VLA-Δ: LEARNING PHYSICAL STATE TRANSITION FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models transfer semantic knowledge acquired through vision-language pretraining to robot control, achieving strong performance across various benchmarks. However, most VLAs directly map observations to demonstrated actions without explicit supervision of intermediate physical state changes, making them prone to overfitting observation–action correspondences. Some methods introduce intermediate reasoning or future-state prediction in language, visual, or latent spaces to guide action generation, but do not explicitly supervise the physical state transition itself. We therefore propose VLA-Δ, a general framework that uses physical changes between current and future states as explicit learning targets. To support this objective, we construct a large-scale pretraining dataset and derive physical state-transition annotations covering task-relevant semantics and visual motion changes, providing more direct supervision for continuous action generation. During inference, the model generates actions directly from current observations without additional inference overhead. VLA-Δ achieves performance competitive with the state of the art on LIBERO and RoboTwin, and outperforms $\pi_{0.5}$ in real-robot experiments.

1 INTRODUCTION

Building generalist robot policies that follow natural-language instructions in diverse environments is a long-standing goal of embodied intelligence. Vision-language-action (VLA) models (Brohan et al., 2023; Kim et al., 2024; Bjorck et al., 2025; Black et al., 2024; Intelligence et al., 2025; Zheng et al., 2026; Community, 2026; Li et al., 2024a) have recently emerged as a promising paradigm toward this goal. Built on pretrained vision-language models (VLMs), VLAs inherit rich semantic knowledge and visual understanding from large-scale vision-language pretraining. They use the VLM backbone to encode task instructions and visual observations, mapping these representations to robot actions and achieving strong performance across diverse benchmarks.

Despite this progress, VLAs often exhibit failures revealing a limited understanding of physical dynamics. For example, a policy may close its gripper before reaching the target object, or continue toward the placement location after a failed grasp. In such cases, the policy appears to replay a memorized trajectory rather than respond to the evolving scene. These failures suggest that reliable manipulation requires reasoning about **how the robot and its environment should change** as the task progresses. A key reason is that most VLAs directly map observations to demonstrated actions and **lack explicit supervision of the intermediate physical state changes**. They tend to overfit observation–action correspondences instead of capturing how physical states change over time.

As illustrated in Fig. 1, existing studies (Li et al., 2026; Cen et al., 2025; Sun et al., 2026; Lin et al., 2026) introduce intermediate reasoning or future-state prediction to guide action generation by implicitly modeling state changes. (b) Reasoning VLAs such as ECoT (Zawalski et al., 2024) autoregressively infer task plans and object locations before generating actions. However, their chain-of-thought supervision focuses on high-level task planning and state reasoning, together with sparse visual localization. It describes what should be done next, but not how the physical states of the robot and objects change between current and future states. (c) World Action Models (WAMs) (Cai et al., 2026) model state changes by generating future images. These methods provide denser visual details, but future-image reconstruction mainly enforces consistency in scene appearance and does not explicitly identify which changes are necessary for task completion. Moreover, generating

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.

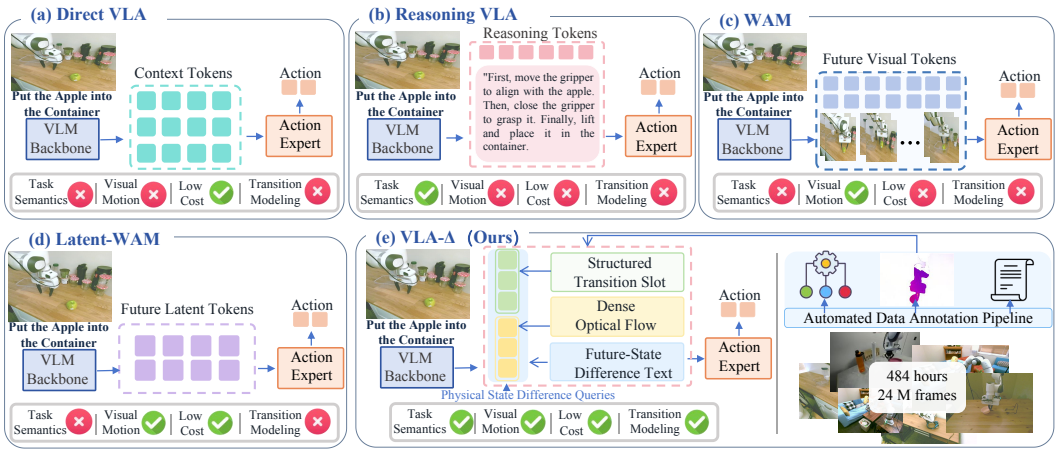


Figure 1: Comparison of transition modeling paradigms in vision–language–action policies. (a) Direct VLA maps observations and instructions directly to actions. (b) Reasoning VLA describes task plans and subgoals in language space. (c) WAM predicts future visual states, while (d) Latent-WAM predicts future latent representations. (e) VLA-Δ models physical state changes through multilevel supervision.

future images typically introduces additional inference cost. (d) Latent-WAMs (Chen et al., 2026) move future prediction into a latent space, reducing the cost of visual reconstruction. However, their latent representations still model only future visual states, rather than signals tied to physical states and motion interactions. Overall, these approaches supervise either high-level task semantics or future visual appearance, but rarely supervise the physical state transition itself, i.e., how object poses, spatial relations, and robot–object contacts change from the current state to the future. Yet it is precisely these physical changes that directly define which actions the robot must take.

