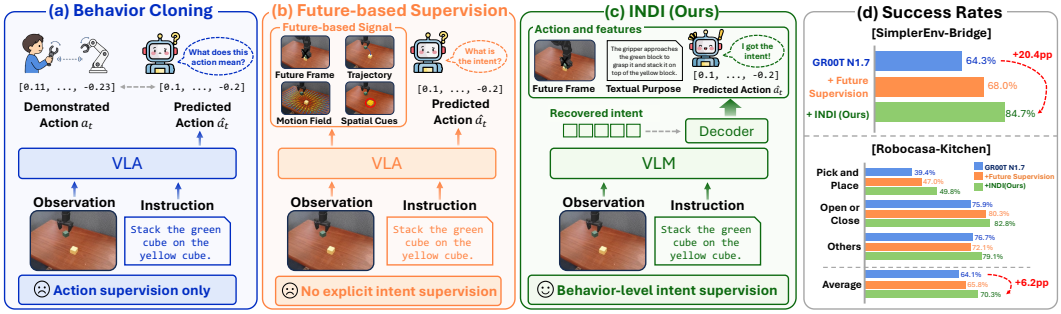


Figure 1: **From behavior cloning to intent distillation.** (a) Behavior cloning directly supervises actions. (b) Future-based supervision adds targets describing future states or motion. (c) During training, INDI supervises an intermediate decoder state using a purpose-conditioned teacher representation of executed behavior and trains subsequent predictions to use it. At deployment, the student recovers this state from current inputs, without the teacher. Right: GR00T-N1.7 results.

Concretely, a frozen teacher VLM processes the current observation, instruction, coarse action summary, and execution video under a prompt asking how the segment advances the task. Its intermediate hidden states over the multimodal input provide the intent target. The student predicts this target at an intermediate decoder layer from its current observation, instruction, and robot state.

Because the decoder also has access to the original VLA context, target alignment alone does not ensure that action prediction uses the recovered state. Context routing intermittently blocks this direct access, while an intent-mismatch objective favors lower downstream loss under matched than substituted states. We additionally supervise predictions of endpoint visual features and features of a teacher-generated purpose statement, training the recovered state to support both outcome and purpose prediction. The teacher is used only during training.

We evaluate whether this supervision improves policy performance and whether the decoder uses the recovered states. On SimplerEnv-Bridge (Li et al., 2025b), INDI improves GR00T-N1.7 from 64.3% to 84.7%. On RoboCasa Kitchen (Nasiriany et al., 2024), it improves the same backbone from 64.1% to 70.3% across 24 tasks. Both benchmarks also show average gains with $\pi_{0.5}$. Average real-world success rises from 62.0% to 68.7% across standard scenes, held-out objects, and distractor conditions. Controlled comparisons support purpose-focused target construction, while ablations show benefits from both state alignment and training for downstream use. Probes recover task and progress information from the learned states, and interventions show that modifying intent and auxiliary states affects downstream predictions and task success.

Contributions.

- We introduce purpose-conditioned supervision inside VLA action decoders by distilling teacher input representations of executed demonstrations into intermediate decoder states.
- We couple this supervision with context routing and intent-mismatch training to promote the use of the recovered states in downstream prediction.
- We demonstrate average success gains on two VLA backbones in simulation and on real-world tasks, with controlled studies of target construction and downstream-use mechanisms.

2 Related Work

Future-based and structured supervision. Future-based supervision trains policies to anticipate the consequences of their actions through future frames, subgoal images, and visual reasoning frames (Wu et al., 2024; Black et al., 2024; Zhao et al., 2025), or through predicted visual features and latent world states (Hu et al., 2025; Tian et al., 2025; Zheng et al., 2025b; Zhang et al., 2025; Xu et al., 2026; Sun et al., 2026; Luo et al., 2026; Won et al., 2026). These predictions can supervise internal representations, as in PHR-VLA, or be learned jointly with action generation, as in JEPA-WAM (Soleymanzadeh et al., 2026; Lin et al., 2026). Related approaches make spatial changes explicit through optical flow, point tracks, visual traces, and grounding masks (Xu et al.,

2025; Bharadhwaj et al., 2024; Zheng et al., 2025a; Huang et al., 2025b), while DIAL and AIM use visual foresight and spatial value maps to guide control (Chen et al., 2026; Fan et al., 2026). These signals describe where objects move or what the scene will become. INDI complements this predictive supervision by asking what an executed segment accomplishes under the instruction. For this purpose, a teacher processes the segment's actions and outcomes under a prompt about its local objective. This adds purpose-focused supervision to the outcome information available for learning, helping the decoder relate the actions it imitates to the task they advance.

Multimodal latent reasoning. Reasoning-based policies make language and visual plans available to action generation (Zawalski et al., 2025; Huang et al., 2025a; Li et al., 2025a). Recent methods learn continuous representations of such plans: Fast-ThinkAct distills linguistic and visual planning through preference-guided learning (Huang et al., 2026a), and LaRA-VLA progressively replaces textual reasoning with latent states while learning visual prediction (Bai et al., 2026). DualCoT-VLA learns parallel visual and linguistic query states supervised by geometric features and text plans, which provide context to an action expert (Zhong et al., 2026). INDI instead supervises the local purpose of an executed segment within the action decoder, using a teacher's representation formed under a purpose prompt. Because the teacher observes both the demonstrated actions and their consequences, it can assess the segment's role using evidence of what actually happened. To use this information for control, the student learns to recover the teacher's input representation from current inputs, while routing and mismatch training encourage the decoder to use this state.

Hierarchical policies and intent representations. Hierarchical policies bridge instructions and motor commands with intermediate plans, ranging from language motions in RT-H (Belkhale et al., 2024) to continuous plans learned from play (Lynch et al., 2020) and language-regularized discrete plans in LADS (Jiang et al., 2025). Trajectory-based abstractions also include the latent-action and future-action representations of WALA and CAC-VLA (Liu et al., 2026; Xiong et al., 2026). IntentVLA also conditions actions on a short-horizon intent representation inferred from recent visual history (Lian et al., 2026). Whereas MINT uses spectral decomposition to separate coarse intent from finer execution details (Huang et al., 2026b), INDI supervises a segment's local objective using a teacher that considers the instruction and observed state changes together with the actions. This lets the supervision relate movement to its role in the task, adding a purpose-focused learning signal alongside trajectory imitation.

3 INDI: INTENTION DISTILLATION

INDI augments action imitation with teacher supervision about what demonstrated behavior accomplishes. A frozen VLM processes each segment's actions and video under the task instruction. The student action decoder learns to predict the teacher's representation from current inputs and to use the recovered state when predicting actions, visual outcomes, and textual purpose (Figure 2).

3.1 PROBLEM FORMULATION

We consider a pretrained VLA with a frozen vision-language module $\mathcal{F}_{\text{vlm}}$ and a trainable flow-matching action decoder. At time t, the policy predicts an action sequence A_t from an RGB observation o_t, instruction ℓ, and proprioceptive state q_t, with VLA context $B_t = \mathcal{F}_{\text{vlm}}(o_t, \ell)$. We also supervise an intermediate state I_t with a teacher representation of executed behavior under ℓ. Executed actions and future observations are available only during training.

3.2 TEACHER TARGETS FROM EXECUTED BEHAVIOR

To obtain supervision without manual purpose annotations, we prompt a frozen VLM to identify how each demonstrated segment advances the instruction. The teacher sees the actions and their observed consequences, allowing it to assess the segment's purpose using what actually happened.

For an H-step segment $\mathcal{W}_t$ (about 1.6 seconds in our simulation benchmarks), the inputs are $\mathcal{E}_t = (o_t, \ell, c_A(\mathcal{W}_t), \text{vid}_{t:t+H})$, where c_A is a coarse textual action summary. The teacher generates a final purpose statement with token span $\mathcal{S}_t$.

Intent target. We construct $I_t^\star$ from middle-layer hidden states over $\mathcal{E}_t$, where the teacher processes the multimodal input under the purpose prompt (Kim et al., 2026a). These states can contain scene,

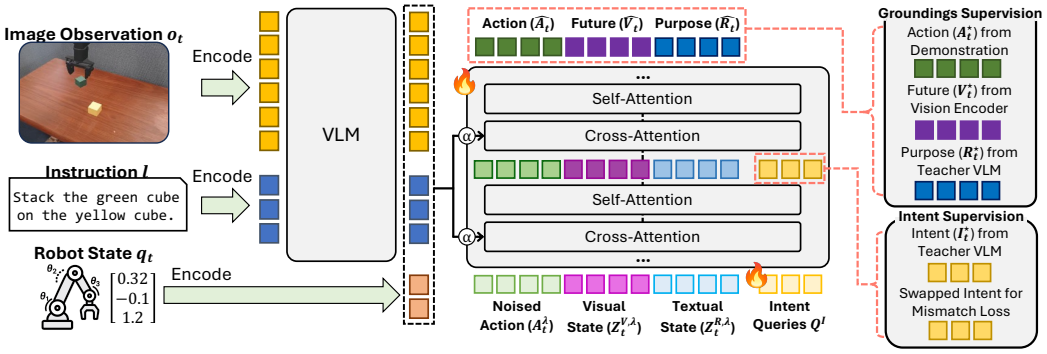


Figure 2: **Overview of INDI.** A frozen teacher processes executed demonstrations to provide targets for intent and textual purpose. These targets come from its input representations and generated purpose statement, respectively. A frozen visual encoder provides the visual-outcome target. The student recovers intent from current inputs and predicts actions and visual and textual features, $\widehat{V}_t$ and $\widehat{R}_t$. During training, α gates context access for non-intent rows, while the mismatch loss favors lower prediction loss with matched intent than with recovered intent swapped from another sample. Dashed lines indicate training signals, and target-generation modules are removed at deployment.

object, and motion information alongside information about the local objective. They are distinct from the generated purpose statement, which supplies a separate target below.

Complementary targets. We also train the decoder to predict the segment's visual outcome and the teacher's description of its purpose. We supervise these predictions with endpoint image features $V_t^\star$ and features of the generated statement $R_t^\star$, building on predictive visual supervision (Wu et al., 2024; Black et al., 2024; Hu et al., 2025; Zhang et al., 2025) and language-based supervision for robot policies (Zawalski et al., 2025; Huang et al., 2025a; Li et al., 2025a; Sumers et al., 2023). The three targets are

$$I_t^\star = \mathrm{pool}_{K_I}\left(T_{\mathcal{E}_t}^{(d_I)}\right), \quad R_t^\star = \mathrm{pool}_{K_R}\left(T_{\mathcal{S}_t}^{(d_R)}\right), \quad V_t^\star = \mathrm{pool}_{K_V}(\Phi_{\mathrm{vis}}(o_{t+H})), \tag{1}$$

where $T_{\mathcal{X}}^{(d)}$ denotes teacher hidden states over span $\mathcal{X}$ at layer d, d_R is the final layer, and Φ_{vis} is a frozen visual encoder. The pooling operators produce K_I, K_R, and K_V target vectors, respectively. We cache all targets offline, as detailed in Section A.4.

3.3 Recovering Intent from Current Inputs

The student learns to predict the teacher representation from the observation, instruction, and robot state available before execution. We add K_I learnable intent query tokens Q^I to the decoder and use their hidden states as the intent rows. Separate visual and textual rows, termed *groundings*, hold the auxiliary predictions. To restrict recovery to current inputs, intent queries read the VLA context, robot state, and one another, without access to noised action or auxiliary rows.

Intermediate alignment. We supervise an intermediate layer so that the recovered state can guide the remaining decoder computation. Following representation alignment in VLA decoders (Zheng et al., 2025b), we align a projection of the resulting state $I_t = H_{I,t}^{L_{\mathrm{tap}}}$ to the teacher target:

$$\mathcal{L}_I = \frac{1}{K_I}\sum_{k=1}^{K_I}\left[1 - \cos\left(P_I(I_{t,k}), I_{t,k}^\star\right)\right]. \tag{2}$$

The decoder uses I_t directly, with P_I mapping to teacher features only for alignment.

3.4 Action and Auxiliary Prediction

After the alignment layer, the decoder uses recovered intent to predict the actions and the visual and textual features introduced above. These prediction tasks train the state to support the demonstrated action, its visual outcome, and its stated purpose. For actions, we retain the backbone's

flow-matching objective (Black et al., 2025a; NVIDIA et al., 2025). It learns a velocity $\widehat{u}_A$ that transforms noise ϵ_A into an action sequence as flow time λ increases:

$$A_t^\lambda = (1-\lambda)\epsilon_A + \lambda A_t, \quad u_A = A_t - \epsilon_A, \quad \mathcal{L}_A = \mathrm{MSE}(\widehat{u}_A, u_A). \tag{3}$$

Visual and textual prediction. We supervise the auxiliary outputs by feature alignment (Yu et al., 2025). For $X \in \{V, R\}$, a projection maps the final decoder-space prediction $\widehat{Z}_t^X$ to the target-space feature $\widehat{X}_t = P_X(\widehat{Z}_t^X)$, with loss

$$\mathcal{L}_X = \frac{1}{K_X} \sum_{k=1}^{K_X} \left[1 - \cos\left(\widehat{X}_{t,k}, X^\star_{t,k}\right)\right], \qquad X \in \{V, R\}. \tag{4}$$

Self-conditioning. Self-conditioning lets each denoising step build on the decoder's own outcome and purpose estimates. A preliminary pass produces estimates $\bar{Z}_t^V$ and $\bar{Z}_t^R$, which we detach and re-noise as $Z_t^{X,\lambda} = (1-\lambda)\epsilon_X + \lambda\,\mathrm{sg}(\bar{Z}_t^X)$ for $X \in \{V, R\}$. Here, sg denotes stop-gradient. These inputs are processed jointly with action and intent rows.

3.5 Training Downstream Use of Intent

The decoder can read both recovered intent and the original VLA context B_t, so teacher alignment alone does not require it to use intent. We address this by controlling context access and comparing predictions made with matched and substituted states.

Context routing. During training, a binary gate α intermittently blocks direct context access for non-intent rows, while intent queries retain access. When $\alpha = 0$, action and grounding rows must read the intent pathway to obtain visual-language context. Grounding rows read intent and one another, while action rows also read grounding and native state/action inputs. Section A.5 details the routing for each backbone.

Intent-mismatch training. We further train predictions made with matched intent to be more accurate than those made with intent from another example. On updates with $\alpha = 0$, we compare decoder continuations from L_{tap} using matched intent or intent from another example, keeping the receiver's other states, inputs, and targets fixed. Their batch-averaged downstream losses define the mismatch objective:

$$\begin{aligned} \mathcal{D}_b &= \mathcal{L}_A^{(b)} + \lambda_V \mathcal{L}_V^{(b)} + \lambda_R \mathcal{L}_R^{(b)}, \quad b \in \{\mathrm{right}, \mathrm{swap}\}, \\ \mathcal{L}_{\mathrm{mis}} &= \max(0, m + \mathcal{D}_{\mathrm{right}} - \mathcal{D}_{\mathrm{swap}})\,. \end{aligned} \tag{5}$$

This objective encourages the matched state to yield lower downstream loss by a margin m.

Full training objective. We combine state alignment, prediction, and mismatch training:

$$\mathcal{L} = \mathcal{L}_A + \lambda_I \mathcal{L}_I + \mu(s)(\lambda_V \mathcal{L}_V + \lambda_R \mathcal{L}_R) + \lambda_{\mathrm{mis}} \mathcal{L}_{\mathrm{mis}}. \tag{6}$$

The schedule $\mu(s)$ gradually introduces auxiliary supervision over training steps s. Sections A.5 and A.6 give the training procedure and loss weights.

Deployment. At deployment, the policy recovers intent from current inputs and carries its own auxiliary states across denoising steps. We enable direct context access with $\alpha = 1$, also sampled during training. The teacher, target caches, alignment heads, and mismatch branch are removed. Intent queries and auxiliary rows remain within the action decoder, so the additional runtime cost comes from these states rather than teacher inference (Table 7).

4 Experiments

We first evaluate policy performance across simulation benchmarks and VLA backbones, then test whether the gains extend to real-world tasks with held-out objects and distractors. Controlled studies examine how target supervision and training mechanisms contribute to performance, what the recovered state encodes, and whether downstream decoding uses it.

Table 1: **Results on SimplerEnv-Bridge.** Success rates (%) on the four evaluated tasks. Controlled variants report mean ± sample standard deviation over three evaluation runs. Published baselines retain their reported values. Bold and underlining mark the best and second-best means per column.

