OMNI-MONO: MUTUALLY INFORMED REASONING ACROSS VISION, LANGUAGE, AND ACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models increasingly use intermediate reasoning to guide robotic manipulation. However, existing approaches often restrict reasoning to one modality or a predefined sequence, limiting mutual feedback across modalities. We propose Omni-Mono, a framework combining joint multimodal reasoning with modality-specific refinement. The Omni Reasoner jointly generates textual, visual, and action representations in a shared latent space through grouped attention with modality relation masks, enabling information exchange across reasoning streams. The Mono Reasoner then refines these representations through dedicated experts, adaptively fusing guidance from Omni representations and preceding experts according to the current reasoning state. Together with the task context, the refined representations condition a flow-matching head to generate action chunks. Experiments on LIBERO, LIBERO-Plus, SimplerEnv, RoboTwin 2.0, and real-world manipulation tasks demonstrate improved performance and robustness to environmental perturbations. Ablations and representation analyses further support the complementary benefits of joint reasoning and modality-specific refinement. Our Code is available at `https://anonymous.4open.science/r/omni-mono/`.

1 INTRODUCTION

Vision-language-action (VLA) models connect the knowledge of pretrained vision-language models with robotic control, learning to map visual observations and natural-language instructions to actions (Zitkovich et al., 2023; Kim et al., 2025b). This paradigm provides a foundation for generalist manipulation, allowing robots to interpret diverse instructions and act in changing environments. As tasks become more complex, successful execution requires reasoning about what to do, how the scene should evolve, and which motions can realize the intended outcome (Zawalski et al., 2025; Zhao et al., 2025). Thus, intermediate reasoning bridges task intent and action execution, supporting the planning and control of complex tasks.

Inspired by chain-of-thought (CoT) reasoning in language models (Wei et al., 2022), recent VLAs introduce intermediate reasoning to connect task understanding with action generation. A common approach is textual reasoning, which guides action generation through textual subtask plans grounded in the observed scene (Zawalski et al., 2025; Sun et al., 2025). In addition, visual reasoning formulates planning as future-state prediction, using generated subgoal images (Zhao et al., 2025) or latent representations of future scenes (Zhang et al., 2025b) to guide action generation. More recently, a few works have explored reasoning directly in action space to bridge the gap between high-level semantics and low-level kinematics (Zhong et al., 2026a). In addition to single-modality reasoning, several works also seek to combine the strengths of different reasoning types, such as ManualVLA (Gu et al., 2026) and HALO (Shou et al., 2026), which integrate textual planning with visual subgoal prediction to guide action generation.

Despite these advances, existing methods often follow a loosely coupled reasoning paradigm, using separate models for modality-specific reasoning (Shou et al., 2026) or organizing cross-modal reasoning in a predefined order (Bai et al., 2026). However, considering that reasoning across different modalities is intended to solve the same task, it would be promising to enable collaborative rather than loosely coupled reasoning. Specifically, individual reasoning streams cannot fully attend to representations from other modalities, limiting their global understanding of the task. As shown in

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.

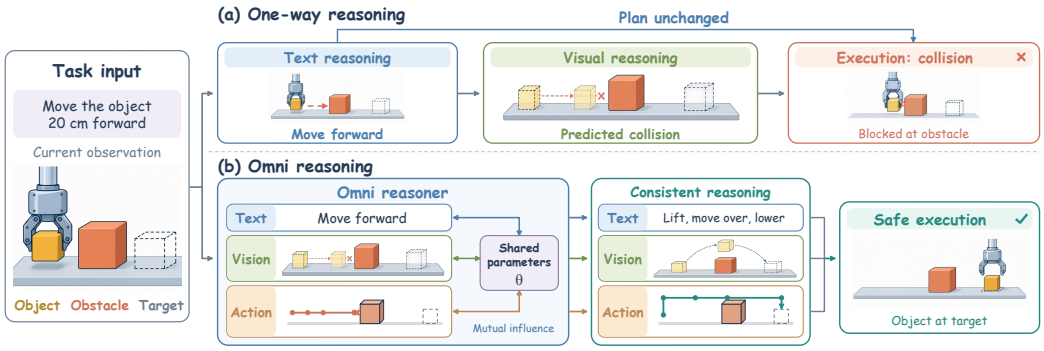


Figure 1: One-way vs. omni-modal reasoning. (a) One-way reasoning can retain inconsistent plans. (b) The Omni Reasoner coordinates textual, visual, and action reasoning through shared parameters and mutual information exchange.

Figure 1(a), textual reasoning may produce a plan to move an object straight ahead, which might be in conflict with the visual prediction of a collision with an obstacle along the path. In a one-way pipeline (Bai et al., 2026; Zhang et al., 2025a), visual reasoning cannot share this visual prediction with textual reasoning to adjust the plan, leaving the action head with inconsistent reasoning inputs. Cross-modal reasoning is therefore expected to form a mutually informative process that integrates their complementary reasoning representations into a global understanding of the task.

Unlike previous approaches that perform loosely coupled reasoning across modalities, we propose to simultaneously generate textual, visual, and action reasoning representations within a unified model, namely the **Omni Reasoner**. As illustrated in Figure 1(b), during reasoning in the Omni Reasoner, visual and textual reasoning implicitly exchange spatial predictions and semantic plans through the shared model, helping form an accurate plan that benefits from both modalities. Such collaborative reasoning helps the model handle more complex scenarios and guide actions that better account for task goals and environmental constraints.

Despite its appeal, training an Omni Reasoner presents two key challenges. First, compared with separate reasoners supervised for individual modalities, an Omni Reasoner jointly models reasoning states across text, vision, and action, expanding the space of reasoning states it needs to learn. Potential conflicts among modality-specific learning objectives further complicate this process (Hao et al., 2026). With limited training data, these demands can hinder stable convergence and compromise the quality of the resulting reasoning. Second, relying solely on an Omni Reasoner may cause modality-specific reasoning to be overshadowed by shared reasoning representations. Prior work on multimodal contrastive learning shows that emphasizing shared representations can discard features unique to individual modalities (Liang et al., 2023). Thus, a desirable Omni Reasoner is expected to preserve modality-specific details for semantic planning, visual anticipation, and motion reasoning.

To address these challenges, we propose **Omni-Mono**, a unified framework comprising an Omni Reasoner for holistic multimodal reasoning and a **Mono Reasoner** for modality-specific refinement. Through the collaboration between Omni and Mono Reasoners, our model maintains consistency across multimodal reasoning while preserving the distinctive characteristics of reasoning within each modality. Given the task inputs, the Omni Reasoner simultaneously produces latent reasoning representations for text, vision, and action. Within the shared model, the three reasoning streams implicitly exchange information as they are generated, allowing each stream to attend to the others and helping maintain coordination across modalities. The Mono Reasoner then uses these representations as coarse guidance for dedicated experts that progressively refine reasoning in text–vision–action order, with each expert adaptively drawing on reasoning representations according to its own preferences. Dedicated modeling allows each expert to retain the modality-specific features and develop the fine-grained details required by its modality. Finally, an action head takes the task context and all three refined Mono representations as input and predicts the final action chunks through conditional flow matching. Our main contributions are summarized as follows:

- As far as we know, we are the first to study a novel problem in the VLA domain: how to generate reasoning across multiple modalities within a unified model while allowing the reasoning streams to attend to one another.
- We propose Omni-Mono, which combines an Omni Reasoner for holistic multimodal reasoning with a Mono Reasoner comprising dedicated modality experts and adaptive guidance fusion.
- Experiments on LIBERO, LIBERO-Plus, SimplerEnv, and RoboTwin 2.0 demonstrate strong manipulation performance. Ablations and representation analyses support the complementary roles of joint reasoning and modality-specific refinement.

2 METHOD

In this section, we present the complete pipeline of our Omni-Mono framework (Figure 2). We first formulate the task and outline the framework in Sec. 2.1, then introduce the Omni Reasoner and Mono Reasoner in Secs. 2.2 and 2.3. Finally, we describe action generation in Sec. 2.4 and the multimodal supervision objectives in Sec. 2.5.

2.1 PROBLEM FORMULATION AND FRAMEWORK OVERVIEW

Problem formulation. We study language-conditioned robotic manipulation. At environment step t, a policy π_Θ receives a natural language instruction ℓ and the current visual observation o_t, and predicts an action sequence to accomplish the task:

$$\widehat{\mathbf{A}}_t = \pi_\Theta(o_t, \ell), \qquad \widehat{\mathbf{A}}_t = [\widehat{\mathbf{a}}_t; \ldots; \widehat{\mathbf{a}}_{t+H-1}], \tag{1}$$

where H is the action horizon and $\widehat{\mathbf{a}}_t \in \mathbb{R}^{d_a}$, with action dimension d_a determined by the robot's control interface.

Framework overview. The Omni-Mono framework integrates joint multimodal reasoning and modality-specific reasoning into action prediction through continuous latent representations. Given an observation image o_t and a language instruction ℓ, the framework first uses a vision-language encoder to construct a shared context sequence $\mathbf{C}_t$ for both reasoners.

The Omni Reasoner performs joint reasoning across text, vision, and action in a shared latent space, allowing the three modalities to exchange information. Conditioned on $\mathbf{C}_t$, it produces three groups of latent reasoning representations $\mathbf{G} = (\mathbf{G}_T, \mathbf{G}_V, \mathbf{G}_A)$:

$$\mathbf{G} = F_\theta^{\text{Omni}}(\mathbf{C}_t). \tag{2}$$

Building on $\mathbf{G}$ and the context $\mathbf{C}_t$, the Mono Reasoner performs modality-specific reasoning in the order of text, vision, and action, passing earlier reasoning outputs to subsequent experts. This process yields latent representations $\mathbf{Z} = (\mathbf{Z}_T, \mathbf{Z}_V, \mathbf{Z}_A)$:

$$\mathbf{Z} = F_\eta^{\text{Mono}}(\mathbf{C}_t, \mathbf{G}). \tag{3}$$

Finally, the action generation module combines the context $\mathbf{C}_t$ with the Mono Reasoner's outputs $\mathbf{Z}$ to predict the action sequence:

$$\widehat{\mathbf{A}}_t = \text{ActionGen}_\omega(\mathbf{C}_t, \mathbf{Z}). \tag{4}$$

In both reasoners, textual, visual, and action representations are produced by their corresponding reasoning branches and supervised using textual plans, dense future visual features, and sparse reference actions, respectively. We detail these supervision objectives in Sec. 2.5.

2.2 OMNI REASONER: JOINT MULTIMODAL REASONING

Textual, visual, and action reasoning provide complementary information for robotic manipulation. To couple these reasoning processes, the Omni Reasoner provides a shared latent space in which each modality updates its representation using the reasoning states of the other modalities.

First, we introduce a group of learnable queries $\mathbf{Q}_m$ for each modality $m \in \{T, V, A\}$, corresponding to text, vision, and action, respectively. After adding modality-type and slot embeddings, the three query groups read the context $\mathbf{C}_t$ and interact through a shared Transformer.

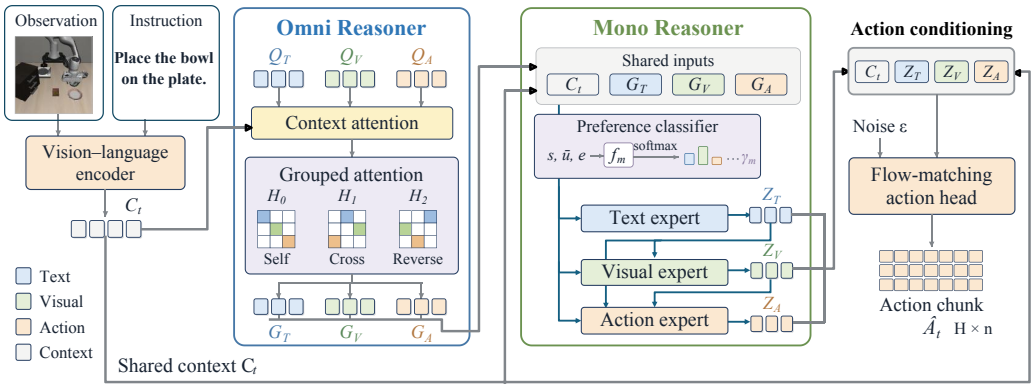


Figure 2: **Overview of Omni-Mono.** The Omni Reasoner jointly produces textual, visual, and action reasoning representations through grouped attention. The Mono Reasoner refines them in text–vision–action order using dedicated experts and adaptive guidance fusion. The shared context and three refined representations condition a flow-matching head that generates action chunks.