To bridge this gap, we propose to **explicitly model physical state transitions as learning targets**. Such transitions should be both linked to task intent and grounded in concrete visual information: task semantics alone cannot specify how objects move, while visual appearance alone cannot indicate which changes matter for the task. Our core design is to extract physical state-transition information between the current and future states, encompassing both task-relevant semantics and visual motion changes. We aim to learn transition representations that capture task semantics while preserving motion details, providing more direct supervision for continuous action generation.

We therefore introduce VLA-Δ, a general framework using physical state transitions as explicit learning targets. The framework incorporates learnable Physical State Difference Queries into the vision-language backbone to form a unified transition representation conditioned on current observations and task instructions. We shape this representation through two supervision pathways: direct prediction and autoregressive generation. ① **Structured Transition Slots** provide structured supervision for variables including spatial positions and contact states, encouraging a semantic division of labor among Queries and giving the transition representation clear physical meaning. ② **Optical Flow** provides dense visual motion supervision, encouraging the representation to preserve motion details of objects and grippers. ③ **Future-State Difference Text** provides coherent semantic supervision for task progress, spatial displacement, contact changes, and associated physical constraints through autoregressive generation. Together, these three signals guide the Query representations to jointly encode task semantics and visual motion changes, supporting continuous action generation.

We construct a pretraining dataset comprising approximately 484 hours and 24 million frames and derive physical state-transition supervision signals from it. The three supervision signals are introduced only during pretraining, while downstream fine-tuning uses only the action-learning objective. This training scheme allows the physical state-transition knowledge acquired during pretraining to provide a representational foundation for downstream policy learning and adapt to the continuous control requirements of specific tasks. During deployment, the model directly generates actions without decoding intermediate supervision targets.

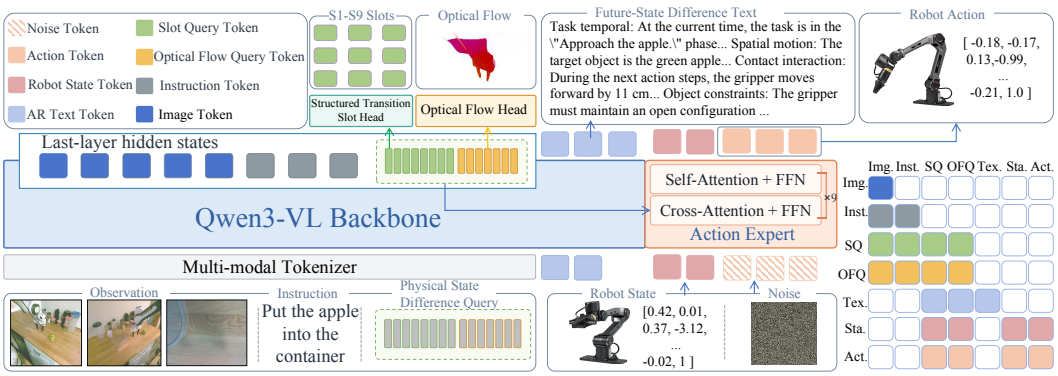


Figure 2: Overall architecture of VLA-Δ. Given multi-view observations, a task instruction, and the robot state, the model jointly learns a physical state transition representation from Structured Transition Slots, Optical Flow, and Future-State Difference Text supervision, and uses these representations to condition continuous action generation. The right panel illustrates the attention-visibility constraints among the different information streams.

VLA-Δ achieves strong performance in both simulation and real-robot experiments. VLA-Δ achieves a success rate of 98.40% on LIBERO and outperforms all compared methods under both the Clean and Random settings on RoboTwin. In real-robot experiments, it outperforms $\pi_{0.5}$ by 5.5%. These results suggest that physical state-transition supervision during pretraining helps the model focus on task-relevant motion changes, enabling it to benefit from pretrained transition representations during downstream fine-tuning.

Our main contributions are as follows:

1. We propose VLA-Δ, a general framework using physical state transitions as explicit learning targets. By explicitly modeling task semantics and visual motion changes, it provides more direct supervision for continuous action generation.
2. We curate approximately 484 hours and 24 million frames of robot demonstrations and develop a complete automated data annotation pipeline to generate large-scale, multilevel physical state-transition annotations for model pretraining.
3. VLA-Δ achieves the highest success rates among the compared methods on LIBERO and RoboTwin and outperforms $\pi_{0.5}$ in real-robot experiments.

2 Method

2.1 Overall Architecture of VLA-Δ

As illustrated in Fig. 2, VLA-Δ employs learnable Physical State Difference Queries to organize vision-language information into a physical state-transition representation for continuous action generation. The overall architecture follows the vision-language backbone and action-expert design of GR00T N1 (Bjorck et al., 2025), using Qwen3-VL-2B-Instruct and a 9-layer Diffusion Transformer.