Method	Spoon	Carrot	Stack	EP-Basket	Avg.
General-purpose VLA policies					
Octo-base (Octo Model Team et al., 2024)	12.5	8.3	0.0	43.1	16.0
Octo-small (Octo Model Team et al., 2024)	47.2	9.7	4.2	56.9	29.5
OpenVLA (Kim et al., 2025)	0.0	0.0	0.0	4.1	1.0
RoboVLMs (Li et al., 2026)	29.2	25.0	12.5	58.3	31.3
SpatialVLA (Qu et al., 2025)	16.7	25.0	29.2	**100.0**	42.7
π_0 (Black et al., 2025b)	46.7	38.7	42.7	39.3	41.8
π_0-FAST (Pertsch et al., 2025)	59.0	79.0	<u>65.0</u>	33.0	59.0
GR00T-N1.5 (NVIDIA GEAR Team, 2025)	30.0	28.0	16.0	42.7	29.2
Action abstraction and semantic reasoning					
LAPA (Ye et al., 2025)	70.8	45.8	54.2	58.3	57.3
UniVLA (Bu et al., 2025)	52.8	55.6	2.8	80.6	47.9
ECoT (Zawalski et al., 2025; Zhang et al., 2026)	40.2	11.7	0.0	28.4	20.1
Controlled backbone comparisons					
GR00T-N1.7 (NVIDIA, 2026)	<u>84.7</u> ±3.1	<u>79.3</u> ±9.9	57.3 ±8.1	36.0 ±6.0	64.3 ±3.3
GR00T-N1.7 + future supervision	81.3 ±2.3	73.3 ±4.2	56.7 ±3.1	60.7 ±4.2	<u>68.0</u> ±1.0
GR00T-N1.7 + INDI (Ours)	**88.7** ±1.2	**84.7** ±6.1	**69.3** ±5.0	<u>96.0</u> ±4.0	**84.7** ±0.8
Δ (Ours − baseline)	+4.0	+5.4	+12.0	+60.0	+20.4
$\pi_{0.5}$ (Black et al., 2025a)	78.0 ±3.5	72.7 ±7.6	32.0 ±3.5	26.7 ±4.2	52.3 ±2.3
$\pi_{0.5}$ **+ INDI (Ours)**	81.3 ±4.2	76.0 ±4.0	39.3 ±5.0	38.7 ±3.1	58.8 ±3.3
Δ (Ours − baseline)	+3.3	+3.3	+7.3	+12.0	+6.5

4.1 EXPERIMENTAL SETUP

Benchmarks. SimplerEnv-Bridge (Li et al., 2025b) covers four tabletop tasks, while RoboCasa Kitchen (Nasiriany et al., 2024) contains 24 household tasks spanning pick-and-place, articulated-object manipulation, and appliance interaction. Our real-world evaluation contains four tabletop tasks under standard scenes, held-out objects, and distractor conditions. Dataset, robot-platform, and evaluation-protocol details are provided in Section A.

Baselines. Our primary comparisons use GR00T-N1.7 (NVIDIA, 2026) and $\pi_{0.5}$ (Black et al., 2025a), with each baseline trained using the same demonstrations, optimization budget, and evaluation protocol as its corresponding INDI model. Our future-supervision control aligns intermediate decoder states with future-observation features in place of the teacher-intent target. Published benchmark results provide additional context, including GR00T checkpoints trained with 3,000 demonstrations per task as data-scale references on RoboCasa Kitchen. For real-world tasks, we compare against GR00T-N1.7 under the same rollout protocol. Descriptions and comparison settings for all baselines are provided in Section A.3.

Implementation details. We retain each backbone's optimizer, learning rate, and vision-language freezing configuration, and use the same INDI configuration across benchmarks unless stated otherwise. By default, we use $K_I = 8$, $K_R = 8$, and 16 visual rows per camera view, an intent-alignment tap at 50% of the decoder depth, margin $m = 0.05$, context-dropout rate 0.5, and Cosmos-Reason2-8B (NVIDIA, 2025a) as the teacher VLM. Full hyperparameters, target-construction details, and implementation choices are provided in Section A.

4.2 RESULTS ON SIMULATION BENCHMARKS

SimplerEnv-Bridge. INDI improves GR00T-N1.7 average success from 64.3% to 84.7%, a +20.4 pp gain with improvements on all four tasks (Table 1). The largest gain occurs on EP-Basket, from 36.0% to 96.0%; the remaining three tasks improve by 7.1 pp on average. INDI also exceeds future supervision (68.0%) by 16.7 pp.

Table 2: **Results on RoboCasa Kitchen.** Success rates (%) on the 24-task benchmark. G_n denotes n demonstrations per task. The G_{3000} checkpoints provide single-value data-scale references, while controlled G_{100} variants report mean ± sample standard deviation over three evaluation runs. Success rates are averaged within each task category, while Avg. is the macro-average across all 24 tasks. Bold marks the best mean in each column, and underline marks the runner-up.

Method	Pick-and Place	Open-or Close	Others	Avg.
GR00T-N1.6 (NVIDIA, 2025b) (G_{3000})	43.2	<u>81.0</u>	75.8	66.2
GR00T-N1.7 (NVIDIA, 2026) (G_{3000})	**53.0**	80.8	<u>79.0</u>	**70.8**
GR00T-N1.7 (G_{100})	39.4 ±1.0	75.9 ±3.4	76.7 ±0.9	64.1 ±1.6
GR00T-N1.7 + future supervision (G_{100})	47.0 ±4.4	80.3 ±2.3	72.1 ±2.0	65.8 ±0.8
GR00T-N1.7 + INDI (G_{100}, Ours)	<u>49.8</u> ±2.3	**82.8** ±1.3	**79.1** ±1.5	<u>70.3</u> ±1.7
Δ (Ours − baseline)	+10.4	+6.9	+2.4	+6.2
$\pi_{0.5}$ (Black et al., 2025a)	14.0 ±1.7	55.1 ±7.4	39.5 ±0.9	34.9 ±2.0
$\pi_{0.5}$ **+ INDI (Ours)**	15.3 ±1.4	56.1 ±3.6	53.5 ±2.4	41.4 ±1.4
Δ (Ours − baseline)	+1.3	+1.0	+14.0	+6.5

The improvement also transfers to $\pi_{0.5}$, increasing average success from 52.3% to 58.8%, a +6.5 pp gain, with improvements on every task. With the same demonstration budget, these gains show more effective learning from the available data on both backbones.

RoboCasa Kitchen. Table 2 reports controlled comparisons on the 24-task RoboCasa Kitchen benchmark. With GR00T-N1.7 trained on 100 demonstrations per task, INDI improves average success from 64.1% to 70.3%, with gains across all three task categories. The future-supervision variant reaches 65.8%, whereas the full method reaches 70.3%, an additional +4.5 pp gain. The improvement again transfers to $\pi_{0.5}$, increasing average success from 34.9% to 41.4%, including a +14.0 pp gain in Others. These controlled comparisons extend the observed gains from tabletop tasks to diverse household manipulation. The complete task-level results are reported in Table 25.

As a data-scale reference, INDI trained with 100 demonstrations per task reaches 70.3% average success, within 0.5 pp of the reported GR00T-N1.7 checkpoint trained with 3,000 demonstrations per task. Table 3 provides broader context from published methods with differing training data and evaluation protocols.

Table 3: **Published RoboCasa results (%).**

Method	Demos/task	Avg.
Base VLA policies		
GR00T-N1 (NVIDIA et al., 2025)	300	49.6
π_0 (Black et al., 2025b)	300	62.5
π_0-FAST (Pertsch et al., 2025)	300	63.6
GR00T-N1.5 (NVIDIA GEAR Team, 2025)	300	65.7
Future and video-based methods		
DreamGen (Jang et al., 2025)	300	57.6
DUST (Won et al., 2026)	300	58.5
Video Policy (Liang et al., 2025)	300	66.0
FLARE (Zheng et al., 2025b)	300	66.4
History and representation methods		
HAMLET (Koo et al., 2026)	300	66.4
RS-CL (Kim et al., 2026b)	300	<u>69.7</u>
Behavior-intent supervision		
INDI **(Ours)**	**100**	**70.3**

4.3 RESULTS ON REAL-WORLD BENCHMARKS

Table 4 reports success rates on the four real-world tasks. INDI improves the baseline from 71.0% to 76.0% under ID clean, and the gain persists under held-out objects (62.0% to 68.0%) and distractors (53.0% to 62.0%). The largest gain is under distractors, where INDI also retains a larger fraction of its ID-clean success (Section C.1). The gains are concentrated in the longer tasks. Averaged across the three conditions, Cross-Bin Stacking and Drawer Storage improve by 7.3 and 10.7 pp, respectively, whereas Threading and Basket Nesting improve by 4.7 and 4.0 pp.

Table 4: **Results on real-world tasks.** Success rates (%) over 50 trials per task and condition. ID clean uses training objects without distractors, held-out substitutes unseen objects with the same role, and distractors add unrelated scene items. Bold marks the better policy per condition.

	ID clean		Held-out		Distractors	
Task	Base	+ INDI	Base	+ INDI	Base	+ INDI
Threading	92.0	**96.0**	**86.0**	84.0	70.0	**82.0**
Basket Nesting	**94.0**	92.0	84.0	**90.0**	76.0	**84.0**
Cross-Bin Stacking	74.0	**80.0**	62.0	**72.0**	58.0	**64.0**
Drawer Storage	24.0	**36.0**	16.0	**26.0**	8.0	**18.0**
Average	71.0	**76.0**	62.0	**68.0**	53.0	**62.0**

The cumulative stage profiles in Table 5 locate the gains at different points in the two longer tasks. On Cross-Bin Stacking, both policies retrieve the base cube at 96.0%. The gap grows when retrieving the top cube (82.0% to 86.0%) and completing the stack (74.0% to 80.0%). On Drawer Storage, INDI improves grasping and transfer from 50.0% to 70.0%, with the advantage persisting through final closure (24.0% to 36.0%). Thus, higher success reflects better initial grasping on Drawer Storage and later-stage completion on Cross-Bin Stacking.

Table 5: **Stage-level completion.** Fraction of trials (%) reaching each stage under ID clean (cumulative).

Task	Stage	Base	+ INDI
Threading	Pole centered	100.0	100.0
	Ring grasped	96.0	**100.0**
	Inserted	92.0	**96.0**
Basket Nest.	Lifted	**100.0**	96.0
	Nested	**94.0**	92.0
Cross-Bin	Base retrieved	96.0	96.0
	Base centered	90.0	**92.0**
	Top retrieved	82.0	**86.0**
	Stacked	74.0	**80.0**
Drawer	Grasped	50.0	**70.0**
	Transferred	50.0	**70.0**
	Opened	40.0	**56.0**
	Placed inside	26.0	**38.0**
	Closed	24.0	**36.0**

4.4 ANALYSIS AND CONTROLLED STUDIES

We distinguish the contributions of teacher supervision and training for its use by varying them separately. Target comparisons retain routing and mismatch training, while mechanism comparisons hold the teacher targets fixed. We then probe what the state encodes, test its effect on execution, and measure deployment cost. Unless stated otherwise, controlled analyses use GR00T-N1.7 on a fixed SimplerEnv-Bridge evaluation run shared across all variants. Main results average three evaluation runs.

Table 6: **Controlled analyses on SimplerEnv-Bridge.** (a) reports supervision and capacity controls under the same backbone, demonstrations, budget, and evaluation run. Intent only retains routing and action-only mismatch training. Free latent removes both $\mathcal{L}_I$ and $\mathcal{L}_{\text{mis}}$. (b) reports original-task success after jointly editing task- or phase-discriminative coordinates of intent and grounding rows at blocks 16–24. Diagonal/off-diagonal task entries use same-/cross-task donors. All values are success rates (%).

Method	Spoon	Carrot	Stack	EP-Basket	Avg.
GR00T-N1.7	82.0	68.0	66.0	30.0	61.5
+ Groundings only	88.0	82.0	38.0	32.0	60.0
+ Future supervision	84.0	70.0	56.0	62.0	68.0
+ Free latent	78.0	76.0	48.0	26.0	57.0
+ Intent only	86.0	**84.0**	70.0	64.0	76.0
w/o $\mathcal{L}_I$	80.0	78.0	58.0	36.0	63.0
Neutral prompt	88.0	76.0	58.0	82.0	76.0
INDI **(Ours)**	**90.0**	78.0	**74.0**	**100.0**	**85.5**

(a) Supervision controls.

	Donor task				Donor phase		
Task	Spoon	Carrot	Stack	EP-Basket	Early	Late	Noise
Spoon	**88.0**	62.0	58.0	24.0	19.0	0.0	4.0
Carrot	50.0	**84.0**	56.0	32.0	6.0	18.0	0.0
Stack	48.0	42.0	**72.0**	36.0	4.0	4.0	0.0
EP-Basket	42.0	48.0	44.0	**94.0**	0.0	0.0	0.0
Avg. (diag/off-diag)	84.5 / 45.2				7.3	5.5	1.0

(b) Task and phase interventions.

How do supervision and training mechanisms contribute? Teacher-intent alignment improves performance beyond the gains from mismatch training. With mismatch retained, intent alignment raises success from 63.0% to 85.5% (+22.5 pp, Table 11). The prompt also matters: asking about

functional purpose instead of neutrally describing the same execution raises success from 76.0% to 85.5% (Table 22). Further controls support two target choices: extracting features from the teacher's input rather than its generated statement, and giving it the executed segment in addition to current inputs (Sections B.6 and B.7).

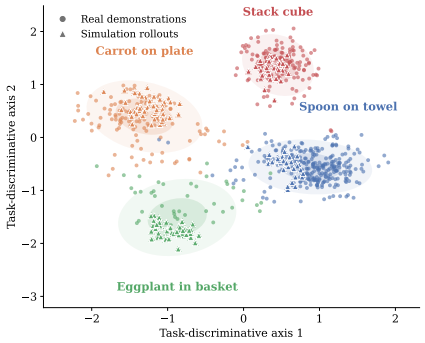
With the teacher targets fixed, removing routing, mismatch, or both reduces success to 70.0%, 74.0%, and 65.5% (Table 10). Thus, the same supervision is more effective when the decoder is trained to use the recovered state. Visual and textual predictions provide additional tasks for that state: adding both to intent-only training raises success from 76.0% to 85.5% (Table 6a).

What does the state encode, and is it used? Quantitative probes recover both task identity and temporal progress from the learned states (Table 13). Figure 3 visualizes this information through supervised task and phase projections of demonstration and rollout states. We also test whether objective-related similarity simply reflects similar actions. Among progress-matched pairs, states with the same objective and dissimilar actions are more similar than states with different objectives and similar actions (Section B.3 and Fig. 6).

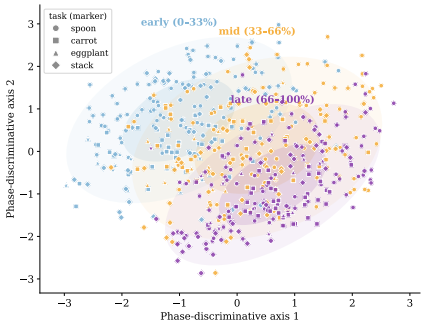
To test their use in execution, we replace task- or phase-related components jointly in the intent and grounding states with those of a selected donor, preserving the other components. We measure original-task success. Same-task interventions retain 84.5% success, compared with 45.2% for cross-task interventions and 1.0% under Gaussian corruption (Table 6b).

Forcing early or late phase content further tests the policy's sensitivity to progress while it executes the same instruction. Phase forcing reduces success to 7.3% for early content and 5.5% for late content. Selected rollouts show approach without grasping or attempted placement with an empty gripper (Figures 8 to 10).

Task substitutions reduce success according to the donor, while phase substitutions produce recognizable changes in execution stage. These results show that task- and progress-related content in the joint intent–grounding state affects action generation. Visual and textual readouts illustrate how replacing intent changes the predicted endpoint and purpose description (Section B.4).



(a) **Task structure.** A task-supervised projection of recovered states from demonstrations and simulation rollouts.



(b) **Progress structure.** A phase-supervised projection of the early, middle, and late thirds of each trajectory.

Figure 3: **Task and progress structure.** Supervised projections of task and temporal information.

What is the additional computational cost? The teacher runs offline, so runtime overhead comes from the added decoder states. For GR00T-N1.7, INDI increases parameters from 3.46B to 3.50B and per-query inference time from 56.1 to 61.5 ms. For $\pi_{0.5}$, the corresponding changes are 3.62B to 3.64B parameters and 24.4 to 29.7 ms. Training costs appear in Section A.7 and Table 7.

5 CONCLUSION

We presented INDI, which augments action imitation with teacher supervision about the local purpose of demonstrated behavior. The student recovers the teacher's representation from current inputs and learns to use it for action prediction. Controlled experiments support both choices: purpose-focused prompts improve performance over neutral descriptions, and routing and mismatch training improve results with the same teacher targets. INDI improves both evaluated VLA backbones on two simulation benchmarks and increases real-world task success. Probes and interventions further show that the recovered states carry task and progress information and affect downstream predictions. These findings show that demonstrations can supervise both actions and the intermediate states used to generate them, with the teacher needed only during training.

6 Reproducibility Statement

Section A documents the backbone integration, datasets, real-world platform, teacher prompt and target construction, implementation details, optimization settings, checkpoint selection, and computational resources. Appendix B details the controlled ablations, representation analyses, and closed-loop interventions, while Appendix C reports the full per-task results. An anonymous repository provides the implementation, training configurations, teacher-target construction scripts, dataset-preparation scripts, and evaluation code at https://anonymous.4open.science/r/intd-vla-ED63.

References

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. *arXiv preprint arXiv:2511.21631*, 2025.

Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, Badong Chen, and Shanghang Zhang. Latent reasoning VLA: Latent thinking and prediction for vision-language-action models. *arXiv preprint arXiv:2602.01166*, 2026. URL https://arxiv.org/abs/2602.01166.

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quan Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. RT-H: Action hierarchies using language. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.049. URL https://www.roboticsproceedings.org/rss20/p049.html.

Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2Act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision*, pp. 306–324, 2024. doi: 10.1007/978-3-031-73116-7_18.

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *International Conference on Learning Representations*, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025b. doi: 10.15607/RSS.2025.XXI.010.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to act anywhere with task-centric latent actions. In *Proceedings of Robotics: Science and Systems*, 2025.

Yi Chen, Yuying Ge, Hui Zhou, Mingyu Ding, Yixiao Ge, and Xihui Liu. Dial: Decoupling intent and action via latent world modeling for end-to-end vla. *arXiv preprint arXiv:2603.29844*, 2026.

Liaoyuan Fan, Zetian Xu, Chen Cao, Wenyao Zhang, Mingqi Yuan, and Jiayu Chen. AIM: Intent-aware unified world action modeling with spatial value maps. *arXiv preprint arXiv:2604.11135*, 2026.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24328–24346. PMLR, 2025. URL https://proceedings.mlr.press/v267/hu25g.html.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. ThinkAct: Vision-language-action reasoning via reinforced visual latent planning. In *Advances in Neural Information Processing Systems*, volume 38, 2025a.