Grouped multi-head attention with modality relation masks. To explicitly guide the modeling of modality relations, we introduce a structured inductive bias into multi-head attention. We partition the heads into three groups, $\mathcal{H}_0$, $\mathcal{H}_1$, and $\mathcal{H}_2$, and use modality relation masks to assign each group a distinct interaction pattern. These masks constrain the source modality accessible to each query, directing the head groups to specialize in complementary dependencies. In $\mathcal{H}_0$, queries attend only to latent representations of their own modality. In $\mathcal{H}_1$, textual, visual, and action queries respectively attend to visual, action, and textual representations. And group $\mathcal{H}_2$ reverses these relations. The three groups collectively cover all intra-modal and directed cross-modal relations.

Specifically, each layer first reads the context and then computes these interactions. Let $\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h$ denote the query, key, and value projections of head h. For $h \in \mathcal{H}_r$, the message is

$$\mathbf{R}_h = \mathrm{softmax}_{\mathrm{key}}\left(\frac{\mathbf{Q}_h \mathbf{K}_h^\top}{\sqrt{d_h}} + \mathbf{M}_r^{\mathrm{rel}}\right)\mathbf{V}_h. \tag{5}$$

The relation mask $\mathbf{M}_r^{\mathrm{rel}}$ is 0 on allowed modality blocks and $-\infty$ elsewhere. Each head selects relevant content within its assigned source, and the head outputs are concatenated and projected to combine their messages. Across multiple layers, the Omni Reasoner produces reasoning representations $\mathbf{G}_T, \mathbf{G}_V, \mathbf{G}_A$, which provide multimodal context for the Mono Reasoner.

2.3 Mono Reasoner: Modality-Specific Reasoning

The Omni Reasoner brings together information from all three modalities, while its shared reasoning mechanism can limit the granularity of modality-specific representations. The Mono Reasoner builds on these joint representations through dedicated experts, each focused on reasoning about one target modality while drawing on multimodal information.

The Mono Reasoner comprises three Transformer experts E_T, E_V, E_A. Each starts from learnable queries $\mathbf{Q}_m^{\mathrm{Mono}}$ and reads both the context $\mathbf{C}_t$ and all Omni representations $\mathbf{G}$. The experts perform textual, visual, and action reasoning in sequence: the textual latent $\mathbf{Z}_T$ guides visual reasoning to produce $\mathbf{Z}_V$, and both latents subsequently guide action reasoning to generate $\mathbf{Z}_A$. This ordering follows a natural progression from task understanding to action generation.

Adaptive guidance fusion with source-wise retrieval. Considering that each expert may exhibit different modality preferences depending on its reasoning objective and the current task context, we introduce an adaptive fusion mechanism that weights multimodal guidance according to each expert's current query state. First, we treat the three Omni representations and the available outputs of preceding experts as separate sources, forming a set $\mathcal{R}_m$. Let $\mathbf{S}_m$ denote the current expert query states and $\mathbf{M}_r$ the representations from source $r \in \mathcal{R}_m$. At each layer, cross-attention CA first

retrieves a message from each source independently, followed by normalization:

$$\bar{\mathbf{U}}_{m,r} = \mathrm{Norm}_{m,r}\big(\mathrm{CA}_{m,r}(\mathbf{S}_m, \mathbf{M}_r, \mathbf{M}_r)\big). \tag{6}$$

Next, the expert estimates each source's contribution using the current query and retrieved content. For query $\mathbf{s}_{m,i}$, a scoring network f_m combines the retrieved message $\bar{\mathbf{u}}_{m,i,r}$ with a source embedding $\mathbf{e}_{m,r}$ and normalizes the scores across sources:

$$\gamma_{m,i,r} = \mathrm{softmax}_{r\in\mathcal{R}_m}\big[f_m(\mathbf{s}_{m,i}, \bar{\mathbf{u}}_{m,i,r}, \mathbf{e}_{m,r})\big]. \tag{7}$$

Finally, the retrieved messages are combined using these weights, allowing each query to select a mixture of information suited to its current reasoning state:

$$\mathbf{u}^{\mathrm{fuse}}_{m,i} = \sum_{r\in\mathcal{R}_m} \gamma_{m,i,r}\bar{\mathbf{u}}_{m,i,r}. \tag{8}$$

The context is read through a separate branch and combined with the fused messages to update the expert states. The resulting representations $\mathbf{Z}_T, \mathbf{Z}_V, \mathbf{Z}_A$ jointly condition the final action head.

2.4 Action Generation Conditioned on Reasoning Representations

The action generation module uses a Transformer conditioned on a sequence $\mathbf{M}_{\mathrm{act}}$ formed from the context $\mathbf{C}_t$ and reasoning representations $\mathbf{Z}_T, \mathbf{Z}_V, \mathbf{Z}_A$. Cross-attention reads this conditioning sequence, while self-attention models dependencies across action steps.

We train the module with conditional flow matching (Lipman et al., 2023). Given a normalized demonstration action chunk $\mathbf{A}^*_t$, we sample noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ and flow time $\tau \sim \mathcal{U}(0, 1)$, and construct $\mathbf{A}_\tau = (1-\tau)\boldsymbol{\epsilon} + \tau\mathbf{A}^*_t$. The action head D^{act}_ω predicts the velocity field that transforms noise into actions, with the objective

$$\mathcal{L}_{\mathrm{act}} = \mathbb{E}_{\tau,\boldsymbol{\epsilon}}\left[\frac{1}{d_a H}\left\|D^{\mathrm{act}}_\omega(\mathbf{A}_\tau, \tau \mid \mathbf{M}_{\mathrm{act}}) - (\mathbf{A}^*_t - \boldsymbol{\epsilon})\right\|^2_F\right]. \tag{9}$$

This objective trains the action generation module to combine the context and multimodal reasoning information to generate a complete sequence of H actions.

2.5 Multimodal Supervision and Joint Training

We define modality-specific supervision targets to train the textual, visual, and action latent representations. At time step t, the textual target y^T_t is generated by a language model from the demonstration segment. The visual target is the image from a fixed camera at $t + \Delta$. The reference action target is obtained by sampling the demonstration actions at a fixed interval. Omni and Mono use the same targets to learn multimodal reasoning. We use $k \in \{\mathrm{O}, \mathrm{M}\}$ to index the two reasoners and write $\mathbf{U}^{\mathrm{O}}_m = \mathbf{G}_m$ and $\mathbf{U}^{\mathrm{M}}_m = \mathbf{Z}_m$.

Textual supervision. The textual representation $\mathbf{U}^k_T$ is projected into a continuous prefix $\Pi^k_T(\mathbf{U}^k_T)$ for a frozen language decoder. Conditioned on this prefix, the decoder predicts the local textual plan y^T_t. We supervise the textual representation using the average token-level cross-entropy:

$$\mathcal{L}^k_T = -\frac{1}{N^T_t}\sum_{j=1}^{N^T_t} \log p_\phi\big(y^T_{t,j} \,\big|\, y^T_{t,<j}, \Pi^k_T(\mathbf{U}^k_T)\big)\,, \tag{10}$$

where N^T_t is the number of target tokens, $y^T_{t,<j}$ denotes the preceding target tokens, and ϕ denotes the frozen decoder parameters. This objective encourages the textual representation to encode the semantics of the operation plan.

Dense visual alignment through spatial queries. To preserve spatial details of the future scene in a compact visual representation, we decode dense features from the full set of visual representations using spatial queries, decoupling latent capacity from supervision resolution. A frozen visual teacher extracts the complete patch grid from $o_{t+\Delta}$ to form the target features $\mathbf{F}^*_t$. Each spatial location is

assigned a position query, collectively denoted by $\mathbf{Q}_{\mathrm{sp}}$. A decoder uses these queries to read the visual representations $\mathbf{U}_V^k$ and predict the features at each location:

$$\widehat{\mathbf{F}}_t^k = D_V^k\big(\mathbf{Q}_{\mathrm{sp}}, \Pi_V^k(\mathbf{U}_V^k)\big), \qquad \mathcal{L}_V^k = \frac{1}{Pd_v}\left\|\widehat{\mathbf{F}}_t^k - \mathbf{F}_t^*\right\|_F^2. \tag{11}$$

Here, Π_V^k is a projection layer, and P and d_v are the number of grid locations and the feature dimension, respectively. Predicting the complete grid supervises the visual representations across all spatial locations in the future scene.

Action representation supervision. To ground action reasoning in demonstrated motion, we sample H_{ref} actions at a fixed integer interval s to construct a sparse reference target:

$$\mathbf{A}_t^{\mathrm{ref},\star} = \mathcal{N}\Big([\mathbf{a}_t^\star; \mathbf{a}_{t+s}^\star; \ldots; \mathbf{a}_{t+(H_{\mathrm{ref}}-1)s}^\star]\Big), \tag{12}$$

where $\mathcal{N}$ is a fixed normalization transform based on training-set statistics. Each sampled action retains its original single-step meaning.

We define the action latent representation $\mathbf{U}_A^k$ as the output of each reasoner's action branch. Conditioned on $\mathbf{U}_A^k$, a reference action prediction module learns to generate $\mathbf{A}_t^{\mathrm{ref},\star}$ using the flow matching objective in Sec. 2.4, with loss $\mathcal{L}_A^k$. Gradients from this objective propagate through the prediction module to train $\mathbf{U}_A^k$ to encode motion information.

Joint optimization. The two reasoners, reference action modules, and final action generation module are trained jointly. Let $\overline{\mathcal{L}}_m^k$ denote the modality loss aggregated over valid samples, and $\overline{\mathcal{L}}_{\mathrm{act}}$ the corresponding aggregate of the final action loss. The joint objective is

$$\mathcal{L}_{\mathrm{total}} = \sum_{k\in\{\mathrm{O},\mathrm{M}\}} \beta_k \sum_{m\in\{T,V,A\}} \lambda_m \overline{\mathcal{L}}_m^k + \beta_{\mathrm{M}}\lambda_{\mathrm{act}}\overline{\mathcal{L}}_{\mathrm{act}}. \tag{13}$$

The weights β_k balance supervision across the two reasoners, while λ_m and λ_{act} weight the modality objectives and final action objective. Multimodal supervision constrains the semantic, spatial, and motion content of the reasoning representations, while the final action objective trains the model to use these representations jointly for action generation.

3 Experiments

3.1 Experimental Setup

Benchmarks. We evaluate Omni-Mono on four manipulation benchmarks: LIBERO (Liu et al., 2023), LIBERO-Plus (Fei et al., 2026), SimplerEnv (Li et al., 2024), and RoboTwin 2.0 (Chen et al., 2025). For LIBERO, we jointly train on 40 tasks across the Spatial, Object, Goal, and Long suites, using 50 demonstrations per task. LIBERO-Plus evaluates the LIBERO checkpoint zero-shot on 10,030 instances spanning seven perturbation categories. In SimplerEnv, we use four WidowX visual-matching tasks, with training on BridgeData V2 (Walke et al., 2023). For RoboTwin 2.0, we evaluate Aloha-AgileX policies on 50 tasks under clean(C2C) and randomized(C2R) conditions, using 50 clean training demonstrations per task (RoboTwin Team, 2026).

Baselines. Action-policy baselines include OpenVLA (Kim et al., 2025b), OpenVLA-OFT (Kim et al., 2025a), and $\pi_{0.5}$ (Physical Intelligence et al., 2025). Reasoning baselines include CoT-VLA (Zhao et al., 2025), ThinkAct (Huang et al., 2025), Fast-ThinkAct (Huang et al., 2026), ACoT-VLA (Zhong et al., 2026a), LaRA-VLA (Bai et al., 2026), and DualCoT-VLA (Zhong et al., 2026b). Together, these baselines cover direct action prediction and reasoning over textual, visual, and action representations in explicit or latent forms.

Implementation Details. Omni and the three Mono experts use independent language backbones initialized from Qwen3-VL-4B-Instruct (Bai et al., 2025), augmented with relation-masked attention for Omni and guidance attention for Mono. A supervised cold-start stage precedes joint fine-tuning. Frozen Qwen3-0.6B (Yang et al., 2025) and DINOv2 ViT-B/14 (Oquab et al., 2023) provide textual decoding and visual supervision, respectively. Architecture and training settings are detailed in Appendices A.1 and A.2.

Table 1: Success rates (%) on four simulation benchmarks (higher is better). **Bold** and underlined values indicate the best and second-best results in each column, respectively.