At time t, the model receives multi-view visual observations o_t, a task instruction ℓ, and the robot proprioceptive state r_t, and predicts a continuous action chunk of length H_a, $A_t = (a_t, \ldots, a_{t+H_a-1})$. The visual observations and task instruction are first encoded into a contextual sequence C_t, which is then fed into the vision-language backbone F_θ together with the learnable Query parameters $Q \in \mathbb{R}^{N_q \times d}$. We extract the final-layer hidden states at the Query positions to obtain the transition representation Z_t:

$$Z_t = \mathrm{Readout}_Q(F_\theta([C_t; Q])) \in \mathbb{R}^{N_q \times d}. \tag{1}$$

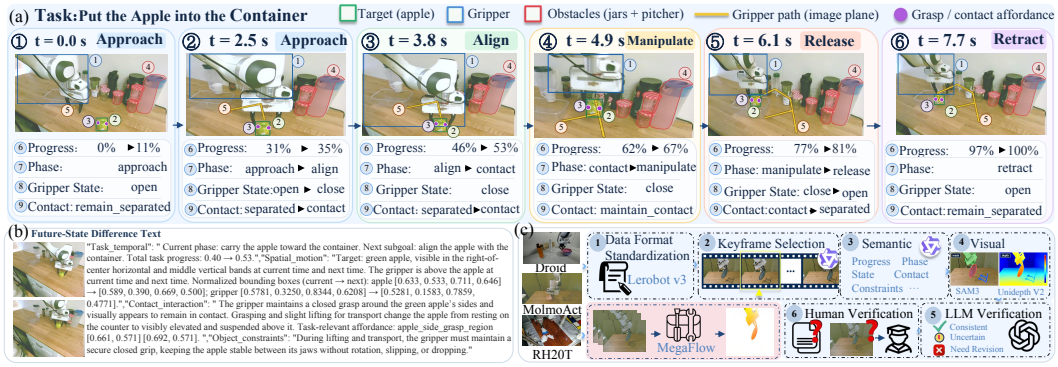


Figure 3: Physical state-transition annotations and the automated data annotation pipeline.

We shape this representation through two pathways: direct prediction and autoregressive generation. The direct prediction pathway reads the designated Query hidden states to predict structured variables, $\hat{y}_t = h_{\text{slot}}(Z_t^{\text{slot}})$, where $\hat{y}_t = (\hat{y}_{t,1}, \ldots, \hat{y}_{t,9})$, and dense optical flow, $\hat{U}_t = h_{\text{flow}}(Z_t^{\text{flow}})$.

The autoregressive generation pathway predicts Future-State Difference Text, $\hat{T}_t = \text{ARDecode}_\theta(Q; M_{\text{attn}})$. Under the attention constraints illustrated in the right panel of Fig. 2, text positions can attend only to the Queries and preceding text, and cannot directly attend to the context C_t. Therefore, information from the current observations and task instruction must pass through the Query representation before contributing to difference-text prediction.

The action expert attends to Z_t through cross-attention and combines it with the independently encoded robot proprioceptive state $E_\eta(r_t)$. It then transforms Gaussian noise into a continuous action chunk through flow matching: $\hat{A}_t = \text{FMDecode}_\psi(\xi; Z_t, E_\eta(r_t)), \xi \sim \mathcal{N}(0, I)$.

The three auxiliary supervision signals are used jointly with action learning only during pretraining, while downstream fine-tuning optimizes only the action-learning objective. During deployment, the model retains the Query representation extraction and action-generation pathways without decoding the auxiliary supervision targets.

2.2 Multilevel Physical State Transition Learning

Structured Transition Slots Supervision. Physical State Difference Queries lack explicit constraints on their semantic roles when trained solely through action prediction. To encourage transition representations to encode physically meaningful information, we introduce Structured Transition Slots, which organize task-relevant state variables and their changes into nine categories of supervision targets with predefined semantic roles: task progress, task phase, target position, gripper position, motion and relative displacement, interactable position, gripper state transitions, contact transitions, and obstacle risk. This supervision encourages the assigned Queries to develop a soft semantic division of labor while preserving information exchange among Queries, allowing them to jointly support action generation. Figure 3 presents an example of Structured Transition Slot annotations for a complete manipulation episode. Further details appear in Appendix B.2.

We assign a subset of Queries to these structured targets. For each slot category, a lightweight prediction head reads the corresponding Query hidden representations to output $\hat{y}_i$. The overall structured objective is $\mathcal{L}_{\text{slot}} = \sum_{i=1}^{9} w_i \mathcal{L}_i(\hat{y}_i, y_i)$, where w_i denotes the weight assigned to each target category. These structured targets provide direct supervision for learning key state variables.

Dense Optical Flow Supervision. Structured Transition Slots supervise predefined state variables and their changes, but do not fully capture fine-grained visual motion during object movement, gripper approach, and local alignment. To complement this information, we introduce Optical Flow supervision using dense two-dimensional displacements between current and subsequent observations from the same viewpoint, encouraging Query representations to preserve local motion details in the image plane. We use a lightweight optical flow decoder that reads only the final hidden states of the last N_{flow} difference queries and decodes their motion information into dense optical flow.