Chi-Pin Huang, Yunze Man, Zhiding Yu, Min-Hung Chen, Jan Kautz, Yu-Chiang Frank Wang, and Fu-En Yang. Fast-ThinkAct: Efficient vision-language-action reasoning via verbalizable latent planning. *arXiv preprint arXiv:2601.09708*, 2026a. URL https://arxiv.org/abs/2601.09708.

Haifeng Huang, Xinyi Chen, Yilun Chen, Hao Li, Xiaoshen Han, Zehan Wang, Tai Wang, Jiangmiao Pang, and Zhou Zhao. Roboground: Robotic manipulation with grounded vision-language priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22540–22550, 2025b.

Renming Huang, Chendong Zeng, Wenjing Tang, Jintian Cai, Cewu Lu, and Panpan Cai. Mimic intent, not just trajectories. *arXiv preprint arXiv:2602.08602*, 2026b.

Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, Loïc Magne, Ajay Mandlekar, Avnish Narayan, You Liang Tan, Guanzhi Wang, Jing Wang, Qi Wang, Yinzhen Xu, Xiaohui Zeng, Kaiyuan Zheng, Ruijie Zheng, Ming-Yu Liu, Luke Zettlemoyer, Dieter Fox, Jan Kautz, Scott Reed, Yuke Zhu, and Linxi Fan. DreamGen: Unlocking generalization in robot learning through video world models. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 5170–5194. PMLR, 2025. URL https://proceedings.mlr.press/v305/jang25a.html.

Haobin Jiang, Jiangxing Wang, and Zongqing Lu. Discrete latent plans via semantic skill abstractions. In *The Thirteenth International Conference on Learning Representations*, 2025.

Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, Beomjun Kim, Byungjun Yoon, Changsung Jang, Daewon Choi, Dongsu Han, et al. RLDX-1 technical report. *arXiv preprint arXiv:2605.03269*, 2026a. URL https://arxiv.org/abs/2605.03269.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Taeyoung Kim, Jimin Lee, Myungkyu Koo, Dongyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. Contrastive representation regularization for vision-language-action models. In *International Conference on Machine Learning*, 2026b. URL https://arxiv.org/abs/2510.01711.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. Hamlet: Switch your vision-language-action model into a history-aware policy. In *International Conference on Learning Representations*, 2026.

Jinming Li, Yichen Zhu, Zhibin Tang, Junjie Wen, Minjie Zhu, Xiaoyu Liu, Chengmeng Li, Ran Cheng, Yaxin Peng, Yan Peng, and Feifei Feng. CoA-VLA: Improving vision-language-action models via visual-text chain-of-affordance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9759–9769, 2025a.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

Xinghang Li, Peiyan Li, Long Qian, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Xinlong Wang, Di Guo, Tao Kong, Hanbo Zhang, and Huaping Liu. What matters in building vision–language–action models for generalist robots. *Nature Machine Intelligence*, 8:158–172, 2026. doi: 10.1038/s42256-025-01168-7.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728. PMLR, 2025b. URL https://proceedings.mlr.press/v270/li25c.html.

Shijie Lian, Bin Yu, Xiaopeng Lin, Zhaolong Shen, Laurence Tianruo Yang, Yurun Jin, Haishan Liu, Changti Wu, Hang Yuan, Cong Huang, and Kai Chen. Intentvla: Short-horizon intent modeling for aliased robot manipulation. *arXiv preprint arXiv:2605.14712*, 2026.

Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. Video generators are robot policies. *arXiv preprint arXiv:2508.00795*, 2025.

Yihan Lin, Jiawei He, Shifeng Bao, Chen Zhao, Yang Li, Xiaobo Wang, Yan Wang, Cheng Chi, and Jing Zhang. JEPA-WAM: Learning vision-language-action policies with joint-embedding world modeling. *arXiv preprint arXiv:2608.09381*, 2026. URL https://arxiv.org/abs/2608.09381.

Jiahao Liu, Zhongpu Xia, Shuai Tian, Huangrui Li, Yuhang Zheng, Ning Ma, Xin Fu, Xiaotian Liu, Jing Li, Yixian Li, ShangQing Zhou, Zebin Xing, Linbo Wang, Chaoyue Li, Haoran Li, and Dongbin Zhao. WALA learning executable latent actions from action-labeled demonstrations and action-free videos. *arXiv preprint arXiv:2607.11397*, 2026. URL https://arxiv.org/abs/2607.11397.

Hao Luo, Wanpeng Zhang, Yicheng Feng, Sipeng Zheng, Haiweng Xu, Chaoyi Xu, Ziheng Xi, Yuhui Fu, and Zongqing Lu. Being-H0.7: A latent world-action model from egocentric videos. *arXiv preprint arXiv:2605.00078*, 2026.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1113–1132. PMLR, 2020.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024.

NVIDIA. Cosmos-Reason2: Physical ai common sense and embodied reasoning models. GitHub repository, 2025a. URL https://github.com/nvidia-cosmos/cosmos-reason2. Released December 19, 2025.

NVIDIA. Gr00t n1.6: An improved open foundation model for generalist humanoid robots. NVIDIA Research project page, December 2025b. URL https://research.nvidia.com/labs/gear/gr00t-n1_6/.

NVIDIA. Nvidia isaac gr00t n1.7: A foundation model for generalist robots. GitHub repository, 2026. URL https://github.com/NVIDIA/Isaac-GR00T. Early access release.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

NVIDIA GEAR Team. GR00T N1.5: An improved open foundation model for generalist humanoid robots. NVIDIA Research project page, 2025. URL https://research.nvidia.com/labs/gear/gr00t-n1_5/.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090.

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.

Physical Intelligence. openpi: Open-source models and packages for robotics. https://github.com/Physical-Intelligence/openpi, 2025.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, et al. $\pi_{0.7}$: a Steerable Generalist Robotic Foundation Model with Emergent Capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Jiayuan Gu, Zhigang Wang, Yan Ding, Bin Zhao, Dong Wang, and Xuelong Li. Spatialvla: Exploring spatial representations for visual-language-action models. In *Robotics: Science and Systems*, 2025.

Qwen Team. Qwen3.5-9B. Hugging Face model repository, 2026. URL https://huggingface.co/Qwen/Qwen3.5-9B.

Davood Soleymanzadeh, Kaidi Zhang, Zhiyuan Zhang, Bihao Zhang, Xiao Liang, Yu She, and Minghui Zheng. PHR-VLA: Planning horizon reasoning for vision-language-action models. *arXiv preprint arXiv:2608.27609*, 2026. URL https://arxiv.org/abs/2608.27609.

Theodore Sumers, Kenneth Marino, Arun Ahuja, Rob Fergus, and Ishita Dasgupta. Distilling internet-scale vision-language models into embodied agents. In *International Conference on Machine Learning*, 2023.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. VLA-JEPA: Enhancing vision-language-action model with latent world model. *arXiv preprint arXiv:2602.10098*, 2026.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *International Conference on Learning Representations*, 2025.

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, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1723–1736. PMLR, 2023.

John Won, Kyungmin Lee, Huiwon Jang, Dongyoung Kim, and Jinwoo Shin. Dual-stream diffusion for world-model augmented vision-language-action model. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2510.27607.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024.

Yifu Xiong, Wenhao Yu, Jiaxuan Lin, Bojun Zou, Jiahao Li, Lu Zhang, Yanyong Zhang, and Jianmin Ji. CAC-VLA: Context-gated action conditioning for vision-language-action models. *arXiv preprint arXiv:2607.04816*, 2026. URL https://arxiv.org/abs/2607.04816.

Mengda Xu, Zhenjia Xu, Yinghao Xu, Cheng Chi, Gordon Wetzstein, Manuela Veloso, and Shuran Song. Flow as the cross-domain manipulation interface. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2475–2499. PMLR, 2025. URL https://proceedings.mlr.press/v270/xu25a.html.

Xiaoxu Xu, Hao Li, Jinhui Ye, Yilun Chen, Jia Zeng, Xinyi Chen, Linning Xu, Dahua Lin, Weixin Li, and Jiangmiao Pang. FutureVLA: Joint visuomotor prediction for vision-language-action model. *arXiv preprint arXiv:2603.10712*, 2026.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *International Conference on Learning Representations*, 2025.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, 2025.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3157–3181. PMLR, 2025.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, Li Yi, Wenjun Zeng, and Xin Jin. DreamVLA: A vision-language-action model dreamed with comprehensive world knowledge. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yuchi Zhang, Churui Sun, Shiqi Liang, Diyuan Liu, Chao Ji, Wei-Nan Zhang, and Ting Liu. Bridging scale discrepancies in robotic control via language-based action representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18809–18817, 2026.

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

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. In *International Conference on Learning Representations*, 2025a.

Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loïc Magne, Avnish Narayan, You Liang Tan, Guanzhi Wang, Qi Wang, Jiannan Xiang, Yinzhen Xu, Seonghyeon Ye, Jan Kautz, Furong Huang, Yuke Zhu, and Linxi Fan. FLARE: Robot learning with implicit world modeling. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3952–3971. PMLR, 2025b. URL https://proceedings.mlr.press/v305/zheng25a.html.

Zhide Zhong, Junfeng Li, Junjie He, Haodong Yan, Xin Gong, Guanyi Zhao, Yingjie Cai, Jiantao Gao, Xu Yan, Bingbing Liu, Yingcong Chen, Liuqing Yang, and Haoang Li. DualCoT-VLA: Visual-linguistic chain of thought via parallel reasoning for vision-language-action models. *arXiv preprint arXiv:2603.22280*, 2026. URL https://arxiv.org/abs/2603.22280.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A EXPERIMENTAL DETAILS

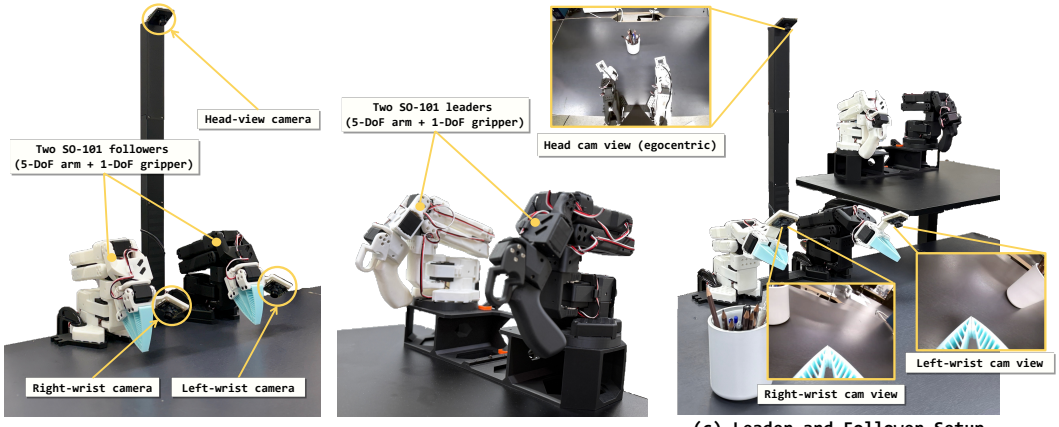


Figure 4: **Real-robot teleoperation and evaluation platform.** We specify the leader and follower robot arm setups, consisting of 5-DoF arms with 1-DoF grippers, along with the multi-camera views.

A.1 MODEL DETAILS

Backbone VLAs. We instantiate INDI on two flow-matching VLA backbones, GR00T-N1.7-3B (NVIDIA, 2026) and $\pi_{0.5}$ (Black et al., 2025a; Physical Intelligence, 2025), using their official implementations. For both models, we freeze the pretrained vision-language module and fine-tune the action decoder together with the newly introduced INDI parameters.

INDI **integration.** For GR00T-N1.7, we append self-conditioned visual and textual grounding rows and clean intent queries Q^I to the pretrained DiT token stream. The grounding rows follow the backbone's time-conditioned normalization, while the intent queries use standard layer normalization and are never noised. The recovered intent is supervised at block 16 of 32. For $\pi_{0.5}$, the action-expert suffix is organized as

$$[A_t^\lambda, Z_t^{V,\lambda}, Z_t^{R,\lambda}, Q^I],$$

where A_t^λ is the noised action sequence, $Z_t^{V,\lambda}$ and $Z_t^{R,\lambda}$ are self-conditioned visual and textual grounding inputs, and Q^I contains the clean intent queries. The proprioceptive state is represented by tokens in the frozen prefix, following the native $\pi_{0.5}$ input format. The recovered intent is supervised at layer 9 of 18. We apply the same asymmetric information-flow pattern during training and sampling. Intent queries attend to the frozen VLA context, proprioceptive state, and one another while remaining isolated from the noised action and grounding rows. Visual and textual grounding rows attend to the recovered intent and one another, whereas action rows attend to the intent together with both grounding streams. For $\pi_{0.5}$, we cache the frozen prefix keys and values during training.

Trainable and frozen components. The vision-language modules, including their visual encoders, remain frozen for both backbones. We optimize the complete action decoder together with the added intent and grounding parameters. These parameters include intent-query embeddings, positional and camera-view embeddings, and three-layer SiLU MLPs for encoding and projecting the target representations. The visual target has width $d_V = 2048$, the teacher targets have width $d_T = 4096$, and the added representations are mapped to the backbone-specific decoder width d_D. Checkpoint-tensor counting gives 3.455B parameters for the GR00T-N1.7 baseline and 3.502B for GR00T-N1.7+INDI, corresponding to 46.4M additional parameters or a 1.3% increase. For $\pi_{0.5}$, the corresponding counts are 3.617B and 3.640B, giving 23.1M additional parameters or a 0.64% increase.

Model dimensions. Unless stated otherwise, we use $K_I = 8$ intent queries, $K_R = 8$ textual grounding rows, and 16 visual grounding rows per camera. The total number of visual rows is $K_V = 16$ for single-camera Bridge and $K_V = 48$ for three-camera RoboCasa Kitchen. We use an intent-mismatch margin of $m = 0.05$ and a context-dropout probability of 0.5.

A.2 DATASETS

SimplerEnv-Bridge. We evaluate on SimplerEnv-Bridge (Li et al., 2025b), a simulation benchmark for evaluating real-world manipulation policies trained on BridgeData V2 (Walke et al., 2023). We use the LeRobot conversion of BridgeData V2, which contains 53,192 episodes and 1,893,026 frames recorded at 5 Hz. Each sample contains a primary RGB observation at 256×256 resolution, an 8-dimensional robot state, and a 7-dimensional action. We evaluate four WidowX manipulation tasks: spoon on towel, carrot on plate, stack cube, and put eggplant in basket. Our controlled GR00T-N1.7 and $\pi_{0.5}$ comparisons use three independent evaluation runs with 50 episodes per task in each run. We report the mean and sample standard deviation of the three run-level success rates. Published baseline results follow their original evaluation protocols.

RoboCasa Kitchen. RoboCasa Kitchen (Nasiriany et al., 2024) is a simulation benchmark for household manipulation in diverse kitchen environments. We use its machine-generated dataset with 100 demonstrations for each of 24 tasks, giving 2,400 episodes and 689,595 frames recorded at 20 Hz. Each sample contains left, right, and wrist RGB observations at 256×256 resolution, a 53-dimensional robot state, and a 12-dimensional action. The benchmark contains 8 pick-and-place tasks, 6 open-or-close tasks, and 10 additional appliance and interaction tasks. We use three independent evaluation runs with 50 episodes per task and run for both GR00T-N1.7 and $\pi_{0.5}$. Category and overall averages are computed within each run and then summarized by their mean and sample standard deviation across the three runs.

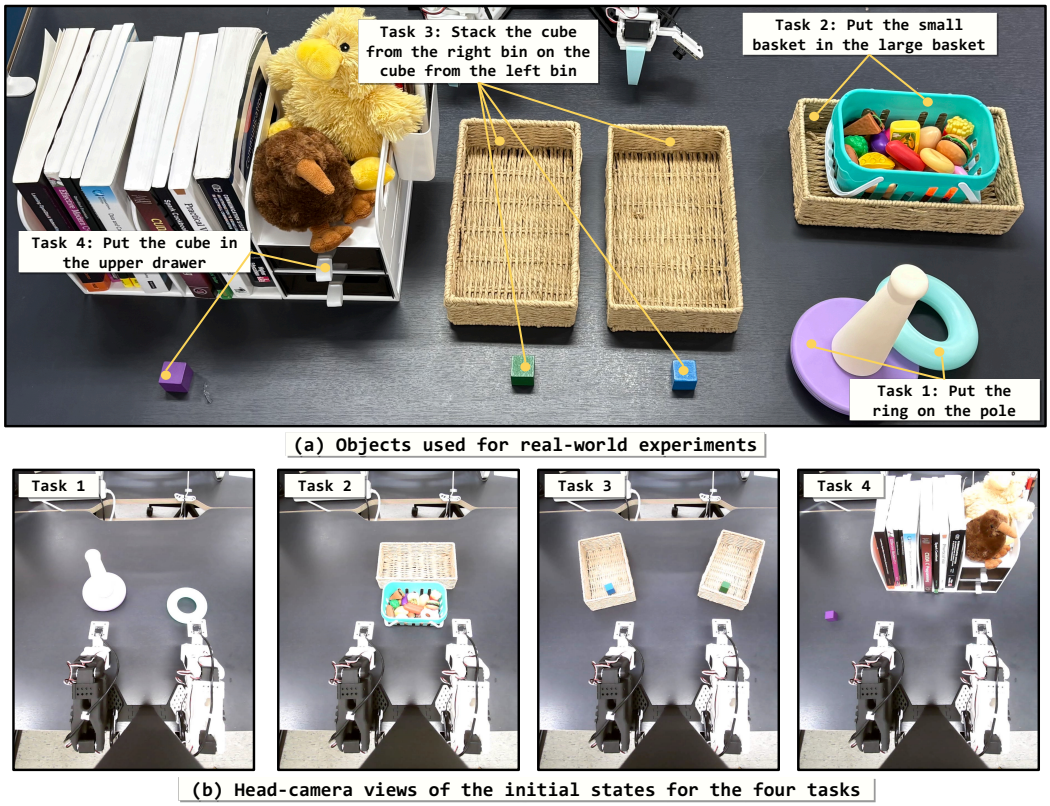


Figure 5: **Real-world task suite.** Four tabletop tasks used for real-world experiments.