Method	LIBERO					LIBERO-Plus		SimplerEnv	RoboTwin 2.0	
	Spatial	Object	Goal	Long	Avg.	Avg.	Min	Avg.	C2C	C2R
Action policies										
OpenVLA	84.7	88.4	79.2	53.7	76.5	15.6	3.3	14.6	48.6	26.4
OpenVLA-OFT	97.6	98.4	97.9	94.5	97.1	69.6	31.9	64.6	79.8	57.6
OFT (joint LIBERO)	97.7	98.0	96.1	95.3	96.8	71.2	36.1	64.6	79.8	57.6
OFT (1 view)	96.9	97.4	96.2	92.7	95.8	55.8	10.4	58.3	75.4	51.8
$\pi_{0.5}$	98.8	98.2	98.0	92.4	96.9	85.7	75.8	70.8	84.8	68.4
Reasoning policies										
CoT-VLA	87.5	91.6	87.6	69.0	83.9	47.8	8.5	39.6	63.2	40.8
ThinkAct	88.3	91.4	87.1	70.9	84.4	50.2	9.8	43.8	66.4	45.6
Fast-ThinkAct	92.0	97.2	90.2	79.4	89.7	64.8	24.1	56.3	74.6	55.8
ACoT-VLA	98.6	99.0	**99.4**	97.0	98.5	86.6	72.6	71.9	85.2	69.6
LaRA-VLA	96.4	**99.8**	98.6	96.6	97.9	83.2	67.2	68.8	83.6	67.4
DualCoT-VLA	99.4	**99.8**	97.8	98.2	98.8	87.2	77.4	73.6	86.8	71.2
Omni–Mono	**99.6**	99.5	**99.4**	**98.8**	**99.3**	**92.4**	**79.6**	**78.3**	**90.6**	**77.2**

Table 2: Ablation SR (%). SGW: static guidance weights.

Method	LIBERO+ Avg. ↑	Simpler Avg. ↑	RoboTwin C2R ↑
Omni–Mono (Ours)	**92.4**	**78.3**	**77.2**
Omni-only	87.4	73.8	71.4
Mono-only	88.0	74.4	72.1
Dense attention	90.8	77.0	75.6
SGW	89.9	76.1	74.6

Table 3: Representation quality on held-out LIBERO trajectories.

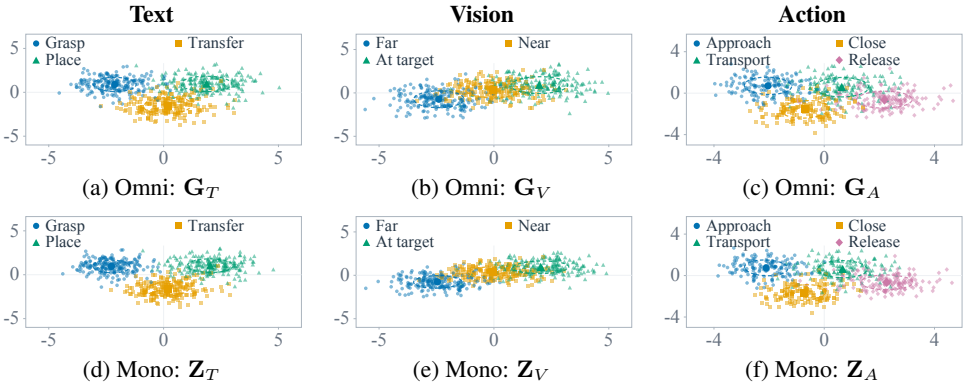
	Text		Vision	Action
Representation	Rel. NLL ↓	Acc. (%) ↑	Rel. MSE ↓	Rel. RMSE ↓
Omni-only: $\mathbf{G}$	1.00	78.0	1.00	1.00
Omni (MAP): $\mathbf{G}$	0.94	81.0	0.93	0.95
Omni–Mono: $\mathbf{G}$	0.96	80.0	0.96	0.94
Omni–Mono: $\mathbf{Z}$	0.81	87.0	0.78	0.83

3.2 Main Results

Table 1 shows that Omni–Mono achieves the highest reported average success rates on LIBERO, LIBERO-Plus, SimplerEnv, and RoboTwin. Its margin over the strongest compared baseline is wider on LIBERO-Plus without additional fine-tuning and in randomized RoboTwin environments than in the corresponding standard or clean settings. The long-horizon result suggests that connecting task understanding with visual and action reasoning helps sustain goal-consistent behavior across successive manipulation steps. The larger margins under perturbations point to an advantage in adapting execution when scene conditions change. Together, these observations suggest that joint reasoning and modality-specific refinement complement each other in maintaining task intent while adjusting actions to the observed scene. Benchmark breakdowns are provided in Appendix F.

3.3 Ablation Study

Omni–Mono outperforms both Omni-only and Mono-only on all three benchmarks in Table 2, supporting the complementary roles of joint multimodal reasoning and modality-specific refinement. The component ablations retain both stages and test the mechanisms used to guide their learning. Replacing modality relation masks with dense attention or adaptive guidance with static guidance weights (SGW) consistently lowers SR under the same training budget. These results indicate that the proposed mechanisms improve the performance reached with a fixed optimization budget. Variant definitions and training controls are in Appendix D.



(a) Omni: $\mathbf{G}_T$ (b) Omni: $\mathbf{G}_V$ (c) Omni: $\mathbf{G}_A$

(d) Mono: $\mathbf{Z}_T$ (e) Mono: $\mathbf{Z}_V$ (f) Mono: $\mathbf{Z}_A$

Figure 3: Task-relevant latent structure before and after refinement.

3.4 Representation Quality and Latent Structure

Representation quality. On held-out LIBERO trajectories, matched probes show that frozen $\mathbf{Z}$ improves plan accuracy over $\mathbf{G}$ from 80.0% to 87.0% and lowers text NLL, future-feature MSE, and action-reference RMSE (Table 3). $\mathbf{Z}$ also outperforms Omni (MAP), an Omni-only control with matched added attention parameters, on all four metrics, with plan accuracy 6.0 points higher. Under matched targets, the gains suggest better probe access to supervised information (Appendix E.1).

Task-relevant latent structure. Figure 3 shows two-dimensional latents colored by operation, object–target relation, and control state. Refinement tightens textual grasp, transfer, and place groups. Visual latents retain the far–near–at-target progression; action latents distinguish approach, close, transport, and release. Neighboring states still overlap despite tighter task-related structure.

3.5 Hyperparameter Analysis

Query capacity. We scale the latent query budgets of both reasoning stages together while fixing supervision resolution. Success peaks at the default budget, while latency grows throughout the sweep (Figure 4(a)). This suggests that smaller budgets constrain reasoning capacity, whereas additional queries beyond the default increase computation without improving task success.

Auxiliary weight. We vary a shared weight for textual, visual, and action supervision while fixing the final-action weight. Success peaks at the default, with nearby settings remaining competitive (Figure 4(b)). The decline with weaker or stronger weights suggests that auxiliary supervision must be balanced with learning the final action policy.

Joint horizons. The visual offset sets how far ahead the model predicts, while the action-reference stride controls temporal coverage at fixed reference length. Their default combination performs best (Figure 4(c)). Shorter or longer offsets and denser or sparser references lower success, suggesting a balance between future context and motion guidance for local execution.

3.6 Inference Efficiency

As shown in Figure 5, Omni–Mono has an average inference latency of 680 ms on a single A100 GPU. Its latency is lower than the published values for explicit CoT methods and 15.5% below the latent reasoning baseline Fast-ThinkAct (Bai et al., 2026, Fig. 8). The two stages add computation over Base, but fixed latent queries feed reasoning representations directly into action generation, avoiding autoregressive textual-plan decoding. The textual decoder and visual and action supervision heads are used only during training and add no inference cost. Appendix E.4 details the timing protocol and scope of the published comparisons.

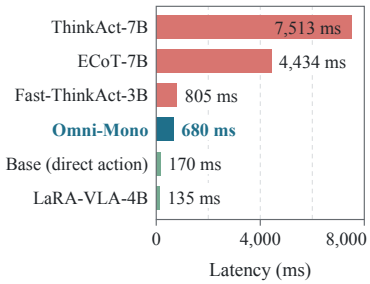


Figure 5: Inference latency.

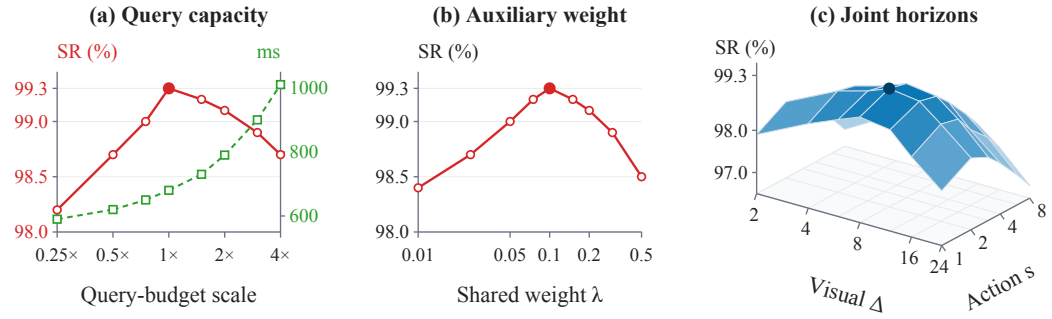


Figure 4: Hyperparameter sensitivity on LIBERO.

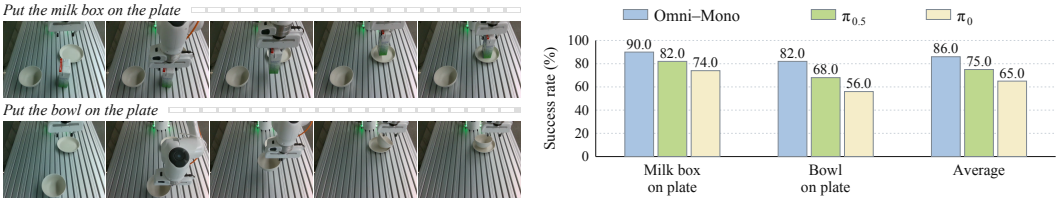


Figure 6: Real-world execution sequences for placing a milk box onto a plate (top left) and a bowl onto a plate (bottom left). Success rates (%, right) use 50 trials per task and method; the average weights the two tasks equally.

3.7 Real-World Experiments

We evaluate Omni–Mono on two tabletop pick-and-place tasks using a Franka-r3 arm with a parallel-jaw gripper. We compare against π_0 (Black et al., 2024) and $\pi_{0.5}$ (Physical Intelligence et al., 2025). The tasks involve placing a milk box onto a plate and placing a bowl onto a plate. All methods use the same demonstrations, observation and action interfaces, and initial configurations. Success requires the instructed object to rest stably at the target after release without human intervention. Evaluation details are provided in Appendix C.5.

Figure 6 shows that Omni–Mono achieves the highest success rate on both tasks: 90.0% for placing the milk box onto the plate and 82.0% for placing the bowl onto the plate, exceeding $\pi_{0.5}$ by 8.0 and 14.0 percentage points, respectively. The larger margin occurs on the bowl task, which requires a rim grasp followed by stable release, coupling grasp selection with placement control. The execution sequences illustrate the progression from grasping through transport to release. Since success is judged after release, the gains reflect more reliable completion of the full placement sequence. These results suggest that joint multimodal reasoning and modality-specific refinement help maintain task intent throughout physical execution.

4 Conclusion

In this work, we propose Omni–Mono, a framework that combines joint multimodal reasoning with modality-specific refinement for vision-language-action models. The Omni Reasoner jointly generates latent reasoning representations for text, vision, and action through structured cross-modal interactions, allowing the three reasoning streams to inform one another. The Mono Reasoner then uses dedicated modality experts and adaptive guidance fusion to refine these representations for action generation. Experiments on four simulation benchmarks and real-world manipulation tasks demonstrate improved manipulation performance and robustness to environmental perturbations. Ablations and representation analyses further support the complementary benefits of joint multimodal reasoning and modality-specific refinement.

AI USE STATEMENT

Generative AI tools were used to polish the text, assist with writing the experimental code, and search for and study relevant literature. All AI-assisted content was reviewed by the authors, who take full responsibility for the final content of this paper.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, et al. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

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. In *Proceedings of the International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.01166.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael 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, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

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

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38574–38583, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Fei_LIBERO-Plus_A_Progressive_Robustness_Benchmark_for_Visual-Language-Action_Models_CVPR_2026_paper.html.