The total loss is defined as $\mathcal{L}_{\text{flow}} = \mathcal{L}_{\text{all}} + \beta_{\text{motion}}\mathcal{L}_{\text{motion}}$, where $\mathcal{L}_{\text{all}}$ is the average error over all valid pixels, and $\mathcal{L}_{\text{motion}}$ is the average error over motion pixels whose target flow magnitude exceeds a predefined threshold.

Optical flow alone does not distinguish task-relevant motion from other image motion. We therefore combine it with Structured Transition Slots and Future-State Difference Text, jointly constraining transition representations through state semantics and visual motion objectives.

Future-State Difference Text Supervision. Structured Transition Slots and Optical Flow supervise physical state changes through predefined state variables and dense visual displacements, respectively. To better support action generation, transition representations also need to capture the relationships among these changes, including how object motion and contact transitions jointly advance the current subgoal and which physical constraints must be satisfied throughout this process. We therefore introduce Future-State Difference Text, which uses coherent natural language to jointly describe task progress, spatial changes, contact transitions, and associated constraints between current and future states. Figure 3(b) presents an example of Future-State Difference Text.

We use the VLM's native autoregressive objective to learn Future-State Difference Text T. Cross-entropy is computed only over valid target text positions Ω_T: $\mathcal{L}_{\text{Text}} = -\frac{1}{|\Omega_T|}\sum_{j\in\Omega_T}\log p_\theta(t_j \mid C, Q, t_{<j}; M_{\text{attn}})$. Current vision-language information is routed exclusively through Queries for difference text prediction.

This supervision encourages Queries to capture the relationships between local state changes and overall task progression, providing transition representations with coherent task semantics for continuous action generation.

Action Generation Conditioned on the Transition Representation. The action expert combines the transition representation Z_t with the independently encoded proprioceptive state $E_\eta(r_t)$ to predict a continuous action chunk. We adopt the Flow Matching objective for continuous actions. Given a normalized demonstration action chunk A_t, we sample Gaussian noise ξ of the same shape and an interpolation time $\tau \in [0, 1]$, and construct $A_t^\tau = (1-\tau)\xi + \tau A_t$, $v_t^\star = A_t - \xi$, $\xi \sim \mathcal{N}(0, I)$.

The action expert v_ψ predicts the vector field from the noisy action, time, and conditional representation, optimized by $\mathcal{L}_{\text{act}} = \mathbb{E}\left[\frac{1}{|\Omega_a|}\sum_{k\in\Omega_a}\left([v_\psi(A_t^\tau, \tau \mid Z_t, E_\eta(r_t))]_k - [v_t^\star]_k\right)^2\right]$.

Training and Inference. During joint pretraining, we optimize physical state-transition prediction and continuous action generation using the following objective:

$$\mathcal{L} = \lambda_{\text{Text}}\mathcal{L}_{\text{Text}} + \lambda_{\text{slot}}\mathcal{L}_{\text{slot}} + \lambda_{\text{flow}}\mathcal{L}_{\text{flow}} + \lambda_{\text{act}}\mathcal{L}_{\text{act}}. \tag{2}$$

When fine-tuning on demonstrations from downstream benchmarks, we optimize only the flow-matching action objective, $\mathcal{L} = \mathcal{L}_{\text{act}}$. This allows the transition representations learned during pretraining to adapt to downstream tasks through action supervision, without requiring additional transition annotations for downstream demonstrations.

During inference, the VLM extracts the transition representation Z_t from the current visual observations and task instruction. The action expert then generates continuous action chunks via flow matching, conditioned on Z_t and the robot proprioceptive state.

2.3 Automated Data Annotation Pipeline

As shown in Figure 3(c), we develop an automated annotation pipeline to produce standardized physical state-transition annotations for robot demonstrations from multiple sources. First, we convert all data sources into the unified LeRobot v3 trajectory format. For keyframe selection, we use Qwen3.8 to segment demonstration videos into manipulation stages based on task instructions, identifying processes such as approaching, aligning, and grasping. Stage boundaries serve as keyframe candidates, and additional keyframes are sampled adaptively according to the complexity of the operations within each stage. For semantic annotation, we use Qwen3.8 to generate structured semantic labels. For visual annotation, we use the object descriptions in these labels to prompt SAM 3 (Carion et al., 2025) to obtain bounding boxes and segmentation masks for target objects, grippers, and obstacles. We then use UniDepth V2 (Piccinelli et al., 2025) to estimate image depth and extract

Table 1: Performance comparison on LIBERO (Success Rates,%).