Real-world platform. As shown in Figure 4, we use a teleoperation setup consisting of SO-101 robot arms for both the leader and follower, equipped with 1-DoF grippers. Three camera views are provided: an egocentric view mounted on the head structure and two local views on the wrists.

On this platform, we design four tabletop tasks as illustrated in Figure 5, each targeting a distinct capability and ordered by horizon length. For each task, we collect 100 demonstrations and train a single multitask policy per method on identical data.

- *Threading* (bimanual) has the left arm bring a pole to the center, after which the right arm inserts a specified ring onto it, testing precise alignment.
- *Basket Nesting* (bimanual) lifts a small basket with both arms into a larger one, testing simultaneous cooperation on a shared object that a single arm cannot lift.
- *Cross-Bin Stacking* (bimanual) has the left arm place a cube from the left bin at the center, after which the right arm stacks a cube from the right bin on top.
- *Drawer Storage* (bimanual) has the left arm pass an object to the right arm, which then opens the specified drawer of a two-tier unit, places the object inside, and closes it.

A.3 BASELINES

We briefly describe the baseline methods included in our evaluations.

- **Octo** (Octo Model Team et al., 2024) is a transformer-based generalist policy trained on cross-embodiment robot data with a diffusion-based action head.
- **OpenVLA** (Kim et al., 2025) fine-tunes a pretrained vision-language model to autoregressively predict discretized action tokens.
- **RoboVLMs** (Li et al., 2026) studies the integration of pretrained vision-language models with continuous and discrete action-prediction architectures.
- **SpatialVLA** (Qu et al., 2025) incorporates explicit 3D spatial information through Ego3D positional encoding and adaptive action grids.
- π_0 (Black et al., 2025b) combines a pretrained vision-language prefix with a flow-matching action expert for continuous action generation.
- π_0**-FAST** (Pertsch et al., 2025) predicts actions autoregressively using frequency-space action tokenization.
- $\pi_{0.5}$ (Black et al., 2025a) extends the π_0 family toward open-world generalization and serves as our second backbone.
- **GR00T-N1** (NVIDIA et al., 2025) combines a vision-language module with a flow-matching diffusion-transformer action decoder.
- **GR00T-N1.5** (NVIDIA GEAR Team, 2025) improves the architecture, training data, and post-training procedure of GR00T-N1.
- **GR00T-N1.6** (NVIDIA, 2025b) further updates the GR00T pretraining mixture and post-training recipe.
- **GR00T-N1.7** (NVIDIA, 2026) is our primary backbone.
- **LAPA** (Ye et al., 2025) learns discrete latent actions between video frames and pretrains a vision-language model to predict these representations before robot-action fine-tuning.
- **UniVLA** (Bu et al., 2025) learns task-centric latent actions from cross-embodiment videos and decodes the predicted latent actions into embodiment-specific robot trajectories.
- **ECoT** (Zawalski et al., 2025) generates embodied reasoning traces before predicting robot actions.
- **DreamGen** (Jang et al., 2025) generates synthetic robot trajectories with a video world model and recovers corresponding pseudo-actions.
- **DUST** (Won et al., 2026) jointly predicts future observations and actions using interacting diffusion streams with decoupled flow-matching objectives.
- **Video Policy** (Liang et al., 2025) combines behavior-video generation and action prediction within an end-to-end robot policy.
- **FLARE** (Zheng et al., 2025b) aligns intermediate action-decoder representations with latent features of future observations.
- **HAMLET** (Koo et al., 2026) incorporates observation history through moment tokens and a temporal memory module.

- **RS-CL** (Kim et al., 2026b) regularizes vision-language representations with a contrastive objective supervised by distances between robot proprioceptive states.

Our controlled comparisons use GR00T-N1.7 and $\pi_{0.5}$ baselines trained with the same demonstrations, optimization budgets, and evaluation protocols as their corresponding INDI models. Moreover, future supervision replaces the teacher-intent target with FLARE-style alignment of mid-depth decoder representations to latent features of the future observation, under the same backbone and training budget. The exact configurations of the supervision and capacity controls are provided in Section B.1.

A.4 TARGET CONSTRUCTION

Teacher inputs and prompt. We use the frozen Cosmos-Reason2-8B model (NVIDIA, 2025a) as the teacher. For each H-step demonstrated behavior segment $\mathcal{W}_t$, the teacher receives the current observation, language instruction, a coarse textual action summary $c_A(\mathcal{W}_t)$, and the corresponding execution video. The teacher processes this multimodal evidence under a functional-intent prompt and generates a functional-purpose statement describing what the executed behavior accomplishes under the instruction. The prompt asks the teacher to reason in at most four sentences about the state change occurring during the segment and how it advances the instruction. It then requests exactly one sentence of approximately 30 to 50 tokens that describes the object-level state change or immediate local subgoal. Low-level control values, camera motion, references to the video, and additional labels or explanations are explicitly excluded. We use the same prompt template for Bridge and RoboCasa Kitchen, substituting $n = 8$ and $n = 16$ video frames, respectively.

Prompt for Teacher Target Construction

System Prompt
You are a robot manipulation reasoning model. You will see a short video of an approximately 1.6-second segment from a longer robot task, a short multi-step segment that may include contact, object motion, or clear setup for the next contact, plus a coarse summary of the commanded robot motion.
First write a brief reasoning block inside `<think>...</think>`, using at most four sentences. In the reasoning, infer what changes over this segment: which object or relation is acted on, what state transition occurs, and why this segment advances the episode instruction.
Immediately after `</think>`, write exactly one sentence of roughly 30–50 tokens. This sentence is the segment's functional intent: the object-level outcome that the behavior accomplishes or clearly progresses toward. Describe the intended object-level state change, not low-level motion, camera appearance, or the video itself. State only changes that are visible or strongly implied by the segment. If contact or object motion has not happened yet, describe the immediate setup or alignment achieved by the segment.
Do not mention "the video," "the clip," "frames," "I," or "the robot intends." Do not add a label, bullet, prefix, or explanation after `</think>`. Output only the one sentence after `</think>`.

User Prompt
Episode instruction: `{instruction}`
Coarse commanded motion over this segment: `{action_summary}`
The video frames show an approximately 1.6-second multi-step segment with `{n}` frames.
Identify the functional intent of this behavior segment by stating the object-level outcome it accomplishes or clearly progresses toward under the episode instruction.

Action and video preprocessing. We convert each continuous action segment into a compact textual representation before providing it to the teacher. For each action dimension, we compute dataset-specific percentile boundaries from the training set and map the continuous values to coarse discrete motion bins. The resulting sequence of bucketed values is serialized as a short description of the translational, rotational, and gripper behavior over the segment. The gripper dimension is interpreted using the open and close convention of each dataset rather than assuming a shared sign convention across embodiments. This representation provides the teacher with the overall motion pattern while avoiding long sequences of raw floating-point values. The execution video covers the same temporal interval as the summarized action segment. On Bridge, we use the 8 frames corresponding to an 8-step segment recorded at 5 Hz. On RoboCasa Kitchen, each target covers a 32-step segment recorded at 20 Hz, from which we uniformly sample 16 frames. Both therefore represent an approximately 1.6-second behavior window. The current observation, action segment, and video

clip are matched using their episode identifier and segment start index, ensuring that all teacher inputs describe the same executed behavior.

Intent and textual-purpose targets. Let $\mathcal{E}_t$ denote the teacher-input span containing the current observation, instruction, coarse action summary, and execution video. Let $\mathcal{S}_t$ denote only the final functional-purpose statement generated after the reasoning span. We extract teacher hidden states from the input span $\mathcal{E}_t$ and the final-statement span $\mathcal{S}_t$, keeping their distinct positions in the teacher sequence. We construct the intent target $I_t^\star$ from the layer-18 hidden states over $\mathcal{E}_t$, corresponding to the midpoint of the teacher. Given the input-span hidden states $H_{\mathcal{E}_t}^{(18)}$, we partition the token sequence into $K_I = 8$ approximately equal contiguous regions and average the hidden states within each region:

$$I_{t,k}^\star = \frac{1}{|\mathcal{G}_{I,k}|} \sum_{i \in \mathcal{G}_{I,k}} H_{\mathcal{E}_t,i}^{(18)}, \qquad k = 1, \ldots, K_I.$$

This preserves the coarse ordering of the multimodal teacher-input representation while producing a fixed number of target slots. These regions are defined by token position, so their boundaries need not coincide with modality boundaries. Individual slots are not assigned to specific modalities or semantic units. We construct the textual-purpose target $R_t^\star$ from the final-layer hidden states over $\mathcal{S}_t$. The statement tokens are likewise partitioned into $K_R = 8$ contiguous regions and mean-pooled:

$$R_{t,k}^\star = \frac{1}{|\mathcal{G}_{R,k}|} \sum_{i \in \mathcal{G}_{R,k}} H_{\mathcal{S}_t,i}^{(\text{final})}, \qquad k = 1, \ldots, K_R.$$

Both targets have hidden width $d_T = 4096$. Thus, $I_t^\star$ is a purpose-prompted representation of the input evidence, whereas $R_t^\star$ represents the generated final statement. The input-span target can encode scene, object, and motion cues together with information about the segment's local objective.

Generated-response intent target for the target-source ablation. We additionally construct an alternative intent target from the teacher-generated response. Let $\mathcal{Y}_t$ denote the complete response span generated after processing $\mathcal{E}_t$, including both the reasoning block and the final functional-purpose statement. Using the same teacher layer as the default intent target, we partition the hidden states over $\mathcal{Y}_t$ into $K_I = 8$ approximately equal contiguous regions and mean-pool each region:

$$I_{t,k}^{\star,\text{resp}} = \frac{1}{|\mathcal{G}_{\text{resp},k}|} \sum_{i \in \mathcal{G}_{\text{resp},k}} H_{\mathcal{Y}_t,i}^{(18)}, \qquad k = 1, \ldots, K_I.$$

This alternative preserves the same target width and number of slots as the default evidence-side target, while replacing the representation formed over the multimodal evidence with one formed over the teacher's complete generated interpretation.

Visual-outcome target. We construct the visual-outcome target $V_t^\star$ from the observation at the endpoint of the behavior segment. Specifically, the endpoint is o_{t+8} for Bridge and o_{t+32} for RoboCasa Kitchen, matching the temporal window used for the corresponding teacher target. We encode this observation using the frozen visual encoder of Cosmos-Reason2-2B. The encoder produces 64 regional features with hidden width $d_V = 2048$, arranged as an 8×8 raster grid. We average each non-overlapping 2×2 region to obtain a 4×4 grid and flatten it in raster order, producing 16 visual target slots per view. For multi-camera observations, each view is encoded and pooled independently. Bridge uses one camera and therefore produces 16 visual target slots. RoboCasa Kitchen uses left, right, and wrist cameras and therefore produces 48 slots in total. Samples that do not contain a valid endpoint observation near an episode boundary are excluded from the visual-alignment loss. All teacher and visual targets are generated once offline and reused across policy-training runs.

A.5 Implementation

Decoder augmentation. We preserve the native state and action representations of each backbone and append self-conditioned visual grounding rows $Z_t^{V,\lambda}$, self-conditioned textual grounding rows $Z_t^{R,\lambda}$, and clean intent queries Q^I to the action-decoder sequence. For GR00T-N1.7, the added rows are incorporated into the diffusion-transformer stream. For $\pi_{0.5}$, they are appended to the action-expert suffix following the frozen vision-language prefix. All added rows use learned positional embeddings, and visual rows additionally use camera-view embeddings. For $\pi_{0.5}$, we cache the frozen prefix keys and values during training.

Asymmetric information flow. Intent queries attend to the robot state, frozen VLA context, and one another, but not to the noised action or grounding rows. Visual and textual grounding rows attend to the recovered intent and one another, while action rows also retain access to their native state and action representations. Direct context access from action and grounding rows is controlled by a binary gate α, whereas intent queries always retain access to the VLA context. During training, we sample $\alpha \sim \mathrm{Bernoulli}(0.5)$, producing updates with direct context access and updates that route context through the intent queries. For $\pi_{0.5}$, this gate is applied to the image and language prefix tokens, while access to the state representation remains available. GR00T-N1.7 applies the gate to the cross-attention output for non-intent rows, after attention normalization. This avoids fully masked cross-attention rows when $\alpha = 0$. For $\pi_{0.5}$, the binary gate is implemented by masking the corresponding prefix keys before softmax, with state-token access preserved. We retain the asymmetric decoder mask at deployment and set $\alpha = 1$, a condition already sampled during training.

Intent alignment. We extract the contextualized intent-query states $H_{I,t}^{L_{\mathrm{tap}}}$ at the midpoint of the decoder, using block 16 for GR00T-N1.7 and layer 9 for $\pi_{0.5}$. A three-layer SiLU MLP projects each intent state to the teacher feature space. The projected rows are aligned one-to-one with the eight teacher intent targets $I_t^\star$ using cosine distance, with stop-gradient applied to the targets. The decoder-space states $H_{I,t}^{L_{\mathrm{tap}}}$, rather than their projected teacher-space representations, are used as the recovered intent by the remaining decoder layers.

Intent-mismatch training. For updates with $\alpha = 0$, we construct mismatched intent states by cyclically shifting the recovered intent representations within the batch. For a local batch of size $N > 1$, sample i receives the intent from sample $(i+1) \bmod N$. Starting from the intent-alignment layer, the matched and swapped continuations share the same decoder blocks and prediction heads and are evaluated together as a doubled batch. The receiver's other hidden states and its action, visual, and textual targets are unchanged. For each branch $b \in \{\mathrm{right}, \mathrm{swap}\}$, we form

$$\mathcal{D}_b = \mathcal{L}_A^{(b)} + \lambda_V \mathcal{L}_V^{(b)} + \lambda_R \mathcal{L}_R^{(b)},$$

using the masked batch-mean action loss and valid-target mean cosine losses. We optimize

$$\mathcal{L}_{\mathrm{mis}} = \max\left(0, m + \mathcal{D}_{\mathrm{right}} - \mathcal{D}_{\mathrm{swap}}\right),$$

with $m = 0.05$. The hinge is applied to the aggregated branch losses, rather than averaged over per-sample hinges. Both branches use the full λ_V and λ_R weights, without the auxiliary-loss warmup factor. The entire swapped starting state is detached, including the receiver's retained rows and the donor intent. Gradients still flow through the downstream decoder blocks and prediction heads in the swapped continuation. Detachment at the branch input does not freeze these weights. The mismatch term is inactive for $\alpha = 1$ and skipped for local batches of size one. Cyclic shifting ensures a different example when $N > 1$, but does not ensure a semantically different local objective.

Grounding self-conditioning and deployment. The intent target $I_t^\star$ supervises intermediate query states and is not supplied to those queries as an input. For visual and textual self-conditioning, a no-gradient forward pass first estimates decoder-space grounding states. The estimates are detached and re-noised with the same base noise and sampled flow time to form the inputs to the gradient-bearing prediction pass. The training sequence therefore comprises a no-gradient estimate, a main prediction pass, and an additional decoder continuation from the alignment layer on mismatch updates. The frozen VLA context is reused, including cached prefix keys and values for $\pi_{0.5}$. During inference, the visual and textual states are carried across denoising steps using the same self-conditioning convention. The teacher VLM, target caches, alignment projections, and intent-mismatch branch are removed at deployment. The policy receives the original VLA inputs and retains the intent queries and auxiliary prediction rows within its action decoder. Equations in the main text use a noise-to-data flow time λ. GR00T-N1.7 follows this convention, while $\pi_{0.5}$ uses native time $\tau = 1 - \lambda$ and the corresponding reversed velocity target. Action and auxiliary inputs use the same noise level within each backbone.

A.6 Training Details

Optimization and learning-rate schedules. We follow the official fine-tuning implementations of GR00T-N1.7 (NVIDIA, 2026) and $\pi_{0.5}$ (Black et al., 2025a; Physical Intelligence, 2025). For each backbone, the baseline and INDI variant use the same benchmark-specific training budget. For

GR00T-N1.7, we use AdamW with a learning rate of 10^{-4}. We train for 60,000 steps with a global batch size of 480 on RoboCasa Kitchen and for 20,000 steps with a global batch size of 1,536 on Bridge. For $\pi_{0.5}$, we use AdamW with a maximum gradient norm of 1.0 and linearly warm up the learning rate for 1,000 steps to 5×10^{-5}, which is held constant for the remainder of training. We use the same benchmark-specific batch sizes and training steps as in the corresponding GR00T-N1.7 experiments. All models are trained with bfloat16 computation, while loss values are accumulated in float32. The global batch sizes are distributed across three GPUs without gradient accumulation.

Loss configuration. We set $\lambda_I = \lambda_V = \lambda_R = 0.5$, $\lambda_{\text{mis}} = 0.1$, and the intent-mismatch margin to $m = 0.05$. The weights of the visual and textual grounding losses are linearly increased from zero to their final values during the first 1,000 training steps. The action and intent-alignment losses are active from the beginning of training. We use a context-dropout probability of 0.5.