Chenyang Gu, Jiaming Liu, Hao Chen, Runzhong Huang, Qingpo Wuwu, Xiaoqi Li, Zhuoyang Liu, Ying Li, Renrui Zhang, Peng Jia, Pheng-Ann Heng, and Shanghang Zhang. From manuals to actions: A unified VLA model for chain-of-thought manual generation and robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2512.02013.

Jitai Hao, Hao Liu, Xinyan Xiao, Qiang Huang, and Jun Yu. Uni-X: Mitigating modality conflict with a two-end-separated architecture for unified multimodal models. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/c8c7ae251c1d2241d0f1186ea90bf7c9-Abstract-Conference.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*, 2025. URL https://arxiv.org/abs/2507.16815.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2601.09708.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems*, 2025a. URL https://arxiv.org/abs/2502.19645.

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, 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, 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. *arXiv preprint arXiv:2405.05941*, 2024. URL https://arxiv.org/abs/2405.05941.

Paul Pu Liang, Zihao Deng, Martin Q. Ma, James Y. Zou, Louis-Philippe Morency, and Ruslan Salakhutdinov. Factorized contrastive learning: Going beyond multi-view redundancy. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/6818dcc65fdf3cbd4b05770fb957803e-Abstract-Conference.html.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision, 2023. URL https://arxiv.org/abs/2304.07193.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael 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, 2025. URL https://arxiv.org/abs/2504.16054.

RoboTwin Team. RoboTwin 2.0 leaderboard: Benchmark setting, 2026. URL https://robotwin-platform.github.io/leaderboard. Aloha-AgileX, 50 clean demonstrations per task, clean-to-clean and clean-to-random evaluation; accessed September 21, 2026.

Quanxin Shou, Fangqi Zhu, Shawn Chen, Puxin Yan, Zhengyang Yan, Yikun Miao, Xiaoyi Pang, Zicong Hong, Ruikai Shi, Hao Huang, Jie Zhang, and Song Guo. HALO: A unified vision-language-action model for embodied multimodal chain-of-thought reasoning, 2026. URL https://arxiv.org/abs/2602.21157.

Qi Sun, Pengfei Hong, Tej Deep Pala, Vernon Toh, U-Xuan Tan, Deepanway Ghosal, and Soujanya Poria. Emma-X: An embodied multimodal action model with grounded chain of thought and look-ahead spatial reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14199–14214. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.695. URL https://aclanthology.org/2025.acl-long.695/.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A dataset for robot learning at scale. In *Conference on Robot Learning*, 2023. URL https://rail-berkeley.github.io/bridgedata/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://papers.nips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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. URL https://proceedings.mlr.press/v270/zawalski25a.html.

Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. UP-VLA: A unified understanding and prediction model for embodied agent. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 74911–74922. PMLR, 2025a. URL https://proceedings.mlr.press/v267/zhang25w.html.

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, 2025b. doi: 10.52202/085713-0818. URL https://proceedings.nips.cc/paper_files/paper/2025/hash/22d4f952efa13970f0b1ffb22170d416-Abstract-Conference.html.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/abs/2503.22020.

Linqing Zhong, Yi Liu, Yifei Wei, Ziyu Xiong, Maoqing Yao, Si Liu, and Guanghui Ren. ACoT-VLA: Action chain-of-thought for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026a. URL https://arxiv.org/abs/2601.11404.

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, 2026b. 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 Implementation and Training Details

A.1 Model Architecture and Information Flow

The policy receives the current RGB observation o_t and task instruction ℓ. A vision-language encoder forms a context sequence $\mathbf{C}_t$, which is available to the Omni Reasoner, each Mono expert, and the final action head. Reasoning is represented by continuous latent sequences throughout the policy forward pass. The text decoder, future-feature decoder, and reference-action heads supply auxiliary training signals; the policy predicts executable actions through the final action head.

Omni and the three Mono experts have independent language backbones initialized from Qwen3-VL-4B-Instruct (Bai et al., 2025). Their language parameters are fully fine-tuned and are not tied across experts. Each backbone retains the 36 pretrained language Transformer blocks with hidden width 2560. The added interaction modules operate at width 768 with 12 attention heads, using projections between their internal states and the backbone states. Relation-masked attention is inserted before Omni blocks 9, 18, 27, and 36. Guidance attention is inserted before blocks 18 and 36 of each Mono expert. The attention updates in Secs. 2.2 and 2.3 describe these interaction layers; the remaining backbone blocks carry out the intervening transformations.

The default query counts are $(n_T, n_V, n_A) = (8, 16, 8)$ in each reasoning stage. Thus, $\mathbf{G}_T, \mathbf{G}_V, \mathbf{G}_A$ and $\mathbf{Z}_T, \mathbf{Z}_V, \mathbf{Z}_A$ retain separate modality and slot identities. Textual queries encode local operation semantics, visual queries summarize the future scene, and action queries encode sparse motion references. The visual latent has 16 slots, while the supervision target contains the complete 16×16 teacher patch grid. These are different quantities: spatial queries in the visual decoder recover dense features from a compact latent sequence.

Omni relation masks. Table 4 makes the allowed query–source relations explicit. Each of the three head groups contains four heads. For a head in group $\mathcal{H}_r$, a query of modality m reads only the source modality $\sigma_r(m)$ shown in the table. Within that source modality, attention weights are learned over all available latent slots. Across the three groups, every query modality can receive messages from text, vision, and action. The relation mask acts on latent-to-latent attention; the context-reading branch supplies observation and instruction information.

Table 4: Allowed source modality for each Omni query modality and head group. Rows identify queries and columns identify head groups.

Query modality	$\mathcal{H}_0$	$\mathcal{H}_1$	$\mathcal{H}_2$
Text	Text	Vision	Action
Vision	Vision	Action	Text
Action	Action	Text	Vision

Mono guidance sources. The experts run in text–vision–action order. Their source sets are

$$\begin{aligned} \mathcal{R}_T &= \{\mathbf{G}_T, \mathbf{G}_V, \mathbf{G}_A\}, \\ \mathcal{R}_V &= \{\mathbf{G}_T, \mathbf{G}_V, \mathbf{G}_A, \mathbf{Z}_T\}, \\ \mathcal{R}_A &= \{\mathbf{G}_T, \mathbf{G}_V, \mathbf{G}_A, \mathbf{Z}_T, \mathbf{Z}_V\}. \end{aligned} \tag{14}$$

Each source is retrieved and normalized separately before its message is weighted. This gives three, four, and five guidance sources to the text, visual, and action experts, respectively. Softmax normalization is over sources for each query, so $\sum_{r \in \mathcal{R}_m} \gamma_{m,i,r} = 1$. The context $\mathbf{C}_t$ is read through a separate branch and is not an additional member of this softmax. Earlier Mono outputs remain available to subsequent experts within the same policy call. The final action head reads the conditioning sequence formed from $\mathbf{C}_t, \mathbf{Z}_T, \mathbf{Z}_V, \mathbf{Z}_A$.

Shared context and parameter ownership. Sharing the context sequence means that all reasoning modules condition on the same observation and instruction; it does not imply that their language backbones share trainable weights. The frozen auxiliary text decoder is Qwen3-0.6B (Yang et al., 2025), and the frozen visual teacher is DINOv2 ViT-B/14 (Oquab et al., 2023). Gradients through the text prefix reach the reasoning latents while decoder parameters remain fixed. The visual teacher provides target features without receiving updates.

Module specifications. We use a single shared, frozen input vision tower initialized from the same Qwen3-VL checkpoint, with a trainable visual projection and instruction embedding module producing width-2560 context tokens. These input modules are separate from the four independently fine-tuned reasoning backbones. Cache $\mathbf{C}_t$ once per policy call and project it to width 768 inside each interaction module. The auxiliary DINOv2 teacher is a separate frozen network. In the reasoning backbones, cross-modality latent communication is restricted to Omni's relation-masked insertions; intervening pretrained self-attention blocks use within-modality latent masks and read $\mathbf{C}_t$. Context tokens cannot read latent tokens, preventing an indirect route around the relation masks. Each Mono backbone updates its own query group; the source-wise guidance insertions supply the other latent groups.

For the auxiliary heads, use separate Omni and Mono linear text-prefix projections from 2560 to the frozen decoder's width of 1024. Use a four-layer visual Transformer decoder with width 768, 12 heads, and feed-forward width 3072; its 256 spatial queries predict 768-dimensional teacher features. Use six-layer Transformers of the same width, head count, and feed-forward width for each reference-action predictor and the final action head, with a flow-time embedding and an output projection to the dataset action dimension. The two reference predictors have separate parameters; their conditioning sequences contain only $\mathbf{G}_A$ or $\mathbf{Z}_A$, respectively. The final head reads $\mathbf{C}_t, \mathbf{Z}_T, \mathbf{Z}_V, \mathbf{Z}_A$. Use a two-layer guidance scorer of widths $2304 \to 256 \to 1$ with SiLU activation on the concatenated query, normalized source message, and width-768 source embedding. Each guidance layer has its own scorer, shared across that layer's sources.

A.2 Training Schedule and Data Splits

Training starts with a separate supervised cold-start stage and then optimizes the reasoning modules and action predictors jointly. Table 5 gives the model and optimization settings. The listed update counts refer to joint training and exclude cold-start updates. Pretrained backbones and newly introduced modules use separate learning-rate groups. The frozen text decoder and visual teacher are excluded from optimizer updates.

Table 5: Architecture, supervision, and joint-training configuration.

Configuration	Value
Backbone initialization	Qwen3-VL-4B-Instruct
Text decoder / visual teacher	Qwen3-0.6B / DINOv2 ViT-B/14, frozen
Input resolution / teacher patch grid	224×224 / 16×16
Added attention width / heads	768 / 12
Omni insertions / insertions per Mono expert	4 / 2
Text / visual / action queries per stage	8 / 16 / 8
Predicted action horizon	$H = 8$
Executed actions per call	8 (LIBERO), 4 (WidowX), 4 (RoboTwin)
Reference length / stride	$H_{\text{ref}} = 8$, $s = 2$
Visual target offset	$\Delta = 8$ recorded steps
Final flow integration	10 Euler steps
Optimizer / global batch size	AdamW / 64
Training hardware	$8\times$ NVIDIA A100 80GB
Backbone / new-module learning rate	10^{-5} / 10^{-4}
Joint updates: LIBERO / Bridge / RoboTwin	30,000 / 50,000 / 60,000
Stage weights	$\beta_{\text{O}} = \beta_{\text{M}} = 1$
Auxiliary weights	$\lambda_T = \lambda_V = \lambda_A = 0.1$
Final-action weight	$\lambda_{\text{act}} = 1$

With these weights, the objective in Eq. 13 reduces to

$$\mathcal{L}_{\text{total}} = 0.1 \sum_{k \in \{\text{O},\text{M}\}} \left(\overline{\mathcal{L}}_T^k + \overline{\mathcal{L}}_V^k + \overline{\mathcal{L}}_A^k \right) + \overline{\mathcal{L}}_{\text{act}}. \tag{15}$$

The six auxiliary terms supervise both reasoning stages, while the final action loss trains their use in control. Equal auxiliary coefficients specify loss weights; they do not imply equal numerical loss magnitudes or gradient norms. The token-, patch-, and action-dimension normalizations in Sec. 2.5 determine the scale of each term.

LIBERO training combines all 40 tasks with 50 demonstrations per task. RoboTwin training combines all 50 Aloha-AgileX tasks with 50 clean demonstrations per task. Sampling is task-balanced within these sources. BridgeData V2 retains its training split and released language instructions. Action normalization is fitted separately for each source using training data. SimplerEnv evaluation trajectories, LIBERO-Plus augmentation data, and RoboTwin randomized demonstrations are excluded from the corresponding training sets.

Hyperparameters are selected on trajectory-level development splits. Final LIBERO and RoboTwin policies are retrained using their full 50-demonstration budgets. The held-out representation study uses a separate split, described in Appendix E.1. Internal simulation policies use three training seeds per source; the LIBERO ablations use the same three seeds. Dataset adapters and joint training schedules change with the source, while the reasoning structure and auxiliary objectives remain the same.

Cold-start and optimization settings. We allocate 3,000, 5,000, and 6,000 cold-start updates to LIBERO, Bridge, and RoboTwin, respectively, followed by the joint-update budgets in Table 5. During cold-start, freeze the pretrained reasoning backbones and input vision tower, and train the latent queries, context projections, interaction modules, prefix projections, auxiliary predictors, and final action head with Eq. 15. Then unfreeze all four reasoning backbones for joint fine-tuning. Keep the input vision tower, text decoder, and visual teacher frozen in both phases; the trainable context modules remain active throughout. No auxiliary decoder output is fed to the policy as a teacher-forced reasoning state.