Methods	VLM Size	Spatial	Object	Goal	Long	Average
TraceVLA	7B	84.6	85.2	75.1	54.1	74.8
openVLA	7B	84.7	88.4	79.2	53.7	76.5
SpatialVLA	4B	88.2	89.9	78.6	55.5	78.1
WorldVLA	7B	87.6	96.2	83.4	60.0	81.8
CoT-VLA	7B	87.5	91.6	87.6	69.0	83.9
ThinkAct	7B	88.3	91.4	87.1	70.9	84.4
SmolVLA	500M	93.0	94.0	91.0	77.0	88.8
DreamVLA	345M	97.5	94.0	89.5	89.5	92.6
π_0	3B	96.8	98.8	95.8	85.2	94.2
UniVLA	7B	95.4	98.8	93.6	94.0	95.5
$\pi_{0.5}$	3B	**98.8**	98.2	98.0	92.4	96.9
GR00T-N1.7	2B	97.7	97.5	<u>98.5</u>	94.4	97.0
openVLA-OFT	7B	97.6	98.4	97.9	94.6	97.1
VLA-Adapter	0.5B	<u>97.8</u>	99.2	97.2	<u>95.0</u>	97.3
ZR-0	2B	97.4	<u>99.4</u>	97.8	**96.4**	<u>97.8</u>
VLA-Δ	**2B**	**98.8**	**99.8**	**98.6**	**96.4**	**98.4**

depth information near the center of each target object. Finally, we align and integrate semantic and geometric information from current and future time steps according to predefined rules, producing unified physical state-transition annotations and constructing training samples from them. For optical flow annotation, we use MegaFlow (Zhang et al., 2026a) to estimate dense two-dimensional displacements offline between the current and subsequent frames from the same camera view.

Quality control is integrated throughout the pipeline. We first use automated checks to identify formatting errors, temporal misalignment, invalid coordinates, inconsistent object identities, and depth anomalies. GPT-5.6 then reviews complex cases or conflicting annotations against visual evidence on a field-by-field basis and flags uncertain information. Finally, we conduct a stratified manual audit to assess annotation quality. We describe the annotation pipeline in detail in Appendix B.3.

3 Experiments

3.1 Experimental Setup

Implementation and training. VLA-Δ is built on the GR00T N1 architecture. The vision-language model is initialized with the weights of ZR-0_{base} (Li et al., 2026). We pretrain the model with the joint training objective on demonstrations from DROID, MolmoAct, and RH20T, annotated using the automated annotation pipeline described in Section 2.3.

Simulation and real-world platforms. We evaluate VLA-Δ on the LIBERO (Liu et al., 2023) and RoboTwin 2.0 (Chen et al., 2025) and assess its generalization on LIBERO-Plus (Fei et al., 2025). We further conduct real-world experiments on the AgileX PiPER 6-DoF dual-arm robot platform. Task success rate serves as the primary evaluation metric. Detailed training and evaluation settings are provided in Appendix C.

3.2 Simulation Benchmark Evaluation

LIBERO and LIBERO-Plus. We first evaluate the model on LIBERO. As shown in Table 1, VLA-Δ achieves an average success rate of **98.4%**, the highest among all compared methods. To further evaluate generalization under out-of-distribution environmental changes, we test the model on four perturbation types in LIBERO-Plus: Language, Light, Background, and Layout.

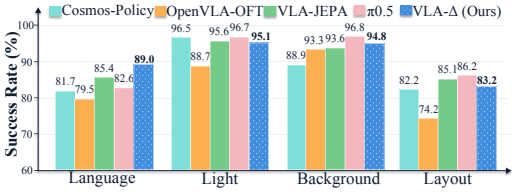


Figure 4: LIBERO-Plus performance under four different perturbations.

As shown in Figure 4, VLA-Δ maintains a high task success rate across all perturbation settings and remains competitive with leading methods. These results indicate that VLA-Δ achieves strong

Table 2: Performance comparison on RoboTwin 2.0 (Success Rates, %) under Clean and Random settings. Full results are provided in Appendix C.2.

Method	AdjustBottle		BeatBlockHammer		BlocksRankingSize		ClickAlarmclock		HangingMug		...	AVG	
	Clean	Random	Clean	Random	Clean	Random	Clean	Random	Clean	Random		Clean	Random
X-VLA	**100**	**99**	92	88	67	74	99	99	23	27	...	72.88	72.84
π_0	99	95	79	84	14	5	77	68	14	11	...	65.92	58.4
$\pi_{0.5}$	**100**	**99**	**96**	**93**	49	26	98	89	18	17	...	82.74	76.76
Motus	89	93	95	88	75	63	**100**	**100**	38	**38**	...	88.52	87.02
ZR-0	**100**	**99**	85	88	70	**81**	96	82	35	33	...	88.70	87.98
VLA-Δ	**100**	98	90	86	**80**	74	92	94	**40**	34	...	**90.06**	**89.74**

performance on the standard LIBERO benchmark without substantially compromising its ability to adapt to changes in visual appearance, background, and layout.

Notably, under the Language perturbation, VLA-Δ achieves **89.0%**, outperforming $\pi_{0.5}$ at **82.6%** by **6.4 percentage points**. This result suggests that the explicit semantic information provided by the Future-State Difference Text enables the policy to rely more on task-relevant state transitions when interpreting instructions, thereby reducing its sensitivity to superficial variations in language.