Loss reduction. Action MSE averages squared velocity errors over the action entries selected by the backbone's loss mask, excluding padded entries. Cosine losses average valid target-vector comparisons across the batch, applying sample- or view-level masks before reduction. Teacher targets are detached in all alignment losses. The main-text alignment equations show the per-segment means with masking omitted for readability.

Backbone-specific training configurations. For both backbones, we freeze the pretrained vision-language module and optimize the action decoder together with the newly introduced INDI parameters. The teacher targets are defined over an approximately 1.6-second behavior window, corresponding to $H = 8$ steps on Bridge and $H = 32$ steps on RoboCasa Kitchen. For GR00T-N1.7, the decoder retains its native 40-step output sequence, while the action and auxiliary losses are applied to the benchmark-valid segment of 8 steps on Bridge and 32 steps on RoboCasa Kitchen. For $\pi_{0.5}$, we retain its native 10-step action horizon and mask the action loss to the valid action dimensions, excluding dimensions introduced only by zero padding. The teacher behavior window is defined independently of the backbone-native policy horizon. We cache the keys and values of the frozen $\pi_{0.5}$ prefix during training.

Checkpoint selection. On RoboCasa Kitchen, we evaluate the final checkpoint obtained after 60,000 training steps. On Bridge, we evaluate checkpoints at 5,000, 10,000, 15,000, and 20,000 steps using two validation rollout seeds and apply the same selection procedure to both backbones.

Controlled ablation protocol. The mechanism, policy-level teacher, and intent-target-source ablations use GR00T-N1.7 on SimplerEnv-Bridge. Unless stated otherwise, all variants use the same demonstrations, student architecture, optimization budget, checkpoint-selection procedure, and fixed evaluation run as the controlled analyses in Section 4.4. Only the component or target source identified by each ablation is changed.

A.7 COMPUTATIONAL RESOURCES

All experiments are conducted on a single node with three NVIDIA B200 GPUs, each with 183 GB of memory. Teacher targets are generated once offline and reused across all subsequent policy-training runs. We summarize parameter counts, inference time per policy query, and benchmark-specific training time in Table 7. All inference-time measurements are obtained on a single NVIDIA B200 GPU.

Table 7: **Modification cost across VLA backbones.**

Method	Parameters	Inference time (ms)	Bridge training (h)	RoboCasa training (h)
GR00T-N1.7	3.455B	56.1 (1.00×)	∼17	∼17
+INDI	3.502B (+46.4M, +1.3%)	61.5 (1.10×)	∼19	∼19
$\pi_{0.5}$	3.617B	24.4 (1.00×)	∼23	∼23
+INDI	3.640B (+23.1M, +0.64%)	29.7 (1.21×)	∼29	∼26

B Additional Ablations and Diagnostics

Unless stated otherwise, the following controlled analyses use GR00T-N1.7 on a fixed SimplerEnv-Bridge evaluation run shared across all variants. The main benchmark tables separately aggregate three evaluation runs. This section details the controls and diagnostic protocols underlying Section 4.4, followed by full quantitative analyses and additional readouts.

B.1 Supervision and Capacity Controls

We compare complete training configurations to examine how supervision and latent capacity contribute to policy performance. All variants use the same GR00T-N1.7 backbone, Bridge demonstrations, optimization budget, and evaluation protocol. The groundings-only control retains the visual and textual grounding streams, $\mathcal{L}_V$, $\mathcal{L}_R$, and grounding self-conditioning while removing the intent queries and intent supervision. The future-supervision control retains the latent and grounding streams but replaces $I_t^\star$ with an endpoint visual representation, providing the segment's visual outcome in place of the multimodal execution evidence. The free-latent control retains the latent queries, bottleneck context routing, context dropout, visual and textual grounding streams, their alignment objectives, and self-conditioning while removing both $\mathcal{L}_I$ and $\mathcal{L}_{\text{mis}}$. The intent-only control retains teacher-intent alignment, bottleneck context routing, context dropout, and action-only intent-mismatch training while removing the visual and textual grounding streams.

Table 8: **Configuration of supervision and capacity controls.** Task-level success rates are reported in Table 9. Free latent removes both $\mathcal{L}_I$ and $\mathcal{L}_{\text{mis}}$. Intent only retains routing and action-only mismatch training.

Variant	Latent target	V/R groundings	Latent queries
Groundings only	–	Yes	No
Future supervision	Endpoint visual	Yes	Yes
Free latent	–	Yes	Yes
Intent only	Teacher intent	No	Yes
INDI	Teacher intent	Yes	Yes

The intent-only configuration reaches 76.0%, exceeding the backbone by 14.5 pp and future supervision by 8.0 pp. These gains are obtained with routing and action-only mismatch retained as part of intent-only training. The remaining configurations test whether complementary visual and textual predictions further strengthen this combination.

Adding the visual stream raises average success from 76.0% to 81.0%, adding the textual stream yields 79.5%, and adding both yields 85.5%. The joint configuration has its largest advantage on EP-Basket, reaching 100.0% compared with 74.0% and 72.0% for the individual streams. The contribution of each stream varies by task, with the visual-only addition performing best on Carrot. Using both streams gives the strongest average performance across these tasks, supporting complementary outcome and purpose supervision within the full training procedure.

B.2 Intent-Dependence Mechanism Ablations

We ablate the two training mechanisms introduced to make downstream decoding depend on recovered intent: bottleneck context routing and intent-mismatch training. The full model samples $\alpha \sim \text{Bernoulli}(0.5)$, requiring action and grounding rows to obtain VLA context through the intent rows when $\alpha = 0$, and applies $\mathcal{L}_{\text{mis}}$ to encourage lower downstream loss under matched than mismatched intent. We compare the full model with variants that remove the context bottleneck, the mismatch objective, or both. The no-bottleneck variant keeps direct context access open and retains the mismatch objective. The no-mismatch variant retains routing and sets $\lambda_{\text{mis}} = 0$. All four variants retain the teacher-intent, visual-outcome, and textual-purpose targets.

Removing the context bottleneck reduces average success from 85.5% to 70.0%, while removing intent-mismatch training reduces it to 74.0%. Removing both mechanisms further lowers success to 65.5%. With all targets held fixed, these results show that the same supervision becomes more

Table 9: **Supervision and capacity controls on SimplerEnv-Bridge.** V and R denote the visual and textual grounding streams. All variants share the backbone, budget, and a fixed evaluation run. Success rates (%).

Variant	V	R	Spoon	Carrot	Stack	EP-Basket	Avg.
GR00T-N1.7	–	–	82.0	68.0	66.0	30.0	61.5
+ Groundings only	✓	✓	88.0	82.0	38.0	32.0	60.0
+ Future supervision	✓	✓	84.0	70.0	56.0	62.0	68.0
+ Free latent	✓	✓	78.0	76.0	48.0	26.0	57.0
+ Intent only	–	–	86.0	84.0	70.0	64.0	76.0
+ Intent, V	✓	–	90.0	**88.0**	72.0	74.0	81.0
+ Intent, R	–	✓	90.0	86.0	70.0	72.0	79.5
INDI (Ours)	✓	✓	**90.0**	78.0	**74.0**	**100.0**	**85.5**

Table 10: **Ablation of intent-dependence mechanisms on SimplerEnv-Bridge.** All variants use GR00T-N1.7, the same training budget, and the same fixed evaluation run. The context-bottleneck ablation allows action and grounding rows to access the VLA context throughout training, while the mismatch ablation sets $\lambda_{\text{mis}} = 0$.

Variant	Spoon	Carrot	Stack	EP-Basket	Avg.
INDI	90.0	78.0	74.0	100.0	85.5
w/o context bottleneck	88.0	76.0	60.0	56.0	70.0
w/o $\mathcal{L}_{\text{mis}}$	92.0	80.0	30.0	94.0	74.0
w/o both	86.0	72.0	58.0	46.0	65.5

effective when the decoder is trained to use it. The interventions below then examine whether the trained decoder actually depends on its recovered states.

Alignment with mismatch held fixed. The mechanism ablations retain intent alignment while changing how downstream use is trained. We now examine the contribution of alignment when mismatch training is held fixed. We set only the coefficient of $\mathcal{L}_I$ to zero, retaining mismatch, routing, V/R supervision, self-conditioning, and the training budget. Table 11 places this comparison alongside the configurations with mismatch disabled.

Table 11: **Intent alignment and mismatch training.** Success rates (%). All four configurations retain routing and V/R supervision while varying intent-target alignment and mismatch training.

$\mathcal{L}_I$	$\mathcal{L}_{\text{mis}}$	Spoon	Carrot	Stack	EP-Basket	Avg.
Off	Off	78.0	76.0	48.0	26.0	57.0
Off	On	80.0	78.0	58.0	36.0	63.0
On	Off	92.0	80.0	30.0	94.0	74.0
On	On	90.0	78.0	74.0	100.0	85.5

With mismatch training active, intent alignment increases average success from 63.0% to 85.5%, a gain of 22.5 pp. Without mismatch, alignment raises success from 57.0% to 74.0%. Conversely, mismatch adds 6.0 pp without alignment and 11.5 pp with alignment. Thus, mismatch benefits an unaligned latent state, and teacher-intent alignment provides an additional gain when mismatch is already present. The full model performs best when the latent state is aligned with the teacher and predictions are trained to favor matched over substituted states.

B.3 FUNCTIONAL ROLE AND STRUCTURE OF RECOVERED INTENT

The preceding ablations establish the policy benefit of jointly training state recovery and downstream use. We now examine whether the learned states carry task-relevant information and participate in the computation that generates behavior.

The action decoder relies on recovered intent. At the intent-alignment layer, we replace all eight recovered intent states with zero vectors while preserving the observation, instruction, proprioceptive state, action noise, grounding states, and remaining decoder computation.

Table 12: **Dependence on recovered intent.** The zero-intent intervention replaces all intent states at the alignment layer while preserving all other policy inputs. Both conditions use the same evaluation run on SimplerEnv-Bridge.

Intent state	Spoon	Carrot	Stack	EP-Basket	Avg.
Recovered intent	**90.0**	**78.0**	**74.0**	**100.0**	**85.5**
Zero intent	58.0	44.0	40.0	40.0	45.5

As shown in Table 12, zeroing recovered intent reduces average success from 85.5% to 45.5%, below the 61.5% action-only baseline. The recovered state is therefore not merely an auxiliary alignment target but a representation used by downstream action decoding.

Task identity is readable across episodes and domains. We collect the eight intent-row hidden states immediately before the alignment layer from 580 Bridge demonstration segments and 202 simulation rollout segments. Task labels distinguish the four evaluation tasks, whose scenes and objects also differ. Each segment is represented by the flattened $K_I d_D$-dimensional recovered intent state. We reduce the representations to 50 dimensions using PCA and fit three shrinkage-LDA axes from the four Bridge task labels. The two-dimensional task-discriminative projection is shown in Figure 3a. Raw-space one-nearest-neighbor classification achieves 92.0% task purity, and episode-disjoint shrinkage LDA achieves 86.2% accuracy. Randomly permuting the task labels gives 31.9% mean accuracy across 20 permutations, with a maximum of 41.4%. A classifier trained only on real-demonstration intents classifies simulation-rollout intents with 63.9% to 80.7% accuracy across tasks. Task-label information is therefore readable across episodes and remains accessible when moving from real demonstrations to policy rollouts in simulation.

Temporal progress is readable across related tasks. The four Bridge tasks manipulate different objects and target relations but share an approach–grasp–transport–place structure. To test whether their states also record progress through execution, we divide normalized episode time into early, middle, and late thirds. We fit a phase-discriminative projection with these elapsed-progress labels pooled across tasks (Figure 3b). Three-way phase prediction reaches 77.8% five-fold cross-validation accuracy, compared with 33.3% chance.

We also ask whether task identity remains readable at each stage. Task purity is 36.6% within the phase plane, which groups segments by elapsed progress. When evaluated within the early, middle, and late intervals, task identity remains readable at 94.0%, 90.0%, and 90.0%, respectively.

We then test whether progress learned from some tasks can predict progress in another. A ridge-regression direction trained in a leave-one-task-out setting yields held-out correlations of 0.62 for Spoon, 0.70 for Carrot, 0.72 for Stack, and 0.69 for EP-Basket, compared with an in-sample correlation of 0.88.

Thus, the states distinguish tasks within each temporal stage and allow progress prediction across these related manipulation tasks. Here, temporal progress can covary with end-effector configuration and object motion. The following analysis tests whether objective-related similarity persists when we control for the measured action sequence.

Objective labels remain distinguishable under action-sequence controls. We ask whether similarity between representations with the same objective label can be explained by similarity in their measured action sequences. We compare progress-matched episode pairs under two opposing conditions: the same objective realized by dissimilar sequences, and different objectives realized by similar sequences. We use episode-disjoint discovery and test splits, with all centering statistics and pair-selection thresholds estimated from discovery episodes. Object and scene identity are not additional matching criteria.

As shown in Figure 6, teacher intent targets separate by objective on held-out episodes, while a matched supervised projection of local action snippets yields weak objective separation. We then compare similarities directly in the representation space, without fitting a supervised projection.

Table 13: **Task and temporal-progress information in recovered intent.** The free-latent control has the same context access and removes both teacher alignment and mismatch training. Task LDA is episode-disjoint. Phase accuracy uses five-fold cross-validation.

Property	Evaluation	Result
Task identity		
Task identity	Raw-space task 1-NN purity	92.0%
Across episodes	Episode-disjoint task LDA	86.2%
Supervision control	Free latent (no $\mathcal{L}_I, \mathcal{L}_{\text{mis}}$)	55.7%
Cross-domain transfer	Train on real, test on simulation	63.9–80.7%
Task retention	Task purity in early/middle/late phases	94.0/90.0/90.0%
Shared execution progress		
Phase readability	Three-way phase prediction	77.8%
Task leakage	Task purity within phase plane	36.6%
Cross-task transfer	Leave-one-task-out progress correlation	0.62–0.72

Recovered-intent similarity is higher for the same objective with dissimilar sequences than for different objectives with similar sequences (0.150 vs. -0.017), with a gap of 0.167 and a 95% episode-bootstrap confidence interval of $[0.089, 0.245]$. Teacher targets show the same ordering (0.470 vs. 0.158, gap 0.312, 95% CI $[0.253, 0.375]$). In these comparisons, both teacher and student states group segments by objective label even when their measured action sequences suggest the opposite grouping.

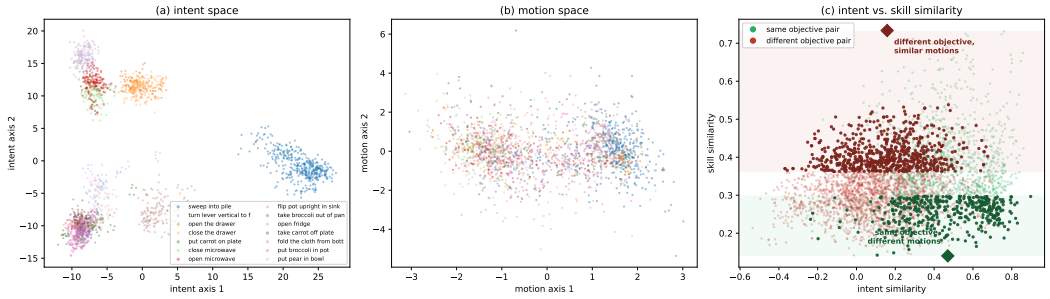


Figure 6: **Objective-label separation under action-sequence controls.** Pairs are selected using their measured action sequences. (a) Teacher intent targets separate 14 behavior objectives on held-out episodes. (b) A matched supervised projection of local action snippets yields weak objective separation. (c) In a projection-free comparison of progress-matched episode pairs, the same objective realized by dissimilar action sequences remains more similar in intent space than different objectives realized by similar sequences. Light points show all eligible pairs, dark points show the selected contrast sets, and diamonds mark their means. Recovered policy intent exhibits the same ordering.

Reading instruction targets across objects and scenes. We next ask whether a probe can recover the target specified by an instruction when tested on objects or scenes excluded from its training data. The analysis uses 50,000 samples from 2,348 episodes. Labels are the target heads parsed from the instructions: for example, "put carrot on plate" has the label "plate." There are 12 labels in total, with an imbalanced distribution, so we report balanced accuracy. Within each object, a probe classifies the two to four target labels available for that object; the reported within-object score averages these results using episode counts as weights. Held-out-object and held-out-scene folds retain only labels shared by their training and test partitions, typically ten to twelve classes. These splits concern the probe's training and evaluation data.

Table 14: **Probing instruction-specified target labels.** Balanced accuracy (%) for instruction-derived target-head labels. Within-object and held-out evaluations use different class sets, so their accuracies describe distinct classification conditions. The variant without intent alignment retains mismatch training and the other full-model components.

Representation	Within object	Held-out objects	Held-out scenes
Instruction embedding	98.5	85.5	50.3
Frozen VLA context	98.0	50.7	41.2
Baseline decoder	98.5	59.0	47.3
INDI recovered intent	98.3	64.1	48.5
Without intent alignment	98.3	60.2	47.0

Within-object classification is near saturation across all evaluated representations (Table 14). On held-out objects, recovered intent reaches 64.1%, exceeding the baseline decoder by 5.1 pp and the frozen VLA context by 13.4 pp. Compared with the variant without intent alignment, recovered intent gains 3.9 pp on held-out objects and 1.5 pp on held-out scenes (48.5% vs. 47.0%). The advantage is largest on held-out objects, where the probe predicts the instruction's target more accurately from recovered intent than from the other decoder or context representations. The labels describe instruction-level target identity and complement the task and progress analyses above.

Task information is readable in different decoder streams. To locate task information beyond the alignment layer, we probe each decoder stream across depth. We therefore measure one-nearest-neighbor task purity for the state, action, visual-grounding, textual-grounding, and intent rows throughout the 32 decoder blocks.