Use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and weight decay 0.01, excluding biases and normalization parameters from decay. Warm up linearly over the first 5% of updates in each phase and then use cosine decay to 10% of the peak learning rate. Reset the optimizer at the phase transition. Clip the global gradient norm to 1.0 and use BF16 computation with sharded optimizer states and activation checkpointing. A per-GPU microbatch of one with eight accumulation steps on eight GPUs gives the existing global batch size of 64.

Use trajectory-split seed 42 and training seeds $\{0, 1, 2\}$. For model selection, reserve five of the 50 demonstrations per task for development in LIBERO and RoboTwin, leaving 45 for training. For Bridge, reserve 5% of trajectories from the released training split, grouped by episode. Fit all normalization statistics on the training portion of each split. Evaluate development loss every 1,000 updates and select hyperparameters by the lowest task-balanced final-action flow loss, averaged over fixed noise/time draws and training seeds; break ties in favor of the smaller configuration. Select the update budget on development data, then retrain with the full available training split and use the final checkpoint at that budget. Benchmark test rollouts do not select checkpoints. The separate 40/5/5 diagnostic split is retained for representation studies.

A.3 Action Interfaces and Execution

LIBERO and Bridge use seven-dimensional actions containing translation, rotation, and a gripper command. RoboTwin uses fourteen-dimensional joint-position commands: six arm joints and one gripper value for each arm. Each dataset adapter preserves the source coordinate frame, rotation convention, and gripper semantics. State used for offline action conversion is not an additional policy input. The policy receives the current image and instruction, as defined in Eq. 1.

The final head predicts a contiguous chunk of $H = 8$ actions. At inference, flow integration begins from Gaussian noise $\mathbf{A}^{(0)}$ and applies the velocity predictor conditioned on $\mathbf{M}_{\text{act}}$. For ten uniform Euler steps, the update is

$$\mathbf{A}^{(j+1)} = \mathbf{A}^{(j)} + \frac{1}{10} D_{\omega}^{\text{act}} \left(\mathbf{A}^{(j)}, \frac{j}{10} \,\middle|\, \mathbf{M}_{\text{act}} \right), \qquad j = 0, \ldots, 9. \tag{16}$$

The resulting normalized actions are mapped back to the dataset's action space. LIBERO executes all eight actions before replanning; WidowX and RoboTwin execute the first four. The WidowX interface runs at 5 Hz. An execution chunk can therefore be shorter than the prediction horizon without changing the shape of the training target.

One policy call follows this sequence:

1. Encode the current image and task instruction into $\mathbf{C}_t$.
2. Update Omni queries to obtain $\mathbf{G}_T, \mathbf{G}_V, \mathbf{G}_A$ using the relation masks.
3. Compute $\mathbf{Z}_T$, then $\mathbf{Z}_V$, then $\mathbf{Z}_A$ using the source sets in Eq. 14.
4. Condition the final action head on $\mathbf{C}_t, \mathbf{Z}_T, \mathbf{Z}_V, \mathbf{Z}_A$ and integrate its velocity field to obtain an action chunk.
5. Convert the chunk to the robot interface, execute the prescribed prefix, and acquire the next observation.

Auxiliary textual-plan decoding, future-feature reconstruction, and sparse reference-action prediction are omitted from this inference sequence.

Normalization convention. For each continuous action coordinate d, let $q_{.01,d}$ and $q_{.99,d}$ be its training-set 1st and 99th percentiles. Use

$$\mathcal{N}_d(a) = 2\frac{\mathrm{clip}(a, q_{.01,d}, q_{.99,d}) - q_{.01,d}}{q_{.99,d} - q_{.01,d}} - 1, \tag{17}$$

and invert this affine map after clipping the generated coordinate to $[-1, 1]$. A constant coordinate maps to zero and decodes to its constant training value. This convention applies coordinate-wise to Cartesian increments and to absolute joint-position targets without changing their semantics. Map binary gripper labels to $\{-1, +1\}$ using each dataset's open/close convention and threshold the prediction at zero. For a continuous gripper aperture, apply the percentile map and retain the continuous output. Use the same transform for sparse-reference and final action targets; apply the robot interface's physical limits after inverse transformation.

B Constructing Multimodal Supervision

B.1 Trajectory Alignment and Target Indexing

Training examples are indexed by a trajectory and a current time step. The selected fields are LIBERO's `agentview_rgb` and `actions`, Bridge's `image_0` and `action`, and RoboTwin's `head_camera/rgb` and `joint_action/vector`. These provide the agentview, external, and head-camera observations, respectively. Encoded RoboTwin images are decoded using the release's image convention. Current observations and future image targets use the same selected camera and preprocessing; wrist views are excluded from these inputs.

Trajectory preprocessing precedes target indexing. Bridge uses the original-data transform described with OpenVLA (Kim et al., 2025b): the initial step is removed, gripper values are binarized, and movement actions are relabeled from differences between adjacent measured end-effector states. The last step without a successor is dropped. Observation, instruction, action, and episode-index fields are sliced together. The time index used for target extraction refers to this aligned trajectory, so a future image and an action reference cannot silently use different offsets caused by preprocessing.

Each training example contains three temporal targets. The final control target is the contiguous chunk $[\mathbf{a}^\star_t; \ldots; \mathbf{a}^\star_{t+H-1}]$. The visual target is the image at $t + \Delta$. The action-reasoning target samples $t, t+s, \ldots, t+(H_{\mathrm{ref}}-1)s$. With the default settings, the final action chunk ends at $t+7$, the future visual target is at $t+8$, and the sparse action reference ends at $t+14$. Their temporal extents need not coincide: they supervise local execution, scene anticipation, and coarser motion information, respectively.

B.2 Textual Plans

A multimodal language model generates local plan annotations from the task instruction and sampled demonstration frames, optionally supported by gripper-state and motion summaries. These are generated supervision labels rather than native step-level chain-of-thought annotations. A plan identifies the operation, manipulated object, and destination when applicable. It describes the next local operation visible in the segment. Content unsupported by the demonstration is excluded from the textual loss, and RoboTwin annotations use the seen training descriptions.

For both reasoning stages, a learned projection converts the textual latent into the continuous prefix consumed by the frozen text decoder. Teacher forcing supplies the preceding target tokens during training. Token NLL is averaged over the valid plan tokens, as in Eq. 10. Using the same annotation at Omni and Mono makes their objectives directly comparable while allowing the latents to encode that target at different levels of refinement.

Annotation template. The following template specifies the content of a local plan:

> Given the task instruction and ordered demonstration frames, describe the next local manipulation operation in one short sentence. Name the object and destination when they are identifiable. Ground the action in the visible segment. Do not assert successful completion beyond what the frames establish. If the next operation is ambiguous, mark the example for review instead of inventing an object, destination, or action.

A useful content structure is operation–object–destination, such as transferring an already grasped object toward its target. The wording can remain natural language; the three fields specify what information the annotation should convey. Segment-level labels must retain their episode and time indices when assigned to individual training examples.

Annotation settings. We use an offline Qwen3-VL-4B-Instruct checkpoint before policy fine-tuning as the annotation model, with the instruction template above, greedy decoding, and at most 48 new tokens. Use eight-step windows starting every four recorded steps, with frames at offsets $0, 2, 4, 6, 8$ when valid. Supply the task instruction and the available gripper/motion summary with the ordered frames. Assign a segment label only to its covered steps for which the named local operation remains valid; split a window at an operation transition. Drop a window with fewer than three distinct valid frames or an ambiguous operation. Require operation, object, and applicable destination to agree with the visible segment, manually inspect a task-balanced 10% sample, and review all flagged labels. Annotate the trajectory partitions separately and never use diagnostic test labels for policy or probe fitting.

B.3 Future Visual Features

The future target $o_{t+\Delta}$ comes from the same episode and camera as the current observation. Frozen DINOv2 encodes its complete patch grid. At resolution 224×224 with patch size 14, there are $P = 16 \times 16 = 256$ spatial locations. A spatial query is associated with each location, and the visual decoder reads the complete visual latent sequence to predict the corresponding teacher feature. Consequently, changing the number of visual latent slots changes representation capacity while keeping the supervision grid fixed.

The MSE in Eq. 11 averages over all valid grid locations and feature channels. Omni and Mono predict the same future feature target. The image itself is used to construct supervision and is not supplied as an inference-time observation. The target is a teacher feature grid, so feature prediction does not require reconstructing RGB pixels or generating an image during policy execution.

The default offset is eight recorded steps. For LIBERO at 20 Hz and Bridge at 5 Hz, this corresponds to 0.4 s and 1.6 s, respectively. These physical durations follow from the recording rates; matching a step offset across sources does not match its duration in seconds. RoboTwin also uses an offset of eight recorded steps.

B.4 Sparse Action References and Validity Masks

Sparse reference targets retain the single-step meaning of the recorded actions. With $H_{\text{ref}} = 8$ and $s = 2$, the reference contains the actions at $t, t + 2, \ldots, t + 14$; intervening commands are not accumulated. For end-effector increments, summing intermediate actions would change the target semantics. For RoboTwin joint-position commands, a sampled entry remains a joint-position target for that recorded step. Normalization uses statistics fitted to the corresponding training source.

A reference-action predictor is conditioned on $\mathbf{G}_A$ or $\mathbf{Z}_A$ and uses the flow-matching objective to predict the same sparse reference. The final action head instead conditions on the context and all refined representations to predict a contiguous executable chunk. The reference head's outputs therefore provide auxiliary motion supervision rather than replacing the final control sequence.

Targets that extend beyond an episode boundary are masked. If an aligned trajectory has length L_e, the visual validity indicator is $v_t^V = \mathbf{1}[t + \Delta < L_e]$, and reference position j has validity $v_{t,j}^A = \mathbf{1}[t + js < L_e]$. These indicators prevent target extraction from crossing into another episode. A masked action loss can be written as

$$\mathcal{L}_{\text{masked}} = \frac{\sum_j v_{t,j}^A \|\widehat{\mathbf{u}}_j - \mathbf{u}_j^\star\|_2^2}{d_a \sum_j v_{t,j}^A}, \tag{18}$$

where $\mathbf{u}_j^\star$ and $\widehat{\mathbf{u}}_j$ denote the target and predicted flow velocities. Samples without a valid target for a modality do not contribute to that modality's loss. Text tokens use their annotation validity and padding masks. The validity mask also applies when aggregating losses over a minibatch, so missing future supervision does not become an artificial zero-valued target. For the contiguous final action chunk, use the corresponding mask $v_{t,j}^{\text{act}} = \mathbf{1}[t + j < L_e]$ for $j = 0, \ldots, H - 1$ and normalize by the number of valid action coordinates. On complete chunks this reduces to the $d_a H$ normalization in Eq. 9.

C Evaluation Protocols

C.1 Success Rates and Aggregation

For task q, training seed k, and evaluation episode i, let $b_{q,k,i} \in \{0, 1\}$ denote success according to the task evaluator. Given $n_{q,k}$ trials, the task success rate is

$$\text{SR}_{q,k} = \frac{100}{n_{q,k}} \sum_{i=1}^{n_{q,k}} b_{q,k,i}. \tag{19}$$

Task means and instance-weighted means answer different questions. LIBERO reports the mean of four suite scores, SimplerEnv reports a four-task mean, and RoboTwin averages equally over its 50 tasks. LIBERO-Plus uses an instance-weighted aggregate over 10,030 instances. Internal results average the corresponding aggregate across training seeds.

For internal comparisons, the initial-state list and success criterion are shared across variants within each benchmark. Per-task trial counts, training seeds, and unrounded success counts determine the aggregate; rounding is applied for presentation. A category minimum identifies the weakest reported category, while the mean measures average performance.

C.2 LIBERO and LIBERO-Plus

The LIBERO policy is jointly trained on the Spatial, Object, Goal, and Long suites (Liu et al., 2023). Evaluation uses 50 initial states for each of 40 tasks, giving 2,000 episodes per training seed. Three seeds give 6,000 episodes per model. Paired initial states are used for internal comparisons. The suite average in Table 1 gives equal weight to the four suites.