RoboTwin 2.0. We further evaluate VLA-Δ on RoboTwin 2.0. As shown in Table 2, VLA-Δ achieves average success rates of 90.06% and 89.74% under the Clean and Random settings, respectively, outperforming all compared methods in both settings. These results exceed those of ZR-0, the strongest baseline in both settings, by 1.36 and 1.76 percentage points. From Clean to Random, the average success rate of VLA-Δ decreases by only 0.32 percentage points, compared with 0.72 for ZR-0 and 1.50 for Motus. This indicates that the model achieves higher success rates while adapting effectively to scene randomization.

Under the Clean setting, VLA-Δ achieves the highest success rates on BlocksRankingSize and HangingMug among all compared methods. BlocksRankingSize requires arranging three blocks by size through repeated grasping, transport, and placement, with successive operations adjusting spatial relations according to earlier placement outcomes. HangingMug involves rotation, regrasping, and hanging, requiring coordinated changes in object poses and transitions among tabletop support, gripper grasping, and rack support. The state reached at each stage influences the conditions for subsequent operations. These two tasks therefore emphasize progressive spatial rearrangement and contact transitions across multiple stages, respectively. The observed gains support our central premise that explicitly learning changes in task progress, spatial motion, and interaction states helps connect task semantics with continuous action generation.

Overall, VLA-Δ outperforms the compared approaches that use intermediate reasoning or future visual state prediction on downstream tasks. These results suggest that treating task-relevant motion and interaction changes as explicit learning targets guides action generation more effectively.

3.3 Real-Robot Evaluation

Figure 5 presents the real-robot platform, task scenarios, and experimental results. We design five manipulation tasks to evaluate obstacle avoidance, long-horizon execution, fine-grained manipulation, spatial reasoning, and generalization under environmental changes. Each task is evaluated over 40 independent episodes, for a total of 200 trials.

Task success rates are reported in Figure 5(b). VLA-Δ achieves an average success rate of 68.0% across the five tasks. On the obstacle avoidance task, it achieves 42.5%, the highest success rate among all compared methods. This result is consistent with the explicit Object Constraints supervision in Future-State Difference Text. These annotations describe obstacles to avoid and object relationships to preserve, helping the model account for manipulation constraints and potential collision risks during action generation.

VLA-Δ also achieves the highest success rates among compared methods on the spatial reasoning and fine-grained manipulation tasks. This advantage may reflect spatial position annotations in Structured Transition Slots and visual details from optical flow. On the long-horizon execution and

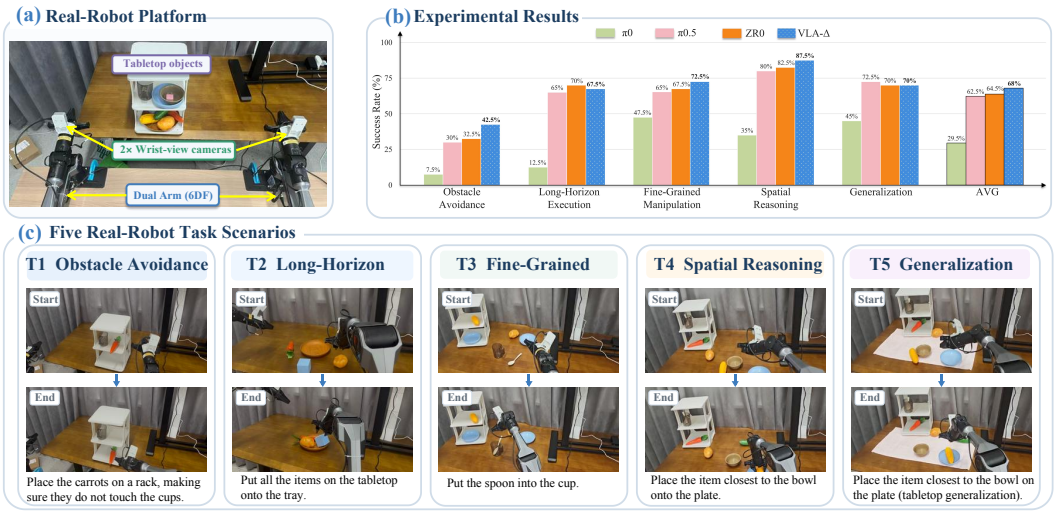


Figure 5: Real-robot evaluation on the AgileX PiPER 6-DoF platform. (a) Dual-arm robot platform. (b) Real-robot experimental results (Success Rates, %). (c) Overview of the five tasks.

Table 3: Ablation results on 40 LIBERO tasks.

Variant	Query	Pretraining	Difference Text	Slots	Optical Flow	LIBERO Avg. (%)
Dense VLM Features	No	No	No	No	No	95.70
Dense VLM Features	No	Yes	No	No	No	95.85
Difference Queries Only	Yes	No	No	No	No	96.25
Difference Queries Only	Yes	Yes	No	No	No	96.45
Query + Difference Text	Yes	Yes	Yes	No	No	97.20
Query + Structured Slots	Yes	Yes	No	Yes	No	96.90
Query + Optical Flow	Yes	Yes	No	No	Yes	97.35
VLA-Δ (Full)	Yes	Yes	Yes	Yes	Yes	**98.40**

environmental generalization tasks, VLA-Δ achieves the second-highest success rates. These results indicate continued effectiveness on complex long-horizon tasks and under environmental changes.