Table 15: **Task information across decoder rows and depth.** The table summarizes task information before and after the intent-alignment layer.

Decoder rows	Blocks 0–15	Blocks 19–31
Intent	$0.63 \rightarrow \mathbf{0.93}$	0.77 ± 0.05
Visual grounding	Near chance	Up to $\mathbf{0.95}$
Textual grounding	Near chance	Up to $\mathbf{0.97}$
Action and state	Near chance	Near chance

Intent-row task purity peaks at 0.93 immediately before the alignment layer. After alignment, the same probe reaches 0.95 and 0.97 in the visual and textual rows, while action and state rows remain near chance under this nearest-neighbor readout. The concentration of readable task information in the intent and auxiliary rows motivates jointly editing these streams in the decoder tail.

Editing task-discriminative coordinates changes closed-loop behavior. Zeroing intent shows dependence on that pathway, but changes the entire state. We next replace task-discriminative coordinates while retaining the orthogonal component, and jointly edit the intent and auxiliary streams. Let $g \in \{I, G\}$ index the edited token group, where I contains the intent rows and $G = [V, R]$ contains the visual and textual grounding rows. Let $h_{g,k} \in \mathbb{R}^{p_g}$ denote the flattened representation after decoder block k. For task t, we compute the block-wise centroid

$$\mu_{g,k}^{(t)} = \mathbb{E}[h_{g,k} \mid \text{task} = t]$$

and form a centered matrix from the four task centroids. The columns of $U_{g,k}$ are the right singular vectors spanning the resulting between-task subspace. For an injected task t_d, we replace only the coordinates inside this subspace:

$$h'_{g,k} = h_{g,k} + U_{g,k}\left[U_{g,k}^{\top}\mu_{g,k}^{(t_d)} - U_{g,k}^{\top}h_{g,k}\right].$$

This sets the task-discriminative coordinates to the donor-task centroid while preserving the orthogonal component. We apply the edit to the intent and grounding groups after every decoder block from 16 through 24, without modifying the action or state rows. Task centroids and bases are estimated from separate rollouts under environment seeds disjoint from intervention evaluation. The

same-task, cross-task, and Gaussian-noise conditions use the same intervention schedule and 50 evaluation episodes per task and condition. We measure success on the receiver's original task. The schedule is matched across conditions, while the resulting displacement magnitudes are unconstrained.

The resulting closed-loop success matrix is reported in Table 6b.

Same-task interventions retain 84.5% success on average, compared with 45.2% for cross-task interventions, while Gaussian corruption reduces success to 1.0%. The contrast shows that original-task performance is sensitive to the content of the jointly edited intent and grounding states.

Intent geometry reflects task relatedness. The cross-task intervention effects in Table 6b raise the question of whether recovered intent reflects which tasks are related. We therefore compare centered cosine similarity between recovered-intent centroids for the four SimplerEnv evaluation tasks. As shown in Figure 7, Spoon and Carrot form the closest pair, Stack is more distinct, and EP-Basket is the most separated from the other tasks. This organization is consistent with Spoon and Carrot sharing a closely related tabletop manipulation structure, whereas EP-Basket differs in both its objective and scene. This descriptive measure of task relatedness offers one possible account of the uneven compatibility of cross-task donors.

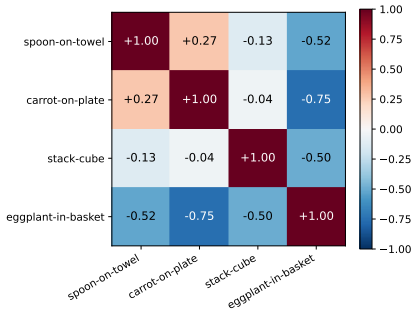


Figure 7: **Task relatedness in recovered-intent space.** Each entry reports centered cosine similarity between the recovered-intent centroids of two SimplerEnv tasks. Spoon and Carrot form the closest pair, while EP-Basket is the most separated from the other tasks.

Phase-coordinate edits alter execution. We repeat the task-coordinate intervention with a phase subspace in place of the task subspace. Here, $U_{g,k}$ spans the early, middle, and late centroids pooled across tasks using the projection of Figure 3b. The same joint token-group and decoder-block schedule is used, with phase centroids defining the replacement coordinates. Forcing an early-stage donor reduces average success to 7.3%, and forcing a late-stage donor reduces it to 5.5%, both approaching the 1.0% obtained under Gaussian corruption.

The selected rollouts in Figures 8 to 10 illustrate approach- and placement-like behaviors under phase-coordinate edits. Early forcing leaves the policy executing approach behavior without transitioning to grasp: it hovers over the target on Carrot, displaces the spoon without closing the gripper on Spoon, and fumbles at the green block without initiating the stack on Stack. Late forcing produces the opposite pattern, skipping the grasp and executing placement. This is clearest on Spoon, where the policy moves to the towel and presses down with an empty gripper. These rollouts connect the low success rates to recognizable changes in execution stage.

To contextualize these changes, the phase-matched control injects a donor from the same task at the receiver's own stage and retains 50%, 70%, and 32% success on Spoon, Carrot, and Stack. Its decline relative to the unmodified policy shows that matching the temporal stage alone is insufficient to preserve execution under this editing procedure. These results expose a challenge for selective progress control: changing stage-related content while preserving the execution state needed to carry out the corresponding action.

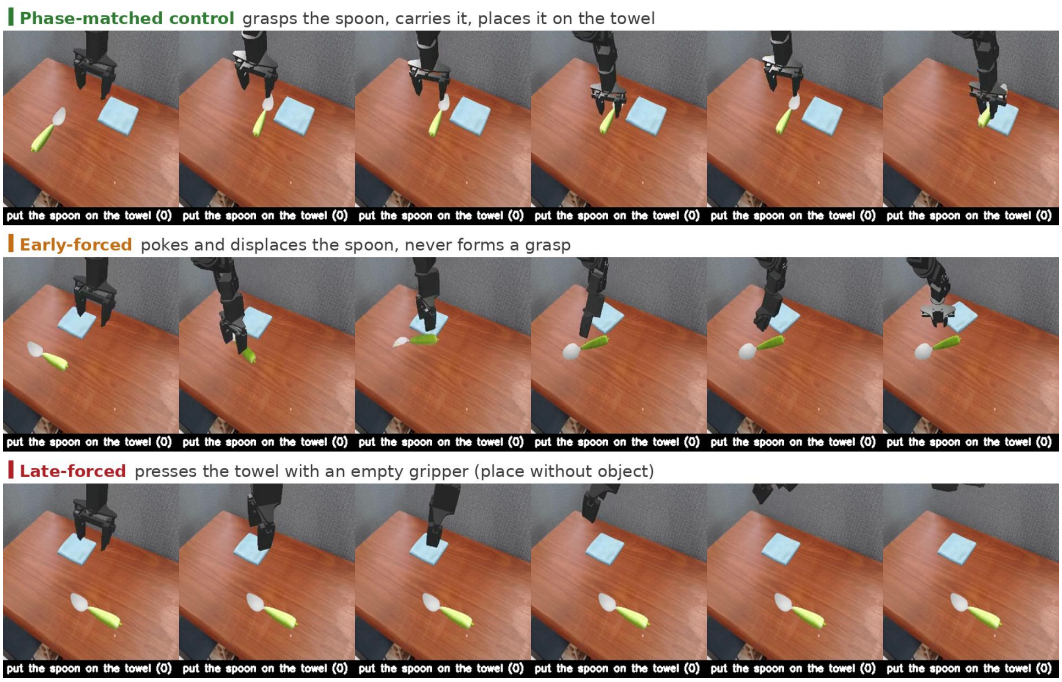


Figure 8: **Stage-forced intervention on Spoon.** Rows show phase-matched, early-forced, and late-forced injection. Early forcing leaves the policy displacing the spoon without ever closing the gripper. Late forcing skips the grasp entirely: the policy moves to the towel and presses down with an empty gripper, executing placement as if it were already carrying the object.

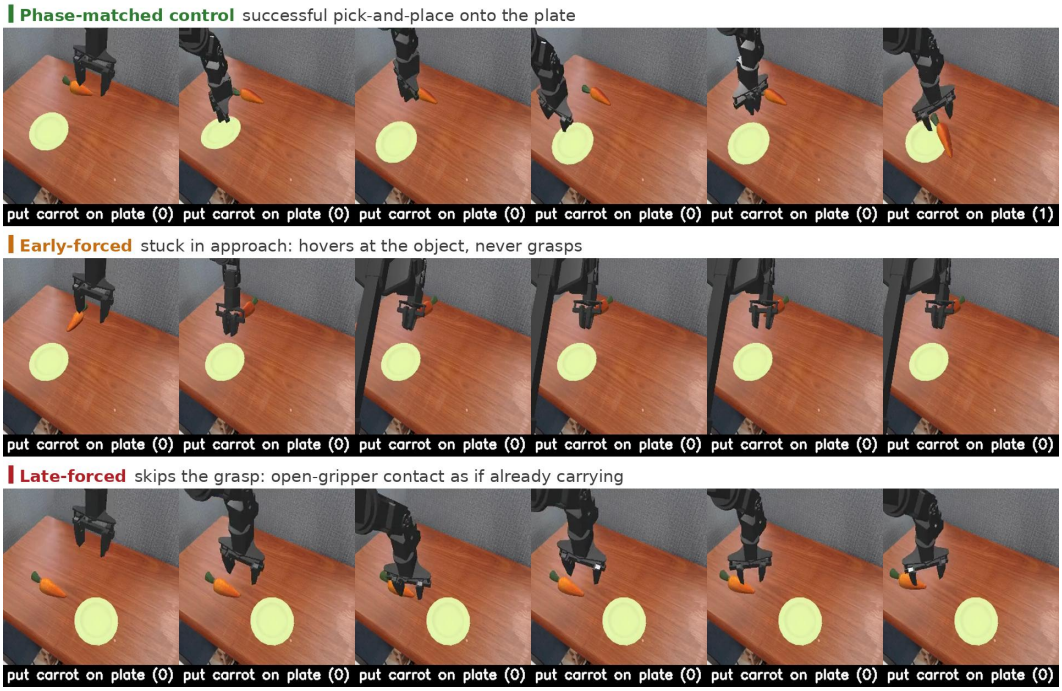


Figure 9: **Stage-forced intervention on Carrot.** Early forcing leaves the policy hovering over the carrot without transitioning to grasp. Late forcing produces open-gripper contact, as if the object were already being transported.

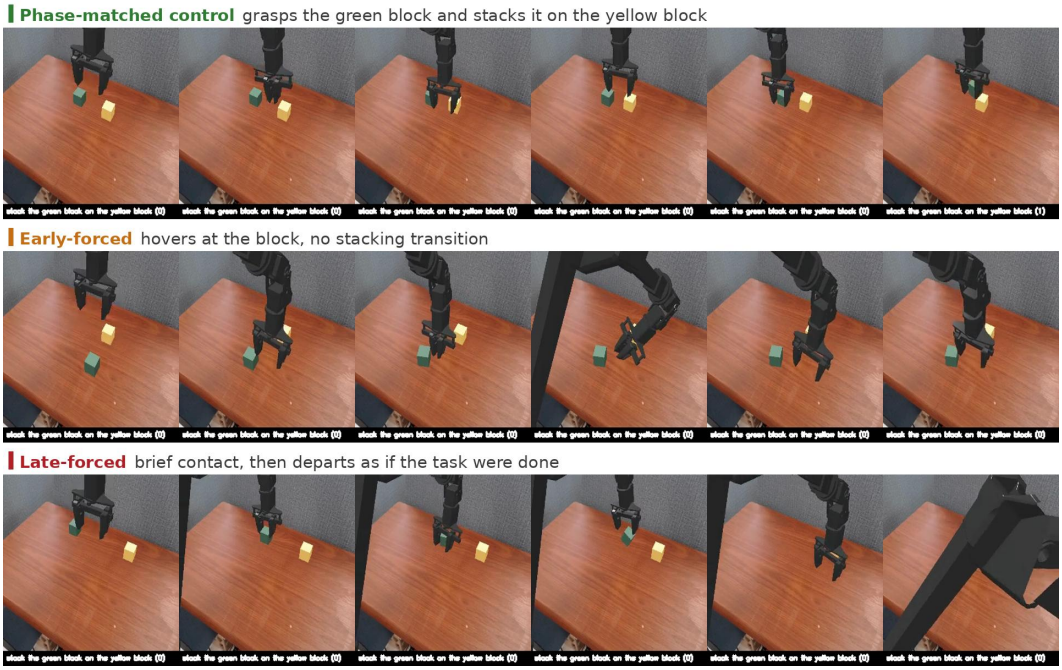


Figure 10: **Stage-forced intervention on Stack.** Early forcing leaves the policy fumbling at the green block without initiating the stack. Late forcing produces brief contact followed by departure, as if the task were already complete.

State replacement with two feasible goals. Table 6b shows that action generation is sensitive to the content of the joint intent–grounding state. We next test whether state replacement can also redirect the policy toward another feasible goal while its instruction remains fixed.

To make a goal change observable, we construct a flat-table scene in which the spoon can be placed on either a towel or a plate. With the corresponding instructions, the unedited policy achieves 82.4% towel success and 76.9% plate success. The respective cross-goal success rates are 0.0% and 5.8%.

All replacements use the deployment routing setting ($\alpha = 1$), with the towel instruction fixed and direct access to the visual-language context retained. We replace intent rows I, grounding rows $G = [V, R]$, or both with fixed donor states from matched-goal (towel) or cross-goal (plate) executions.[1]

Table 16: **State replacement with two feasible goals.** Success rates (%) on the original and alternative goals with the towel instruction fixed. Matched/cross donors pursue the towel/plate goal.

Edited rows	Donor	Towel (original)	Plate (alternative)
None	–	82.4	0.0
I only	Matched	78.0	6.0
I only	Cross	72.0	0.0
G only	Matched	78.0	2.0
G only	Cross	82.4	0.0
I+G	Matched	76.0	6.0
I+G	Cross	78.0	0.0

Across the replacement conditions, original-goal success ranges from 72.0% to 82.4% (Table 16). No cross-goal donor condition attains the plate goal in these rollouts; matched-donor replacements produce plate success in 2.0%–6.0% of episodes.

[1]The pilots use 51/52 episodes for the towel/plate instructions; replacement conditions use 50 episodes each, except 51 for cross-goal G-only replacement.

Although the joint state influences execution (Table 6b), this influence does not yield goal redirection under the tested replacements. The decoder continues to condition on the unchanged instruction and current observation, which may constrain the effect of a donor state. Developing edits that coordinate goal-related content with the ongoing execution remains a promising direction for extending INDI toward selective goal control.

B.4 VISUAL AND TEXTUAL READOUT OF INTENT

Task and progress labels provide a coarse account of what the learned states contain. We now examine whether their features can recover the segment's visual outcome and a description of its purpose, and whether replacing intent changes these readouts. We freeze the trained policy for all readout experiments. Let

$$\widehat{I}_t = P_I\left(H_{I,t}^{L_{\text{tap}}}\right), \qquad \widehat{V}_t = P_V\left(H_{V,t}^{L_{\text{out}}}\right), \qquad \widehat{R}_t = P_R\left(H_{R,t}^{L_{\text{out}}}\right)$$

denote the projected intent and the final visual and textual grounding predictions. We evaluate them using nearest-neighbor retrieval and separately trained readout decoders while keeping the policy frozen.

Embedding-space readout. We extract $\widehat{V}_t$, $\widehat{R}_t$, $V_t^\star$, and $R_t^\star$ for approximately 10,000 held-out segments. We center representations using per-slot means from the unmodified policy condition and reuse these means for the interventions. Each retrieval bank contains the paired positive target and negatives from other episodes. Other targets from the query episode are excluded. Shuffled cosine uses targets from different episodes with slot order preserved.

Table 17: **Embedding-space readout of the visual and textual groundings.** Correct cosine measures centered similarity to the paired target. Shuffled cosine uses randomly permuted targets. Median rank is zero-indexed.

Readout	Correct cosine	Shuffled cosine	Margin	Top-1	Top-10	Median rank
$\widehat{V}_t \to V_t^\star$	**0.650**	0.003	**0.647**	**72.6%**	**95.3%**	**0**
$\widehat{R}_t \to R_t^\star$	0.158	−0.004	0.161	1.5%	5.0%	1211

The visual grounding provides a highly readable endpoint representation, retrieving the paired target at rank one in 72.6% of held-out segments and within the top ten in 95.3%. The textual grounding also distinguishes paired from shuffled targets in cosine similarity, but retrieves the exact paired target less consistently. This difference motivates comparing direct feature retrieval with the learned textual readout below.

Visual grounding depends on intent. We repeat the visual readout after zeroing recovered intent or replacing it with recovered intent from another task while preserving the remaining policy inputs.

Table 18: **Effect of intent intervention on visual grounding.**

Intent condition	Centered margin	Top-1 retrieval
Recovered intent	**0.647**	**72.6%**
Zero intent	0.513	59.1%
Cross-task intent	0.393	41.8%

Visual readout quality decreases from recovered to zero to cross-task intent. The predicted visual outcome therefore depends on the content supplied through the intent pathway.

Visual-outcome retrieval. We retrieve actual endpoint frames using $\widehat{V}_t$. For the intervention montage, the retrieval bank includes the designated current-task and injected-task endpoints.

In Figure 11, cross-task intent moves the retrieved endpoint toward the donor behavior, showing that the intent pathway affects the visual content predicted downstream. Each donor brings its associated object and scene, so the observed shift concerns the visual outcome of that donor execution.