LIBERO-Plus (Fei et al., 2026) evaluates the LIBERO-trained policy without additional fine-tuning. Each expanded instance receives one trial per evaluated checkpoint. The category counts are 1,599 Camera, 1,550 Robot, 1,537 Language, 1,142 Light, 1,076 Background, 1,601 Noise, and 1,525 Layout instances. If N_c is the category count and s_c its success rate, the aggregate is

$$\text{SR}_{\text{Plus}} = \frac{\sum_{c=1}^{7} N_c s_c}{\sum_{c=1}^{7} N_c}, \qquad \sum_{c=1}^{7} N_c = 10{,}030. \tag{20}$$

The category mean and the instance-weighted mean generally differ because the seven categories contain different numbers of instances. Reported minimum scores should be read together with the category-level results.

C.3 SimplerEnv

SimplerEnv evaluation uses the WidowX visual-matching setting (Li et al., 2024), with policies trained on BridgeData V2 (Walke et al., 2023). The four tasks are putting a spoon on a towel,

putting a carrot on a plate, stacking blocks, and putting an eggplant in a basket. Table 14 lists the available task-level results. Internal variants use the same observations, action conversion, rollout controller, and initial configurations. Four actions are executed before the next policy observation.

The aggregate gives equal weight to the four task success rates. The main comparison reports 78.3% for Omni–Mono, and the stage and component ablations use the values in Table 2.

Rollout budget. We use 100 initial configurations per task and checkpoint, with one rollout per configuration and training seeds $\{0, 1, 2\}$. Reuse the same initial configurations across internal variants. This gives 400 episodes per checkpoint and 1,200 per variant, with task rates averaged across seeds before taking the four-task mean. A pooled task score has resolution $100/300 \approx 0.333$ percentage points; the aggregate has resolution $100/1200 \approx 0.083$ points.

C.4 RoboTwin 2.0

RoboTwin uses the Aloha-AgileX embodiment and 50 tasks from RoboTwin 2.0 (Chen et al., 2025). The policy is trained jointly on 50 clean demonstrations per task. Clean-to-clean (C2C) and clean-to-random (C2R) evaluate the same clean-trained policy under different test conditions. The protocol uses 100 episodes per task in each condition, resulting in 5,000 C2C and 5,000 C2R episodes per training seed (RoboTwin Team, 2026). Success is determined by the task evaluator and averaged equally across tasks.

The selected configurations are from release commit `6dde57155eaf`. The clean condition disables the listed domain randomizations and uses seen instructions. The randomized condition changes background, clutter, lighting, and table height and uses unseen instructions. Head-camera displacement is zero in both configurations. Table 6 summarizes these settings. C2R therefore measures the combined shift across these factors; it does not isolate a single source of variation.

Table 6: RoboTwin evaluation conditions for the same clean-trained policy.

Setting	C2C	C2R
Configuration	`demo_clean`	`demo_randomized`
Evaluation tasks	50	50
Episodes per task and seed	100	100
Instruction variants	Seen	Unseen
Background / clutter / lighting	Disabled	Enabled
Table-height randomization	Disabled	Enabled
Head-camera displacement	0	0
Aggregate	Mean task SR	Mean task SR

C.5 Real-World Evaluation

Real-world experiments use a Franka-r3 arm with a parallel-jaw gripper. The two tabletop tasks are placing a milk box onto a plate and placing a bowl onto a plate. Omni–Mono, π_0 (Black et al., 2024), and $\pi_{0.5}$ (Physical Intelligence et al., 2025) use the same demonstration set, observation and action interfaces, and initial configurations. Each method receives 50 trials per task.

A successful trial requires grasping and releasing the instructed object so that it rests stably on the plate without human intervention. Reaching the target while continuing to hold the object does not satisfy this placement criterion. The task average weights the two tasks equally. Because each task has 50 trials, one successful trial changes a task score by two percentage points. The results in Figure 6 are tabulated in Appendix F.5.

Real-world configuration. For each evaluated task, we collect 50 demonstrations and reserve five for development. After selecting the training budget, we include these held-out demonstrations in the final training set. Use one fixed frontal RGB camera approximately 0.8 m from the workspace center and 0.6 m above the tabletop, tilted toward the manipulation area. Calibrate intrinsics with a checkerboard and extrinsics against the robot base frame. Resize the image to 224×224; calibration and measured robot state are used by the action adapter, not as extra policy inputs. Use

seven-dimensional Cartesian translation/rotation increments and a gripper command in the robot base frame. Predict eight actions, execute four per observation, and issue Cartesian targets at 5 Hz with the low-level controller maintaining the latest target during policy computation. Control frequency does not assert a 200 ms policy latency.

For Omni–Mono, use 2,000 cold-start and 20,000 joint updates with the optimizer settings above. For π_0 and $\pi_{0.5}$, use 20,000 fine-tuning updates, global batch size 64, and a peak learning rate of 10^{-5}, retaining their native training objectives. Adapt all methods to the same camera, action representation, and executed horizon. Use one development-selected configuration per method for the reported 50 trials; these trials are not three training-seed evaluations. Sample object and target positions independently within ± 5 cm of their nominal tabletop positions and object yaw within $\pm 15^\circ$, rejecting overlapping or unreachable placements. Share the 50 accepted initial configurations per task across methods. End a trial after 30 s or successful placement, and require the released object to remain stable for 2 s.

D Controlled Variants and Ablation Definitions

D.1 Reasoning Stages and Parameter Control

The stage comparison changes the reasoning representations provided to the action head. Base is the direct-action reference used in the timing comparison: its action head conditions on $\mathbf{C}_t$. Omni-only retains the joint reasoner and conditions the action head on $\mathbf{C}_t, \mathbf{G}_T, \mathbf{G}_V, \mathbf{G}_A$. Mono-only retains the sequence of modality experts while removing Omni guidance. The complete model supplies $\mathbf{C}_t, \mathbf{Z}_T, \mathbf{Z}_V, \mathbf{Z}_A$ after both stages. The prediction horizon and robot interface are unchanged within a dataset.

In Mono-only, the textual expert reads the context without an Omni source. The visual expert can use the preceding textual output, and the action expert can use both preceding expert outputs. In terms of guidance sources, this gives $\mathcal{R}_T = \varnothing$, $\mathcal{R}_V = \{\mathbf{Z}_T\}$, and $\mathcal{R}_A = \{\mathbf{Z}_T, \mathbf{Z}_V\}$. An empty guidance set bypasses the guidance branch; it does not invoke a softmax over an empty set. This preserves the sequential expert structure while removing the joint multimodal stage.

The representation study additionally includes a larger Omni-only control, Omni (MAP), with matched added attention-module parameters. The configuration below keeps the added-parameter difference within 5%. The matching excludes the pretrained backbones, which are fully fine-tuned. Accordingly, "matched added parameters" in Table 3 refers to the added attention budget. Different backbone counts remain part of the stage comparison.

Parameter-matched configuration. Omni (MAP) uses width 768 and 12 heads, with 15 interaction insertions before backbone blocks 3, 5, 8, 10, 12, 15, 17, 20, 22, 24, 27, 29, 32, 34, and 36. Each insertion contains one context-attention unit and one relation-masked unit, with one down/up projection pair shared by its branches. The full model retains its four Omni and two-per-expert Mono insertions. For width $d = 768$ and backbone width $b = 2560$, an attention unit contributes approximately $4d^2$ weights and a down/up pair $2bd$. Thus, the full model uses 38 attention units and ten projection pairs: eight units in Omni and $2[(3+1)+(4+1)+(5+1)] = 30$ in Mono. Including six guidance scorers, this gives about 133 million added parameters; the 15-insertion MAP control has about 130 million, a difference of about 2%.

Table 7 gives the corresponding parameter accounting. Let C be shared context-encoder parameters, C_{tr} its trainable subset, L one independent language backbone, F the final action head, D the trainable auxiliary heads for one reasoning stage, and T the frozen text decoder plus visual teacher. All language backbones are trainable during joint fine-tuning. P denotes the added interaction parameters, including projections and guidance scorers. Separate copies count separately; shared frozen teachers count once.

The Mono-only configuration bypasses the empty textual guidance branch and retains context attention at all six insertions. Dense attention has the same parameter count as the full model. Static guidance replaces about 3.5 million scorer weights with per-layer source logits. These controls isolate relation masks and input-dependent source weighting, while the stage comparison also changes the number of pretrained backbones.

Table 7: Parameter accounting for the evaluated configurations. Totals include training-only modules; P values are in millions.

Variant	P (M)	Total parameters	Trainable parameters
Base	0	$C + F$	$C_{\mathrm{tr}} + F$
Omni-only	35	$C + L + P + F + D + T$	$C_{\mathrm{tr}} + L + P + F + D$
Mono-only	54	$C + 3L + P + F + D + T$	$C_{\mathrm{tr}} + 3L + P + F + D$
Omni (MAP)	130	$C + L + P + F + D + T$	$C_{\mathrm{tr}} + L + P + F + D$
Omni–Mono	133	$C + 4L + P + F + 2D + T$	$C_{\mathrm{tr}} + 4L + P + F + 2D$

D.2 Component Ablations

Dense attention. This variant retains the number of heads, query groups, and attention projections while removing the modality relation masks. Every head can then retrieve from all three modality groups. The comparison tests the relation-specific allocation of heads under the same added-attention shape. Dense attention still permits cross-modal exchange, so its result compares two interaction structures rather than interaction versus its absence.

Static guidance weights. This variant retains source-wise retrieval and normalization but makes the source weights independent of the current query and observation. A static mixture can be expressed as

$$\gamma_{m,r}^{(l)} = \frac{\exp(\alpha_{m,r}^{(l)})}{\sum_{r' \in \mathcal{R}_m} \exp(\alpha_{m,r'}^{(l)})}, \tag{21}$$

where the learned logits are specific to an expert and guidance layer. The retrieved messages remain input-dependent, but the coefficients used to combine them are shared across queries and observations. This isolates the contribution of adapting the mixture to the current reasoning state.

The four ablations reported in Table 2 are Omni-only, Mono-only, dense attention, and static guidance weights. Each is retrained using the same data and optimization budget as its full-model comparison. The retained modalities keep their supervision targets; removing a complete reasoning stage also removes that stage's auxiliary losses. Dense and static variants retain supervision at both stages. Reported differences are percentage-point differences from the full model, not relative percentage changes.

Across LIBERO-Plus, SimplerEnv, and RoboTwin C2R, the Omni-only losses are 5.0, 4.5, and 5.8 points, and the Mono-only losses are 4.4, 3.9, and 5.1 points. These stage ablations support combining joint multimodal reasoning with modality-specific refinement. With both stages retained, dense attention loses 1.6, 1.3, and 1.6 points, while static guidance loses 2.5, 2.2, and 2.6 points under the same optimization budget.

D.3 Capacity-Controlled Refinement

The stage ablations change both the reasoning structure and the number of language backbones. We complement them with a comparison that retains four independent backbones and the six auxiliary objectives in Eq. 15.

Generic refinement. The control retains Omni and replaces the three modality-specific Mono experts with three sequential generic refiners. The refiners process $(2, 4, 2)$, $(4, 8, 4)$, and $(2, 4, 2)$ textual, visual, and action queries, respectively. Their total query lengths remain 8, 16, and 8, matching the original experts. Each query retains its modality and slot identity and is assigned to one refiner through a fixed partition. Queries within a refiner can exchange information across modalities. The first refiner reads $\mathbf{G}_T, \mathbf{G}_V, \mathbf{G}_A$; the second additionally reads the first refiner's eight-slot output; the third reads both preceding outputs. This preserves the original three, four, and five guidance branches and their source lengths. After refinement, outputs are regrouped into $\mathbf{Z}_T, \mathbf{Z}_V, \mathbf{Z}_A$ in their original slot order, giving the same 8/16/8 final query budget. The regrouped representations receive the existing Mono supervision and condition the unchanged action head.

This comparison tests modality-specific allocation and its associated sequential routing against mixed-modality refinement at the same backbone budget. Both configurations use the training data, optimizer, update counts, and cold-start schedule in Appendix A.2, with training seeds $\{0, 1, 2\}$. The resource-matching criteria are differences below 1% in total and trainable parameters and below 5% in total training FLOPs, including cold-start and auxiliary prediction. Identical attention tensor shapes and dense kernels keep mask-dependent kernel changes out of the compute comparison.

Evaluation. LIBERO checkpoints are evaluated on LIBERO and zero-shot LIBERO-Plus using Appendix C.2. The same comparison extends to Bridge-trained policies in SimplerEnv and clean-trained policies in RoboTwin C2R. Table 9 reports the comparison with generic refinement alongside the directional controls below. Omni–Mono exceeds generic refinement by 1.7, 1.4, and 1.8 percentage points on LIBERO-Plus, SimplerEnv, and RoboTwin C2R, respectively. These results support modality-specific allocation and sequential routing beyond the benefit of additional backbones alone.