Overall, the experimental results support the effectiveness of multilevel physical state-transition supervision in helping policies learn task-relevant state changes and their constraints, and translate this information into continuous control in real-world environments.

3.4 Ablation Experiments

To analyze the contributions of Physical State Difference Queries, Future-State Difference Text, Structured Transition Slots, and Dense Optical Flow, we conduct ablation experiments on LIBERO. All variants use only the action-learning objective during downstream fine-tuning.

As shown in Table 3, compared with the baseline using dense vision-language features, introducing the query mechanism improves the success rate by 0.55 percentage points, indicating that compact query representations support action generation. To distinguish the effects of pretraining data from those of auxiliary supervision, we further pretrain both configurations using only action supervision, which yields limited performance gains.

We then introduce Future-State Difference Text, Structured Transition Slots, and Dense Optical Flow separately, and all three configurations yield further performance improvements. We investigate why Structured Transition Slots improve model performance by extracting the hidden states of Slot Queries from models pretrained with and without structured slot supervision and projecting them into two dimensions using t-SNE. The nine slot targets use seven fixed Query groups. Task progress and phase share G12, while gripper state and contact transitions share G78, with separate output branches for each target. Appendix D.1 details the grouping, prediction heads, and supervision

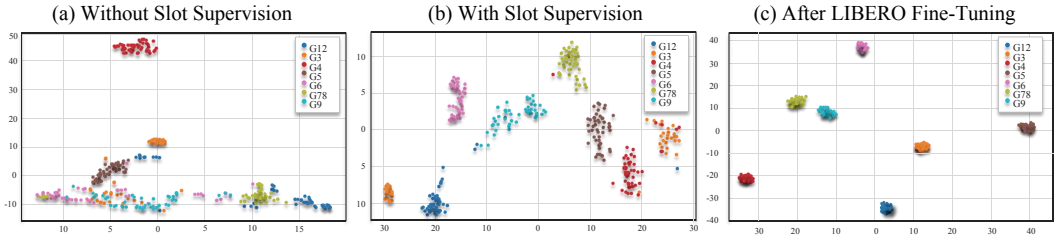


Figure 6: t-SNE visualization of Slot Query hidden states.

objectives. As shown in Figures 6(a) and 6(b), query representations are relatively dispersed without structured supervision during pretraining. With Structured Transition Slots, the query groups form more compact clusters with clearer boundaries. This observation suggests that structured slot supervision enhances the semantic separability of query representations.

To examine whether this structure transfers to downstream tasks, we extract the Slot Query hidden states after task-specific fine-tuning on LIBERO. As shown in Figure 6(c), the Slot Queries still form seven compact, well-separated clusters, although fine-tuning uses only the action-generation objective. This indicates that the query grouping structure established during pretraining persists through downstream fine-tuning, suggesting that the semantic prior introduced by Structured Transition Slots can continue to guide the learning of task-relevant transition representations.

Finally, the complete VLA-Δ model combines all three supervision types, achieving an average success rate of **98.4%**. This result suggests that multilevel physical state-transition supervision jointly shapes more comprehensive transition representations, improving task execution.

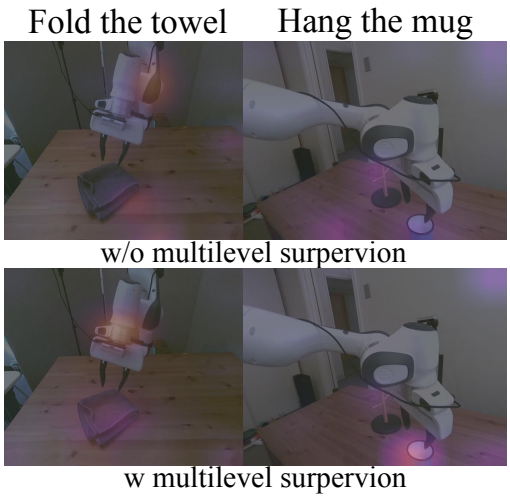


Figure 7: Query attention maps for towel folding and mug hanging.

We also visualize query attention distributions over the input images for the complete VLA-Δ model and the variant without the three transition-supervision signals. As shown in Figure 7, queries trained with all three types of supervision attend more strongly to the target object, gripper, contact regions, and other task-relevant objects. In contrast, the model without transition supervision exhibits more diffuse attention and assigns greater attention to task-irrelevant background regions.

4 Conclusion

We propose VLA-Δ, a general framework that uses physical changes between current and future states as explicit learning targets. Using robot demonstrations from multiple sources, we develop an automated annotation pipeline to construct Future-State Difference Text, Structured Transition Slots, and Optical Flow annotations, supervising transition representations through task semantics, structured state variables, and dense visual motion. These supervision signals are used only during pretraining to learn representations that support downstream policy learning through action-only fine-tuning, while deployment requires no decoding of intermediate supervision targets. Experiments demonstrate that VLA-Δ achieves competitive performance on LIBERO and RoboTwin and the highest average success rate among the compared methods in real-robot experiments. Ablation studies further validate the contributions of individual supervision signals, indicating that explicitly learning physical state changes provides an effective representational foundation for downstream continuous action generation.