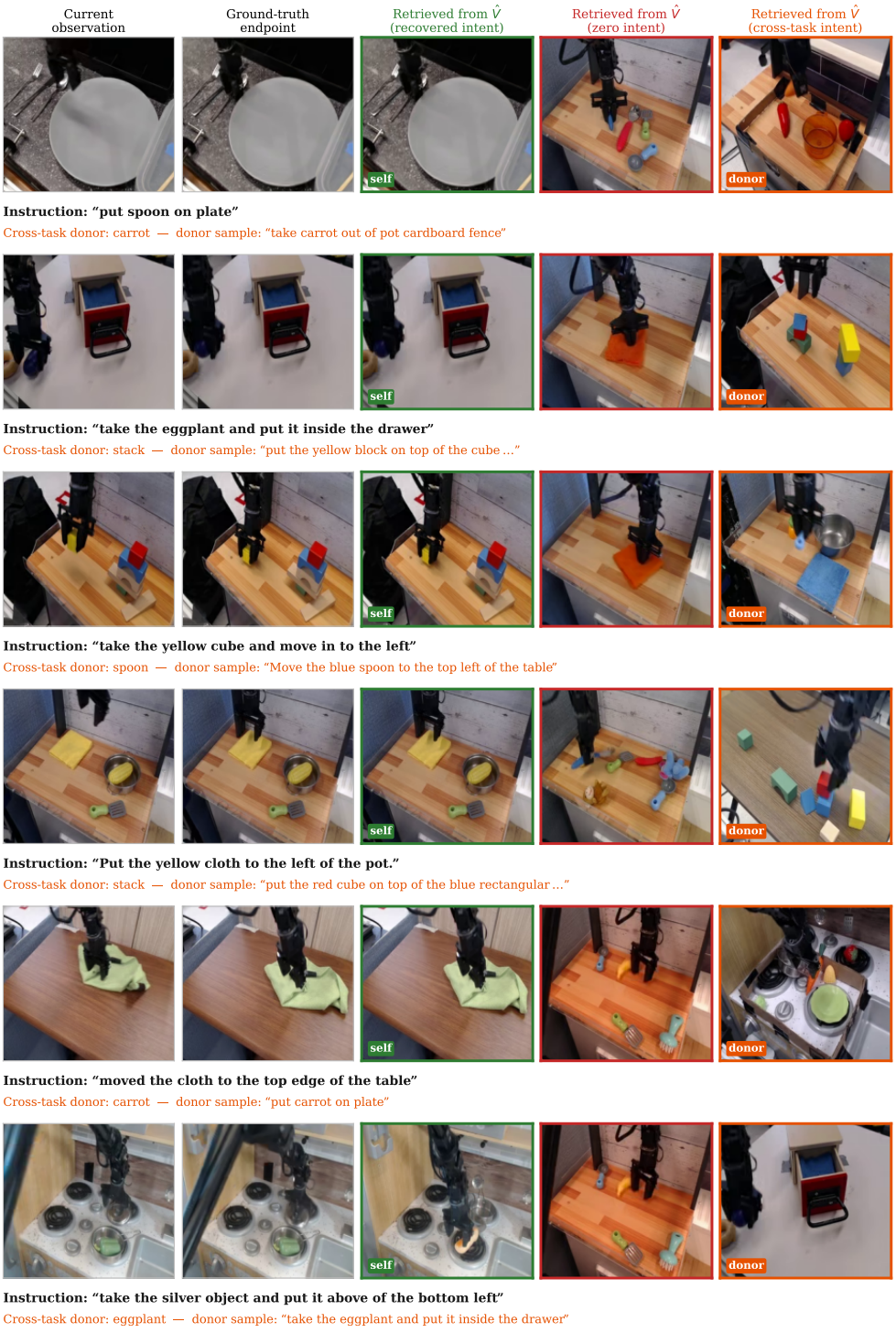


Figure 11: **Non-parametric visual-outcome readout.** Each row shows the current observation, ground-truth endpoint, and endpoint retrieved from $\widehat{V}_t$ under recovered, zero, and cross-task intent. Recovered intent retrieves the current behavior outcome, while cross-task intent redirects retrieval toward the injected behavior. Across the evaluated pairs, recovered intent retrieves the current endpoint in 70.5% of cases. After cross-task replacement, the injected endpoint is retrieved in 78.6% of cases, while the original endpoint remains in only 0.1%.

Learned visual and textual decoding. We additionally train lightweight decoders to read endpoint images and functional-purpose statements from the feature representations. The visual decoder maps 16 ordered visual slots to an RGB image using a transformer bottleneck and convolutional upsampling. We train it on teacher visual targets $V_t^\star$ paired with endpoint observations, keeping the policy frozen.

We first check reconstruction from the teacher target itself. On held-out data, decoding $V_t^\star$ yields a changed-region L_1 error of 0.0819, compared with 0.3042 for copying the current observation. This quantitative check evaluates the teacher target and its decoder. Reconstructions from policy predictions $\widehat{V}_t$ are shown separately in Figure 12. The changed-region mask selects pixels whose mean absolute RGB difference between current and endpoint images exceeds 0.08 on the $[0, 1]$ scale. The error averages absolute RGB reconstruction error over those pixels.

For textual readout, we compare the final textual grounding $\widehat{R}_t$ with the projected intent $\widehat{I}_t$. The intent decoder is trained on approximately 9,900 unmodified intent–statement pairs using an episode-disjoint split. We report deterministic object F1, state-change verb F1, and ROUGE-L.

Table 19: **Textual-purpose readout.** The oracle row decodes the teacher textual target. The remaining rows decode the predicted textual grounding and recovered intent.

Readout representation	Object F1	Verb F1	ROUGE-L
$R_t^\star$ (oracle target)	**0.998**	**1.000**	**0.933**
$\widehat{R}_t$	0.314	0.195	0.255
$\widehat{I}_t$	**0.693**	**0.432**	**0.380**

The recovered intent is substantially more readable than the final textual grounding, improving object F1 from 0.314 to 0.693 and verb F1 from 0.195 to 0.432. Recovered intent therefore provides the more informative input for this readout, with particularly strong gains in identifying objects.

The decoded images and statements let us inspect how intent replacement changes the predicted outcome and purpose. The examples in Figure 12 also make the remaining readout errors concrete: the first decodes approach for grasping where the teacher describes positioning for placement, and the last substitutes the left side for the top edge of the table. More faithful decoding of these action and relational details is a natural next step toward using the states as an interpretable interface to the policy.

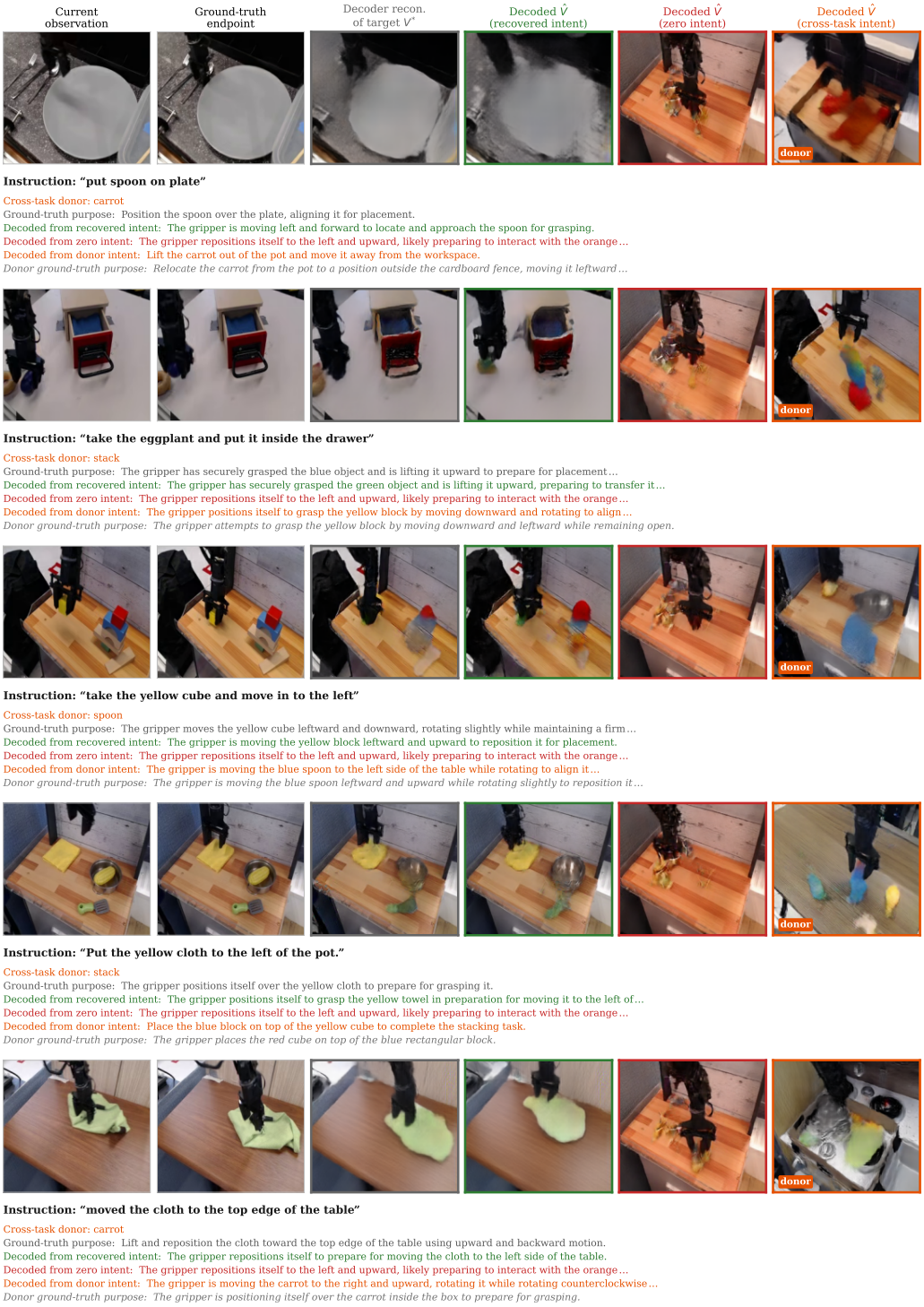


Figure 12: **Learned visual and textual readout of intent.** Each row shows the current observation, ground-truth endpoint, visual reconstruction from $V_t^\star$, and visual outcomes decoded from $\widehat{V}_t$ under recovered, zero, and cross-task intent. The text labeled "ground-truth purpose" is a teacher-generated reference, followed by statements decoded from recovered, zero, and injected intent. Cross-task examples use donor states from different task scenes.

B.5 Teacher Quality and Intent-Target Geometry

We examine how the teacher checkpoint and resulting target geometry relate to policy performance. We repeat the controlled SimplerEnv-Bridge training while changing only the teacher cache, keeping the demonstrations, student architecture, optimization budget, checkpoint selection, and evaluation protocol fixed.

Teacher choice matters. Table 20 shows that teacher choice substantially changes downstream performance. Qwen3.5-9B (Qwen Team, 2026) reduces average success to 54.5%, below the 61.5% action-only baseline, whereas Qwen3-VL-8B-Instruct (Bai et al., 2025) reaches 73.5% and Cosmos-Reason2-8B reaches 85.5%. Cosmos-Reason2-8B (NVIDIA, 2025a) is obtained by physical-AI post-training of Qwen3-VL-8B-Instruct. This checkpoint comparison changes the complete multimodal representation supplied to the student, motivating a closer examination of the resulting targets.

Table 20: **Controlled teacher swap on SimplerEnv-Bridge.** All variants use the same GR00T-N1.7 student, demonstrations, optimization budget, and fixed evaluation run. Only the teacher cache is changed.

Teacher	Spoon	Carrot	Stack	EP-Basket	Avg.
Action-only baseline	82.0	68.0	66.0	30.0	61.5
Qwen3.5-9B	84.0	78.0	40.0	16.0	54.5
Qwen3-VL-8B-Instruct	90.0	98.0	34.0	72.0	73.5
Cosmos-Reason2-8B	90.0	78.0	74.0	100.0	**85.5**

Target geometry varies with teacher choice. We characterize each teacher's $I^\star$ targets on the same 17,288 valid Bridge samples. The *angular budget* measures the mean cosine distance of L2-normalized targets from their mean direction, and therefore how much sample-dependent directional variation is available to the cosine-alignment objective. The *effective rank* summarizes the spectrum of the centered target distribution, while the *within-episode share* measures the fraction of centered variance that occurs within episodes rather than between them.

Table 21: **Intent-target geometry across teachers.** All statistics use the same valid Bridge samples ($n = 17{,}288$). Bridge Avg. reports the corresponding controlled policy performance.

Teacher	Angular budget	Eff. rank	Within-ep. share	Bridge Avg.
Qwen3.5-9B	0.0027	139.4	0.308	54.5
Qwen3-VL-8B-Instruct	0.0098	7.2	0.693	73.5
Cosmos-Reason2-8B	0.0327	3.4	0.723	**85.5**

Qwen3.5 produces an almost degenerate alignment signal: its angular budget is only 0.0027, so a nearly constant prediction can satisfy much of the cosine-alignment objective, and only 30.8% of its residual variation occurs within episodes. Qwen3-VL targets vary more across samples and within individual episodes, accompanied by a 12.0-percentage-point gain over the action-only baseline. Cosmos preserves the high within-episode share while increasing the angular budget by $3.3\times$ over Qwen3-VL, accompanied by a further 12.0-point downstream gain. The high effective rank of Qwen3.5 describes the structure of a small residual signal and does not predict downstream utility on its own.

Across these three teachers, angular budget and within-episode share increase with policy success. The geometry thus complements the policy comparison by showing how much the targets vary across segments and within executions, providing a practical diagnostic of the alignment signal.

Prompt control under identical execution evidence. We test whether asking the teacher what a segment accomplishes produces better policy supervision than asking it to describe the same segment neutrally. The policy comparison replaces only $I^\star$, retaining the original $V^\star/R^\star$ targets and the full model's student and evaluation settings. Before comparing policies, we verify that the prompt change affects the extracted targets. This paired check uses 3,945 segments, holding the video, caption, and extraction mask fixed.

Paired-target measurement	Value
Segments	3,945
Mean cosine (raw)	0.917
Mean relative L2 change	0.389
Centered cosine, paired / shuffled	0.288 / 0.016

Changing the prompt changes the target features, as measured by the relative L2 difference. Although raw cosine similarity is high, the centered comparison is larger for paired segments than for shuffled pairs. The targets therefore retain correspondence to the same segment despite the change in prompt.

Table 22 reports policy performance with the two sets of targets.

Table 22: **Teacher prompts under matched execution evidence.** Success rates (%) under the same execution evidence and fixed evaluation run.

Teacher prompt	Spoon	Carrot	Stack	EP-Basket	Avg.
Functional intent	90.0	78.0	74.0	100.0	85.5
Neutral description	88.0	76.0	58.0	82.0	76.0

The functional prompt reaches 85.5% average success, compared with 76.0% for the neutral description, a gain of 9.5 pp. Functional prompting improves success on all four tasks, with the largest gains on Stack ($+16.0$ pp) and EP-Basket ($+18.0$ pp). With the actions and video unchanged, the student learns a better policy when the teacher is prompted about what the segment accomplishes.

B.6 Intent-Target Source Ablation

The teacher both processes the executed segment and generates a description of its purpose, providing several candidate representations for supervising intent. We compare the default input-span target $\mathcal{E}_t$ with two response-side alternatives: the complete generated response $\mathcal{Y}_t$, including reasoning and the final purpose statement, and the final-purpose span $\mathcal{S}_t$ alone. All three targets use the same teacher inputs, layer 18, and $K_I = 8$ slots. We retain the original visual and textual targets, student architecture, optimization budget, and evaluation protocol, isolating the choice of teacher feature span.

Table 23: **Intent-target source ablation on SimplerEnv-Bridge.** Success rates (%). All targets use teacher layer 18 and $K_I = 8$ contiguous-region pooling, with the original V/R targets retained.

Intent target	Spoon	Carrot	Stack	EP-Basket	Avg.
Evidence-side input span $\mathcal{E}_t$	90.0	78.0	74.0	100.0	**85.5**
Generated-response span $\mathcal{Y}_t$	76.0	86.0	76.0	80.0	79.5
Final-purpose span $\mathcal{S}_t$	78.0	80.0	66.0	84.0	77.0

All three sources improve over the 61.5% action-only baseline. The evidence-side target achieves the highest average success at 85.5%, compared with 79.5% for the complete response and 77.0% for the final purpose alone. This comparison supports using the teacher's input representations for intent alignment. The prompt control in Table 22 addresses a separate choice: with the same inputs, asking about functional purpose also improves policy performance. The two controls favor both the input span and the functional prompt used to construct $I^\star$.

B.7 Execution Evidence for Intent Targets

Executed demonstrations let the teacher observe how an action segment unfolds, providing evidence beyond the current observation and instruction. To examine the value of this additional information for intent supervision, we construct a current-only target by removing the execution video and action summary from the teacher inputs and adapting the prompt to the available observation. The teacher, extraction layer, slot count, and student training are unchanged, and both variants retain the original visual-outcome and textual-purpose targets. The comparison therefore tests the contribution of execution evidence specifically to constructing $I^\star$.

Table 24: **Execution evidence for the intent target on SimplerEnv-Bridge.** Success rates (%). Both conditions include the instruction and retain the original V/R targets; current refers to the current observation.

Teacher evidence for $I^\star$	Spoon	Carrot	Stack	EP-Basket	Avg.
Current + action + video	90.0	78.0	74.0	100.0	**85.5**
Current only	78.0	70.0	64.0	78.0	72.5

Including the execution evidence improves average success from 72.5% to 85.5%, a gain of 13.0 percentage points, with improvements on all four tasks. This supports using completed executions to enrich supervision for a policy that must act from current inputs. The three controls address what the teacher sees, what it is asked, and which hidden states supervise the student. Each favors the corresponding choice in INDI (Tables 22 to 24).