D.4 Directional Information-Flow Ablations

The variants in Table 8 retain all four backbones, both stages of auxiliary supervision, and the final action head. They vary the communication available to reasoning latents. The shared context $\mathbf{C}_t$ remains available throughout, and the action head continues to read $\mathbf{C}_t, \mathbf{Z}_T, \mathbf{Z}_V, \mathbf{Z}_A$.

Table 8: Controls with four independent language backbones and six auxiliary losses. Arrows denote source-to-query communication. All variants retain shared context and final multimodal action fusion.

Variant	Refinement	Omni communication	Refinement guidance
Omni–Mono	Modality-specific	All directed relations	Full source sets
Generic refinement	Mixed-modality	All directed relations	Full source sets
Unidirectional	Modality-specific	$T \to V$, $T \to A$, $V \to A$	Forward and own-modality
Intra-modal	Modality-specific	Own-modality only	Own-modality only
Omni-intra	Modality-specific	Own-modality only	Full source sets

Unidirectional and intra-modal reasoning. The unidirectional variant permits the cross-modal relations $T \to V$, $T \to A$, and $V \to A$, together with all within-modality relations. Its textual expert reads only $\mathbf{G}_T$; its visual expert reads $\mathbf{G}_T, \mathbf{G}_V, \mathbf{Z}_T$; and its action expert retains $\mathbf{G}_T, \mathbf{G}_V, \mathbf{G}_A, \mathbf{Z}_T, \mathbf{Z}_V$. The intra-modal variant restricts every reasoning query to latents of its own modality in both Omni and Mono. Its textual, visual, and action experts therefore receive $\mathbf{G}_T$, $\mathbf{G}_V$, and $\mathbf{G}_A$, respectively. The Omni-intra control restricts Omni alone and retains the full Mono guidance sets, testing whether later interaction can compensate for the missing early exchange.

All attention heads and guidance branches remain active. An Omni head assigned to a prohibited source is redirected to its query's own modality. A prohibited Mono source is replaced by the corresponding $\mathbf{G}_m$, using fixed cyclic or evenly spaced slot indices to preserve the original source length. Each branch retains its projections, normalization, source embedding, and scorer. These replacements preserve trainable capacity while changing the information available to it. The same access rules apply to intervening backbone attention, and context tokens cannot read latents. Applying the restrictions to both stages prevents an omitted direct edge from being bypassed by a reverse path through another modality. Each variant is retrained with its specified routing and the common training protocol.

Effect of communication direction. Table 9 shows that unidirectional reasoning reduces SR by 2.3, 1.9, and 2.7 percentage points on LIBERO-Plus, SimplerEnv, and RoboTwin C2R, respectively. Restricting both stages to intra-modal communication increases these losses to 3.5, 3.0, and 4.2 points. Restricting Omni alone gives smaller losses of 1.5, 1.3, and 1.5 points. The stronger Omni-intra results relative to the intra-modal variant indicate that later cross-modal interaction recovers part of the benefit lost when early exchange is removed. The full model performs best across the three benchmarks, supporting cross-modal exchange in both reasoning stages at the same backbone capacity.

Table 9: Success rates (%) for comparisons retaining all four language backbones and both stages of auxiliary supervision.

Method	LIBERO-Plus Avg. ↑	SimplerEnv Avg. ↑	RoboTwin C2R ↑
Omni–Mono	**92.4**	**78.3**	**77.2**
Generic refinement	90.7	76.9	75.4
Unidirectional	90.1	76.4	74.5
Intra-modal	88.9	75.3	73.0
Omni-intra	90.9	77.0	75.7

D.5 Controlled Obstacle and Path Interventions

Paired scenes. We construct a diagnostic set of ten LIBERO manipulation tasks with twenty held-out initial states per task. Each state has two versions: an obstacle outside the nominal transport path and the same obstacle blocking that path while leaving a feasible detour or lifting motion. The instruction, manipulated object, destination, camera, and robot state are shared within each pair. Geometric collision and path checks determine scene eligibility before policy evaluation. Five additional pairs per task are reserved for protocol development. The 200 test pairs give 400 episodes per checkpoint and 1,200 across three training seeds for each variant. This diagnostic is evaluated separately from the official LIBERO-Plus aggregate.

Each rollout begins from the same predefined transport decision state. The environment is paused during policy computation for every variant, separating information-flow effects from inference latency. Besides task SR, we record whether the robot or held object makes unintended contact with the added obstacle. Intended grasp contacts are excluded from this collision measure.

Plan validity. At the decision state, the existing textual prefix projection and frozen decoder read only $\mathbf{Z}_T$, using greedy decoding with at most 48 tokens. Decoded text is used for diagnosis and is never fed back to the policy. Two annotators, blind to model identity, judge the operation, object, destination, and applicable path constraint, accepting multiple feasible motions. Plan validity rate (PVR) counts plans consistent with both the task and the current obstacle geometry. A plan that omits a required path constraint is counted as invalid. This distinguishes PVR from the operation–object–destination accuracy in Table 3.

Table 10: Results on the blocked-path diagnostic. All values are percentages. PVR denotes plan validity rate.

Method	SR ↑	Collision ↓	PVR ↑
Omni–Mono	**78.0**	**18.0**	**74.0**
Generic refinement	74.0	22.0	70.0
Unidirectional	71.0	25.0	65.0
Intra-modal	66.0	31.0	58.0

Table 10 shows that Omni–Mono achieves the highest task SR and PVR and the lowest obstacle collision rate. Its SR exceeds generic refinement, unidirectional reasoning, and intra-modal reasoning by 4.0, 7.0, and 12.0 percentage points, respectively. These gaps are larger than those on the standard benchmarks, where transport adjustments are not required in every scene. Collision and success can occur in the same episode, so their rates are not complements.

Visual-to-text intervention. To examine the pathway $\mathbf{G}_V \to \mathbf{Z}_T \to \widehat{\mathbf{A}}_t$, we cache a natural forward pass in each blocked scene and obtain $\mathbf{G}_V^{\mathrm{clear}}$ from its paired clear-path scene. We use $\mathbf{G}_V^{\mathrm{clear}}$ to compute the visual keys and values at both guidance insertions of the Mono textual expert, holding $\mathbf{C}_t, \mathbf{G}_T, \mathbf{G}_A$ and action noise fixed. We recompute only the textual representation, denoted $\mathbf{Z}_T^{\mathrm{swap}}$, and condition the action head on $(\mathbf{C}_t, \mathbf{Z}_T^{\mathrm{swap}}, \mathbf{Z}_V^{\mathrm{natural}}, \mathbf{Z}_A^{\mathrm{natural}})$. This restricts changes in the predicted action to the altered textual latent, since the other action-conditioning inputs are unchanged.

We execute the resulting first action chunk from the same blocked state, then resume the natural policy for the remainder of the episode. We compare plan validity, obstacle collision, and task success with the natural blocked-scene rollouts.

Table 11: Changes from a visual-to-text swap relative to the natural blocked-scene rollout. All changes are in percentage points.

Metric	Change
Plan validity rate	-7.0
Closed-loop SR	-4.0
Obstacle collision rate	$+3.0$

Table 11 shows that replacing the blocked-scene visual guidance with clear-path guidance lowers PVR by 7.0 points and closed-loop SR by 4.0 points, while increasing obstacle collisions by 3.0 points. These changes in plan validity and task success are consistent with a control-relevant role for the visual-to-text pathway. The swap combines states from different scenes and can be out of distribution, so we interpret it together with the retrained directional controls. The unchanged $\mathbf{C}_t$, $\mathbf{G}_T$, and $\mathbf{G}_A$ may also carry obstacle information, providing other routes for the policy to use it. PVR assesses the constraints expressed by the decoder and do not capture unexpressed information in the latent.

Result aggregation. Success, collision, and plan-validity rates are computed from episode records and averaged across the three training seeds. Comparisons use the same tasks, initial states, and rollout seeds. Table 10 reports blocked-scene rates, and Table 11 reports percentage-point changes relative to the natural blocked-scene rollouts.

E Representation, Hyperparameter, and Timing Protocols

E.1 Representation Diagnostics

The representation study uses a separate LIBERO split of 40 training, five validation, and five test trajectories per task. Diagnostic policies are retrained on the 40-trajectory training split. Policy optimization and probe fitting exclude the test trajectories. This differs from the 50-demonstration policies used for the main benchmark comparison. Probes are matched within each modality in architecture, optimization budget, and stopping rule, and read only the corresponding frozen latent. Thus, a text probe does not additionally receive the visual latent or shared context when comparing textual representations.

Text metrics. Token NLL averages the log loss over generated plan-target tokens. Semantic plan accuracy evaluates 200 task-balanced test observations with separately annotated operation, object, and destination labels. Model identity is hidden during scoring. These semantic labels are manual annotations for the diagnostic rather than native LIBERO chain-of-thought labels. The metric tests recovery of the local plan semantics encoded by the frozen textual representation.

Visual and action metrics. Future-feature MSE averages prediction error over the full teacher patch grid and feature channels. Action-reference RMSE uses the same sampled, training-normalized action targets across variants. The target stride, reference length, and prediction budget are matched. Both metrics measure agreement with demonstration-derived targets; success rates evaluate closed-loop manipulation outcomes.

The three error metrics in Table 3 are normalized by the corresponding Omni-only error. For metric E_m, the reported value is

$$\widetilde{E}_m(\mathbf{U}) = \frac{E_m(\mathbf{U})}{E_m(\mathbf{G}^{\text{Omni-only}})}. \tag{22}$$

Each modality has its own denominator. Consequently, a lower relative visual MSE cannot be compared numerically with text NLL as if they shared units. Within Omni–Mono, refinement reduces NLL from 0.96 to 0.81, MSE from 0.96 to 0.78, and RMSE from 0.94 to 0.83. These correspond

to relative reductions of approximately 15.6%, 18.8%, and 11.7%, calculated from the displayed values. Semantic accuracy rises from 80.0% to 87.0%, a 7.0-point gain. The 6.0-point gain reported against the matched-attention Omni control uses a different reference, 81.0%, and is consistent with that comparison.

Probe-fitting details. For every representation, we fit a fresh text-prefix projection, four-layer visual decoder, and six-layer reference-action head with the respective architectures in Appendix A.1. Use identical initializations across compared representations for each seed, frozen policy latents, and the same frozen text decoder and visual teacher. Feed the entire corresponding latent sequence to each probe; do not pool slots for these prediction metrics. Train for at most 10,000 updates with batch size 64, AdamW at 10^{-4}, 500 warm-up updates, cosine decay, and weight decay 0.01. Select each probe checkpoint by its modality-specific validation loss every 500 updates. Pair policy and probe seeds in $\{0, 1, 2\}$ and average each absolute metric across the three pairs before forming its relative ratio. For reference RMSE, generate eight sparse actions with ten Euler steps and matched Gaussian draws across variants.

For semantic accuracy, select five test observations per task, retain the same 200 observations across variants, and decode greedily with at most 48 tokens. Count an observation as correct only when its operation, object, and applicable destination all match the reference; accept a fixed synonym list decided before inspecting model outputs. Use an explicit "not applicable" destination for operations without a target and require the decoded plan not to invent one. Two annotators score independently with model identity hidden and adjudicate disagreements.

The Omni-only denominators are 0.82 nats per token, 0.16 teacher-feature MSE, and 0.18 normalized-action RMSE. The corresponding Omni–Mono $\mathbf{G}/\mathbf{Z}$ errors are $0.7872/0.6642$, $0.1536/0.1248$, and $0.1692/0.1494$, respectively.

E.2 Latent-Structure Visualization

Figure 3 displays two-dimensional views of the Omni and Mono representations. The textual groups are grasp, transfer, and place operations. Visual groups describe far, near, and at-target object–target relations. Action groups describe approach, close, transport, and release control states. Axes are in arbitrary units, and the dashed contours denote one standard deviation.

These plots complement the quantitative probes by exposing the local organization of task-related states. The textual groups become more compact after refinement; visual states retain an ordered progression with neighboring overlap; action states remain distinguishable with residual overlap. Distances across different modalities have no common semantic scale. The shared projection below supports compactness comparisons between $\mathbf{G}$ and $\mathbf{Z}$ within each modality.

Projection and annotation procedure. We use the seed-0 diagnostic checkpoint and sample 25 valid observations per task from its held-out test trajectories, giving 1,000 paired $\mathbf{G}/\mathbf{Z}$ observations, with sampling seed 42. Mean-pool the slots within each modality to one vector per observation. For each modality separately, center the concatenated $\mathbf{G}$ and $\mathbf{Z}$ vectors using one shared mean, fit a single two-component PCA with deterministic SVD and no whitening, and apply that same projection and axis limits to both stages. Do not standardize the two stages separately.

Annotate grasp, transfer, and place from the demonstrated operation. For object–target relations, use offline task geometry: call a state at-target when the object center is within half an object diameter of the task's goal center, near up to two diameters, and far beyond that. These labels describe relative position, not successful release. Label action states using motion and gripper transitions: approach before contact, close during grasp closure, transport while carrying, and release at the target. Exclude ambiguous transition frames from colored groups. Draw each contour as the covariance ellipse with principal radii $\sqrt{\lambda_1}$ and $\sqrt{\lambda_2}$, where λ_i are the projected group covariance eigenvalues. This is a one-standard-deviation ellipse along each principal axis, not a confidence interval for the group mean. The geometric and gripper information is used only for offline labels and is not added to the policy observation.

E.3 Hyperparameter Sweeps

Figure 4 examines query capacity, auxiliary loss weight, and supervision horizons on LIBERO. Results are mean success rates across three training seeds. Hyperparameter selection uses the trajectory-level development splits in Appendix A.2. Each sweep changes the indicated setting while keeping the other settings at their defaults.

Query capacity. Both stages scale the default query budget $(8, 16, 8)$ by a common factor. The evaluated factors are 0.25, 0.5, 0.75, 1, 1.5, 2, 3, and 4. For example, the smallest budget uses $(2, 4, 2)$ slots and the largest uses $(32, 64, 32)$. The full visual teacher grid remains 16×16 throughout. The sweep therefore changes latent capacity rather than the number of spatial supervision targets. Query-budget timing follows Appendix E.4.

Auxiliary loss weight. The three auxiliary coefficients share a common value: $\lambda_T = \lambda_V = \lambda_A = \lambda$. The evaluated values are 0.01, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2, 0.3, and 0.5. The final-action weight remains one, as do the two stage weights. Thus, changing λ jointly rescales all six auxiliary terms in Eq. 15, preserving their relative coefficients. The selected setting is $\lambda = 0.1$.

Supervision horizons. The visual offsets are $\Delta = 2$, 4, 6, 8, 12, 16, and 24. Each is paired with reference strides $s = 1$, 2, 3, 4, 6, and 8. Reference length remains $H_{\mathrm{ref}} = 8$, and the prediction and execution horizons are unchanged. These sets define 42 combinations. A reference target spans $7s$ recorded intervals, so the sweep changes its temporal coverage from seven to 56 intervals. The visual offset changes independently. The selected setting is $(\Delta, s) = (8, 2)$.

Filled markers in Figure 4 identify the selected settings. Hyperparameter axes use logarithmic spacing, and the surface connects the evaluated combinations. Its interpolated appearance should be read as a visualization of the sampled configurations. The figure supports choosing the default among the evaluated settings rather than a global optimum over all possible configurations.

E.4 Inference Timing

Internal latency is measured on one NVIDIA A100 80GB GPU with BF16 precision, batch size one, and one 224×224 RGB observation. The timing boundary covers input processing, reasoning, and final action generation through an eight-action output chunk with ten flow-integration steps. Base and Omni–Mono use the same input and final action-head settings. The auxiliary supervision heads are excluded from inference.

Figure 5 reports a mean policy-call latency of 680 ms for Omni–Mono and 170 ms for Base. The added reasoning therefore increases mean latency by 510 ms, giving a fourfold policy-call latency in this internal comparison. Compared with Fast-ThinkAct at 805 ms, the reduction is $(805 - 680)/805 \approx 15.5\%$. The figure also includes LaRA-VLA at 135 ms, so the comparison does not establish that Omni–Mono has the lowest latency among all latent-reasoning policies.

Policy-call latency and robot control frequency are distinct quantities. The former measures the computation needed to produce a chunk; the latter also depends on how many actions are executed before replanning and on controller scheduling. The dataset-specific executed horizons are given in Table 5.

Timing implementation. We run 100 warm-up calls followed by 1,000 timed calls on cached observations and instructions, with a fixed checkpoint and no gradient tracking. Start a host monotonic timer after device synchronization and before image resize/normalization, tokenization, and host-to-device transfer. Include context encoding, all active reasoners, ten flow steps, inverse action normalization, and transfer of the eight-action chunk to the host. End after a second device synchronization. Exclude disk reads, sensor capture, robot communication, and physical action execution. Use the same cached inputs, precision, and boundaries for Base and Omni–Mono, and keep training-only networks unloaded. The arithmetic mean over calls is 170 ms for Base and 680 ms for Omni–Mono.

F Detailed Benchmark Results and Supplementary Analysis

F.1 Aggregate Gains Across Benchmarks

Table 12 derives the margin over the strongest reported baseline in each aggregate column of Table 1. The strongest baseline is selected separately for each column. All differences are computed from the displayed success rates.

Table 12: Aggregate success rates and margins derived from the main comparison. Gains are percentage points.

Benchmark	Strongest listed baseline	Baseline	Omni–Mono	Gain
LIBERO	DualCoT-VLA	98.8	**99.3**	+0.5
LIBERO-Plus	DualCoT-VLA	87.2	**92.4**	+5.2
SimplerEnv	DualCoT-VLA	73.6	**78.3**	+4.7
RoboTwin C2C	DualCoT-VLA	86.8	**90.6**	+3.8
RoboTwin C2R	DualCoT-VLA	71.2	**77.2**	+6.0

On LIBERO, Omni–Mono scores 99.6, 99.5, 99.4, and 98.8 on Spatial, Object, Goal, and Long, respectively. Their mean rounds to 99.3. The Long score exceeds the strongest listed baseline's 98.2 by 0.6 points. The larger aggregate margins on LIBERO-Plus and RoboTwin C2R provide additional evidence under changed evaluation conditions, alongside the near-saturated standard LIBERO result.

F.2 LIBERO-Plus Results by Perturbation Category

Table 13 reports category-level success rates. The aggregate uses Eq. 20, giving an average of 92.4 and a minimum of 79.6 for Omni–Mono. The reported aggregate margin over DualCoT-VLA is 5.2 points, while the minimum-score margin is $79.6 - 77.4 = 2.2$ points.

Table 13: LIBERO-Plus category success rates. Avg. and Min follow the main comparison.

Method	Camera	Robot	Lang.	Light	Backg.	Noise	Layout	Avg.	Min
OpenVLA-OFT	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6	31.9
OFT (1 view)	10.4	38.7	70.5	76.8	93.6	49.9	69.9	55.8	10.4
$\pi_{0.5}$	75.8	79.4	83.3	95.5	95.0	89.6	87.0	85.7	75.8
ACoT-VLA	72.6	82.6	87.5	97.7	96.5	87.8	88.1	86.6	72.6
Omni–Mono	79.6	94.1	95.4	98.2	98.8	91.5	93.2	**92.4**	**79.6**

Omni–Mono has its lowest success rate under Camera perturbations and higher rates under Light and Background perturbations. Noise and Layout fall between these categories. With category counts in Appendix C.2, the displayed values give 92.4068%, which rounds to 92.4%, and their minimum is exactly 79.6%. OpenVLA has main-table Avg./Min values of 15.6/3.3.

F.3 SimplerEnv Results by Task

Table 14 reports WidowX task-level success rates. Omni–Mono's aggregate of 78.3 exceeds the main-table DualCoT-VLA result of 73.6 by 4.7 points and the LaRA-VLA result of 68.8 by 9.5 points.

Table 14: SimplerEnv WidowX task success rates. Avg. is the four-task mean.

Method	Spoon on towel	Carrot on plate	Stack blocks	Eggplant in basket	Avg.
OpenVLA	8.3	4.2	0.0	45.8	14.6
ThinkAct	58.3	37.5	8.7	70.8	43.8
LaRA-VLA	95.8	62.5	25.0	91.7	68.8
Omni–Mono	97.3	76.3	44.3	95.3	**78.3**

Block stacking has the lowest success rate for Omni–Mono, while spoon and eggplant placement reach 97.3% and 95.3%, respectively. The four displayed task scores average to 78.3%. Each task uses 300 rollouts across three checkpoints.

F.4 RoboTwin Clean and Randomized Results

Table 15 expands the C2C/C2R comparison by reporting the absolute drop under randomized evaluation. All values are taken from Table 1; the drop is their arithmetic difference. Omni–Mono falls from 90.6 to 77.2, a 13.4-point drop. DualCoT-VLA falls from 86.8 to 71.2, a 15.6-point drop. The 2.2-point difference between these drops explains why Omni–Mono's margin grows from 3.8 in C2C to 6.0 in C2R. This comparison concerns the combined distribution shift in Appendix C.4.

Table 15: RoboTwin success rates and the clean-to-randomized drop, derived from the main comparison. Drop is C2C minus C2R in percentage points.

Method	C2C	C2R	Drop
OpenVLA	48.6	26.4	22.2
OpenVLA-OFT	79.8	57.6	22.2
OFT (joint LIBERO)	79.8	57.6	22.2
OFT (1 view)	75.4	51.8	23.6
$\pi_{0.5}$	84.8	68.4	16.4
CoT-VLA	63.2	40.8	22.4
ThinkAct	66.4	45.6	20.8
Fast-ThinkAct	74.6	55.8	18.8
ACoT-VLA	85.2	69.6	15.6
LaRA-VLA	83.6	67.4	16.2
DualCoT-VLA	86.8	71.2	15.6
Omni–Mono	**90.6**	**77.2**	**13.4**

The evaluation budget is 5,000 episodes per condition and checkpoint, or 15,000 per condition across three seeds. At the displayed success rates, this corresponds to approximately 4,530 C2C and 3,860 C2R successes per 5,000 trials, or 13,590 and 11,580 across three seeds.

F.5 Real-World Task Results

Table 16 transcribes the scores in Figure 6. The two task scores average to 86.0 for Omni–Mono, 75.0 for $\pi_{0.5}$, and 65.0 for π_0. The corresponding Omni–Mono gains over $\pi_{0.5}$ are 8.0 and 14.0 percentage points. The larger margin occurs for placing the bowl onto the plate, consistent with the rim-grasp and stable-release demands described in Sec. 3.7.

Table 16: Real-world task success rates transcribed from Figure 6. Each task uses 50 trials per method.

Method	Milk box on plate	Bowl on plate	Avg.
π_0	74.0	56.0	65.0
$\pi_{0.5}$	82.0	68.0	75.0
Omni–Mono	**90.0**	**82.0**	**86.0**

These scores correspond to 45 and 41 successful Omni–Mono trials, 41 and 34 successful $\pi_{0.5}$ trials, and 37 and 28 successful π_0 trials under the 50-trial protocol.

G Related Work

Vision-Language-Action Models. VLA models transfer pretrained vision-language knowledge to robotic control through demonstration learning. RT-2 (Zitkovich et al., 2023) represents actions as tokens, while OpenVLA (Kim et al., 2025b) provides an open-source policy adaptable to new manipulation settings. Subsequent work improves continuous control through flow-matching action generation in π_0 (Black et al., 2024) and parallel action chunking in OpenVLA-OFT (Kim et al., 2025a). $\pi_{0.5}$ (Physical Intelligence et al., 2025) further combines heterogeneous robot and web data with semantic supervision to improve generalization. Our work builds on these advances by enriching the intermediate reasoning representations that condition action generation.

Reasoning in Vision-Language-Action Models. Intermediate supervision connects task planning to physical execution: ECoT (Zawalski et al., 2025) grounds reasoning in scene and motion annotations, while CoT-VLA (Zhao et al., 2025) and ACoT-VLA (Zhong et al., 2026a) derive guidance from future images and reference actions, respectively. To reduce explicit decoding, Fast-ThinkAct (Huang et al., 2026) distills reasoning into latent thoughts, and LaRA-VLA (Bai et al., 2026) internalizes textual reasoning alongside future visual prediction. HALO (Shou et al., 2026) combines specialized experts through shared attention with causal dependencies across modalities. DualCoT-VLA (Zhong et al., 2026b) produces visual and linguistic reasoning in one backbone pass using parallel latent queries. Omni–Mono jointly models textual, visual, and action reasoning through interactions among all three streams, then uses dedicated experts to refine each modality with adaptive multimodal guidance.