C Additional Results

C.1 Full Quantitative Results

We provide complete RoboCasa Kitchen results to further examine the generality of INDI across VLA backbones and manipulation tasks. Table 25 reports the per-task success rates summarized by task category in the main text. The table compares INDI with controlled GR00T-N1.7 and $\pi_{0.5}$ baselines, the GR00T-N1.7 future-supervision variant, and off-the-shelf GR00T checkpoints trained with larger demonstration sets.

The per-task results show how the average gains are distributed across the benchmark. For $\pi_{0.5}$, INDI improves 13 tasks, reduces success on nine, and leaves two unchanged. Coffee Press Button rises from 20.0% to 74.7%, while Open Double Door falls from 32.0% to 10.7%. For GR00T-N1.7, future supervision has a different task profile, improving Open Double Door and Turn Off Stove while reducing Turn On Sink Faucet and Coffee Press Button. The full breakdown complements the category averages by identifying which tasks benefit from each supervision choice.

Performance under changed objects and distractors. We measure relative retention as success under a shifted condition divided by ID-clean success. Retention is comparable for the baseline and INDI under held-out objects (87.3% and 89.5%) and higher for INDI under distractors (74.6% and 81.6%). Thus, the absolute gains in Table 4 persist as the scene changes from training conditions, with better retention in the distractor setting.

Interpreting cumulative real-world stage completion. The stage profiles in Table 5 help locate where the real-world policies begin to differ. On Cross-Bin Stacking, the gap grows when retrieving the top cube and completing the stack. On Drawer Storage, INDI raises grasping success from 50.0% to 70.0%, with final completion increasing from 24.0% to 36.0%. Because the stage rates are cumulative, final completion among trials that reach grasping is 48.0% for the baseline and 51.4% for INDI. This conditional view locates much of the observed Drawer advantage at the initial grasp, complementing the later-stage gains on Cross-Bin Stacking.

Table 25: **Full RoboCasa Kitchen per-task results.** Success rates are reported across all 24 tasks. Controlled GR00T-N1.7 variants and $\pi_{0.5}$ results report mean $\pm$ sample standard deviation over three evaluation runs. The GR00T G_{3000} columns provide single-value data-scale references. Bold marks the best controlled result within each backbone in the category-summary rows.

	GR00T					$\pi_{0.5}$	
Task	N1.6 G_{3000}	N1.7 G_{3000}	N1.7 G_{100}	N1.7 + future supervision G_{100}	N1.7 + INDI G_{100}	Baseline	+ INDI
Pick-and-Place							
PnP from Cab to Counter	41.0	65.0	26.0 $\pm$16.0	44.7 $\pm$1.2	50.0 $\pm$4.0	18.7 $\pm$2.3	10.7 $\pm$10.1
PnP from Counter to Cab	47.5	60.0	37.3 $\pm$3.1	35.3 $\pm$5.0	58.7 $\pm$10.3	18.7 $\pm$2.3	23.3 $\pm$7.6
PnP from Counter to Microwave	19.0	30.0	22.0 $\pm$5.3	57.3 $\pm$5.8	28.0 $\pm$0.0	6.7 $\pm$2.3	6.7 $\pm$2.3
PnP from Counter to Sink	46.0	60.0	54.7 $\pm$11.7	29.3 $\pm$11.4	53.3 $\pm$10.3	10.0 $\pm$7.2	16.0 $\pm$4.0
PnP from Counter to Stove	63.2	60.0	46.7 $\pm$7.0	56.0 $\pm$6.0	50.7 $\pm$9.9	12.0 $\pm$6.9	8.0 $\pm$0.0
PnP from Microwave to Counter	24.5	19.0	23.3 $\pm$3.1	26.7 $\pm$2.3	32.0 $\pm$2.0	8.0 $\pm$0.0	8.0 $\pm$0.0
PnP from Sink to Counter	50.0	65.0	50.7 $\pm$8.3	64.7 $\pm$9.9	62.7 $\pm$3.1	24.7 $\pm$9.9	21.3 $\pm$2.3
PnP from Stove to Counter	54.5	65.0	54.7 $\pm$4.2	62.0 $\pm$3.5	62.7 $\pm$8.1	13.3 $\pm$2.3	28.7 $\pm$4.2
Open-or-Close							
Close Double Door	88.5	80.0	88.0 $\pm$7.2	60.0 $\pm$5.3	90.0 $\pm$2.0	57.3 $\pm$25.7	72.0 $\pm$6.9
Close Drawer	100.0	100.0	92.7 $\pm$8.1	68.0 $\pm$8.0	100.0 $\pm$0.0	77.3 $\pm$39.3	98.7 $\pm$2.3
Close Single Door	96.0	95.0	96.0 $\pm$4.0	89.3 $\pm$9.5	97.3 $\pm$3.1	73.3 $\pm$11.5	68.7 $\pm$1.2
Open Double Door	39.0	25.0	52.0 $\pm$5.3	94.0 $\pm$6.0	75.3 $\pm$3.1	32.0 $\pm$5.3	10.7 $\pm$2.3
Open Drawer	81.1	95.0	58.0 $\pm$8.7	88.0 $\pm$3.5	55.3 $\pm$5.8	42.7 $\pm$1.2	36.0 $\pm$14.4
Open Single Door	81.5	90.0	68.7 $\pm$3.1	82.7 $\pm$8.1	78.7 $\pm$7.0	48.0 $\pm$18.3	50.7 $\pm$6.1
Others							
Coffee Press Button	98.5	100.0	98.0 $\pm$2.0	32.0 $\pm$2.0	100.0 $\pm$0.0	20.0 $\pm$8.0	74.7 $\pm$6.1
Coffee Serve Mug	63.5	85.0	74.7 $\pm$9.0	73.3 $\pm$4.2	81.3 $\pm$7.0	25.3 $\pm$9.2	23.3 $\pm$9.9
Coffee Setup Mug	31.0	30.0	32.0 $\pm$10.6	79.3 $\pm$13.3	30.7 $\pm$1.2	14.7 $\pm$12.2	4.0 $\pm$0.0
Turn Off Microwave	96.0	95.0	99.3 $\pm$1.2	92.7 $\pm$4.2	99.3 $\pm$1.2	72.0 $\pm$16.0	90.7 $\pm$4.6
Turn Off Sink Faucet	93.5	100.0	93.3 $\pm$6.4	85.3 $\pm$7.6	98.0 $\pm$2.0	74.7 $\pm$19.7	89.3 $\pm$6.1
Turn Off Stove	31.0	25.0	27.3 $\pm$8.1	93.3 $\pm$5.8	28.7 $\pm$2.3	22.7 $\pm$22.0	16.7 $\pm$4.2
Turn On Microwave	91.5	95.0	89.3 $\pm$8.1	92.7 $\pm$4.6	93.3 $\pm$4.2	53.3 $\pm$2.3	72.7 $\pm$6.4
Turn On Sink Faucet	89.0	95.0	84.0 $\pm$8.0	18.7 $\pm$9.0	96.0 $\pm$4.0	33.3 $\pm$18.5	61.3 $\pm$12.9
Turn On Stove	76.5	85.0	72.0 $\pm$4.0	60.0 $\pm$3.5	70.7 $\pm$2.3	33.3 $\pm$11.5	41.3 $\pm$10.1
Turn Sink Spout	87.0	80.0	97.3 $\pm$3.1	93.3 $\pm$4.2	93.3 $\pm$5.8	45.3 $\pm$2.3	61.3 $\pm$6.1
Pick-and-Place	43.2	53.0	39.4 $\pm$1.0	47.0 $\pm$4.4	**49.8** $\pm$2.3	14.0 $\pm$1.7	**15.3** $\pm$1.4
Open-or-Close	81.0	80.8	75.9 $\pm$3.4	80.3 $\pm$2.3	**82.8** $\pm$1.3	55.1 $\pm$7.4	**56.1** $\pm$3.6
Others	75.8	79.0	76.7 $\pm$0.9	72.1 $\pm$2.0	**79.1** $\pm$1.5	39.5 $\pm$0.9	**53.5** $\pm$2.4
Average	66.2	70.8	64.1 $\pm$1.6	65.8 $\pm$0.8	**70.3** $\pm$1.7	34.9 $\pm$2.0	**41.4** $\pm$1.4

C.2 Qualitative Results

Figures 13 and 14 show representative rollouts from third-person, head, and wrist-camera views. Complete videos are available on the project page.

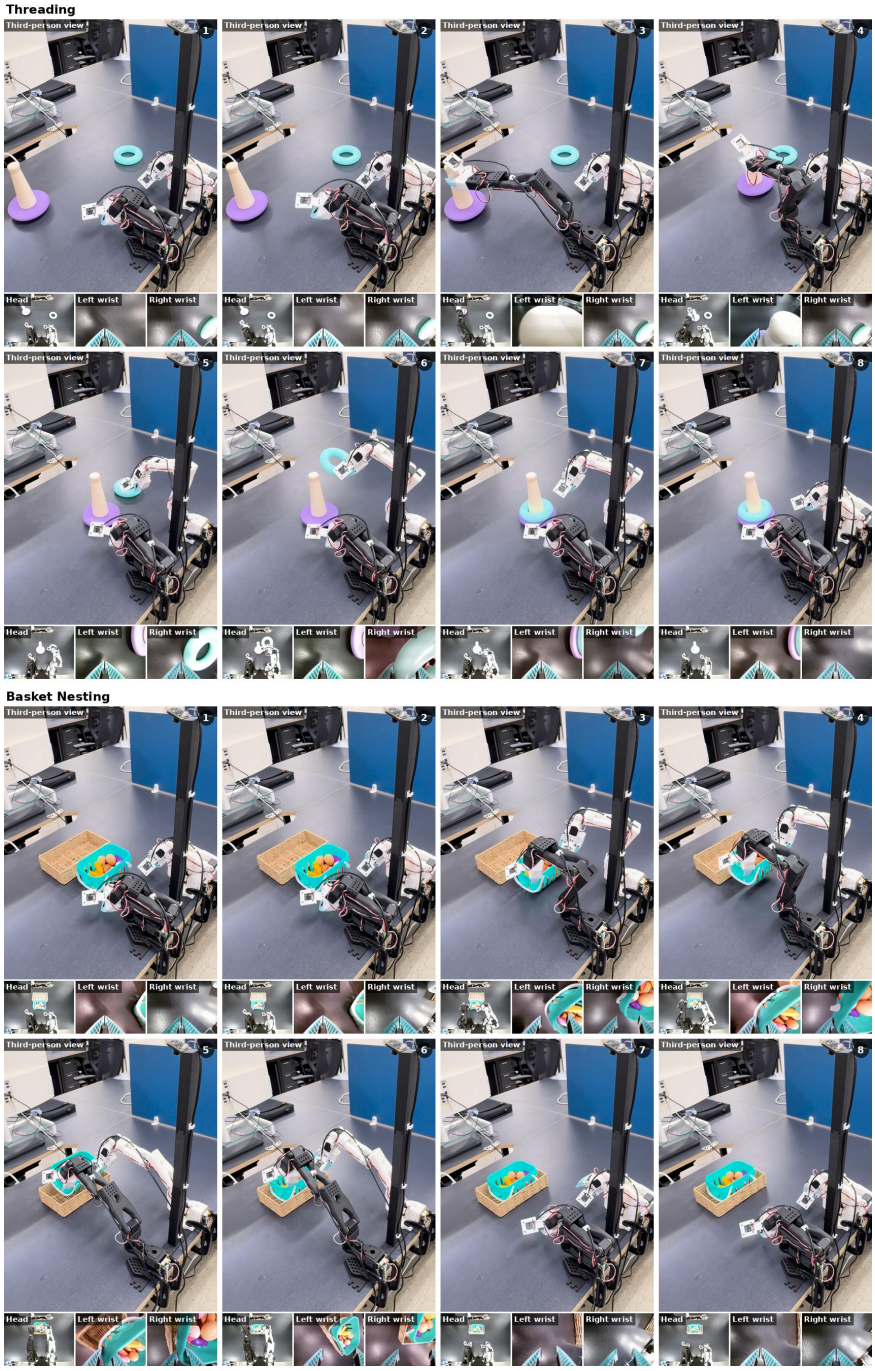


Figure 13: **Representative real-world rollouts on Threading and Basket Nesting.** The filmstrips show successful execution across the third-person, head, and wrist-camera views.

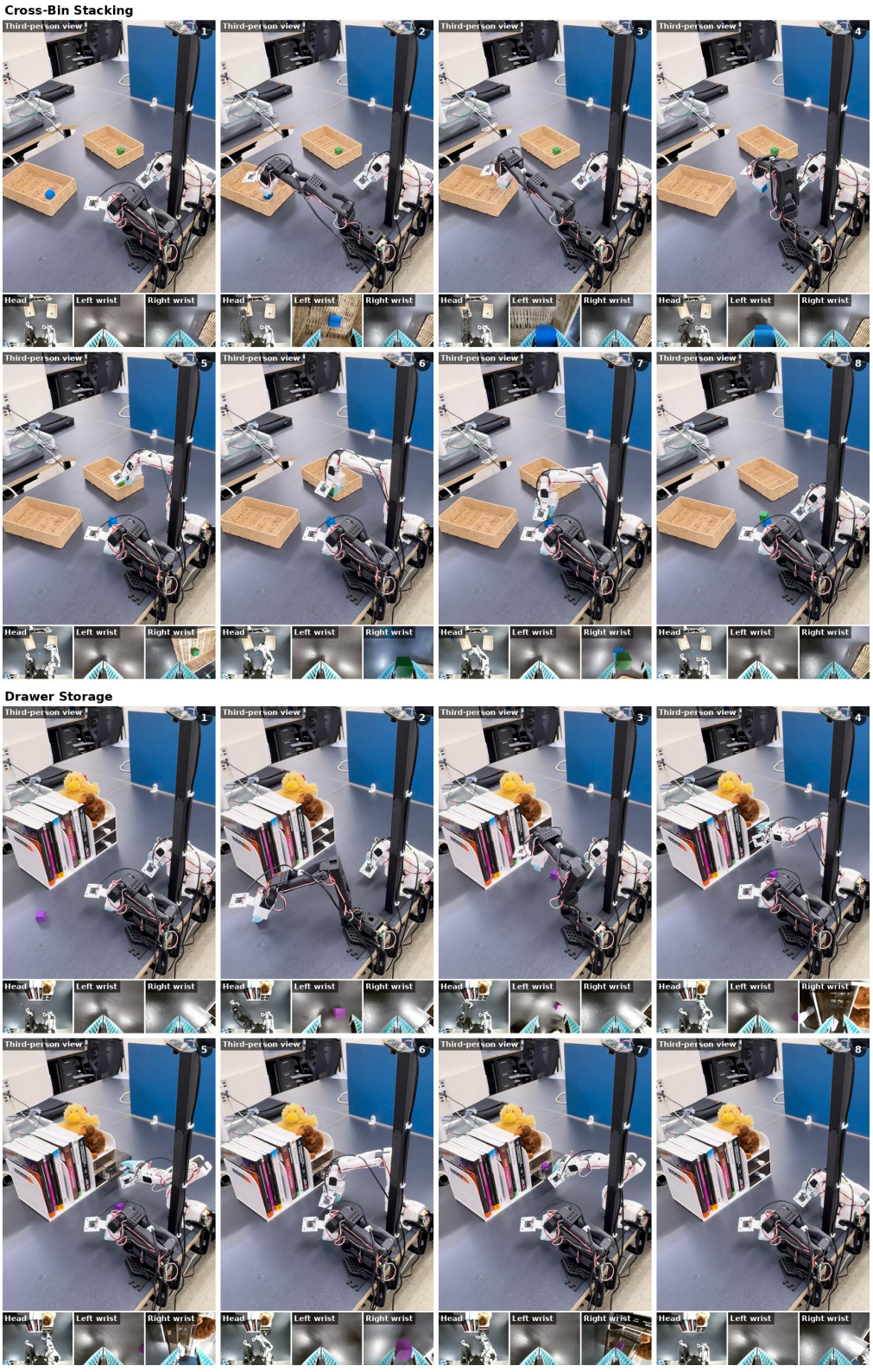


Figure 14: **Representative real-world rollouts on Cross-Bin Stacking and Drawer Storage.** The examples illustrate coordinated execution across the multiple stages required by the two longer-horizon tasks.

D LIMITATIONS AND FUTURE WORK

INDI learns from teacher representations that contain scene and motion cues together with information about purpose. The teacher and feature-source comparisons show that the choice of representation affects policy learning. Separately evaluating its objective, object, and scene information could help identify which teacher features are most useful for supervising the local objective.

At deployment, INDI forms one deterministic behavior state per query. Tracking and revising candidate objectives over time could extend this approach toward planning and human correction. The fixed-instruction replacement study (Table 16) highlights a further challenge: redirecting the policy through a state edit while preserving the ongoing execution and accounting for the observation and instruction that it still receives.

Our teacher targets use short windows from successful demonstrations. Incorporating failed and corrected executions, and linking segment objectives across temporal scales, would extend this supervision toward recovery behavior and longer plans.

E USE OF LARGE LANGUAGE MODELS

We used AI-assisted tools, including Claude Code and Codex, for code navigation and experiment orchestration, feedback on experimental design and interpretation of results, literature discovery and reference checking, and drafting and revising portions of the manuscript. The authors reviewed and verified the resulting text, references, implementation changes, experimental procedures, and reported results, and take responsibility for the final content. The Cosmos-Reason2 teacher generates the purpose statements and training targets described in Section A.4; this component of the proposed method is distinct from assistance with research and manuscript preparation.