AI USE STATEMENT

Generative AI tools were used for language editing and polishing, translation, LaTeX formatting, compilation troubleshooting, and the retrieval and discovery of related work. AI tools also supported data annotation and quality auditing, including semantic annotation generation, review of ambiguous or conflicting annotations against visual evidence, and bounding-box reference generation for selected audit fields, as detailed in Appendix B.3. The authors reviewed all AI-assisted work and take responsibility for the accuracy and integrity of the final manuscript, references, data, results, and claims.

REFERENCES

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

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. Rt-h: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024.

Homanga Bharadhwaj, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani. Gen2act: Human video generation in novel scenarios enables generalizable robot manipulation. *arXiv preprint arXiv:2409.16283*, 2024a.

Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision*, pp. 306–324. Springer, 2024b.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

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

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauzá, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving generalist agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

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

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025a.

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

Junhao Cai, Zetao Cai, Jiafei Cao, Yilun Chen, Zeyu He, Lei Jiang, Hang Li, Hengjie Li, Yang Li, Yufei Liu, et al. Internvla-a1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint arXiv:2601.02456*, 2026.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, et al. Lawam: Latent world action models for efficient dynamics-aware robot policies. *arXiv preprint arXiv:2606.15768*, 2026.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

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

StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

Shengliang Deng, Mi Yan, Songlin Wei, Haixin Ma, Yuxin Yang, Jiayi Chen, Zhiqi Zhang, Taoyu Yang, Xuheng Zhang, Wenhao Zhang, et al. Graspvla: a grasping foundation model pre-trained on billion-scale synthetic action data. *arXiv preprint arXiv:2505.03233*, 2025.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Jiafei Duan, Wentao Yuan, Wilbert Pumacay, Yi Ru Wang, Kiana Ehsani, Dieter Fox, and Ranjay Krishna. Manipulate-anything: Automating real-world robots using vision-language models. *arXiv preprint arXiv:2406.18915*, 2024.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, et al. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026.

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

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, 2025. *URL* `https://arxiv.org/abs/2507.16815`, 2025.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

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

Haoyang Li, Guanlin Li, Youhe Feng, Chen Zhao, Zhuoran Wang, Yang Li, Qizhe Wei, Shifeng Bao, Haitao Shen, Yihan Zhao, et al. Training vision-language-action models with dense embodied chain-of-thought supervision. *arXiv preprint arXiv:2606.30552*, 2026.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024a.

Xiaoqi Li, Mingxu Zhang, Yiran Geng, Haoran Geng, Yuxing Long, Yan Shen, Renrui Zhang, Jiaming Liu, and Hao Dong. Manipllm: Embodied multimodal large language model for object-centric robotic manipulation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18061–18070. IEEE, 2024b.

Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong, Hanbo Zhang, and Huaping Liu. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*, 3(7):8, 2024c.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. In *International Conference on Learning Representations*, volume 2024, pp. 26703–26721, 2024d.

Yihan Lin, Jiawei He, Shifeng Bao, Chen Zhao, Yang Li, Xiaobo Wang, Yan Wang, Cheng Chi, and Jing Zhang. Jepa-wam: Learning vision-language-action policies with joint-embedding world modeling. *arXiv preprint arXiv:2608.09381*, 2026.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025.

Tianyi Lu, Hui Zhang, Zijie Diao, Junke Wang, Shengqi Xu, Xingyao Lin, Guojin Zhong, Ziyi Ye, Peng Wang, Zuxuan Wu, et al. Thinkingvla: Interleaved vision and language reasoning for robotic manipulation. *arXiv preprint arXiv:2606.17937*, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

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

Luigi Piccinelli, Christos Sakaridis, Yung-Hsu Yang, Mattia Segu, Siyuan Li, Wim Abbeloos, and Luc Van Gool. Unidepthv2: Universal monocular metric depth estimation made simpler. *arXiv preprint arXiv:2502.20110*, 2025.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 18567–18602, 2026.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. In *European Conference on Computer Vision*, pp. 478–497. Springer, 2026.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. *arXiv preprint arXiv:2401.00025*, 2023.

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

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, volume 2024, pp. 10641–10662, 2024.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026.

Chengbo Yuan, Chuan Wen, Tong Zhang, and Yang Gao. General flow as foundation affordance for scalable robot learning. *arXiv preprint arXiv:2401.11439*, 2024a.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024b.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

Dingxi Zhang, Fangjinhua Wang, Marc Pollefeys, and Haofei Xu. Megaflow: Zero-shot large displacement optical flow. *arXiv preprint arXiv:2603.25739*, 2026a.

Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. Up-vla: A unified understanding and prediction model for embodied agent. *arXiv preprint arXiv:2501.18867*, 2025.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, et al. Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. *Advances in Neural Information Processing Systems*, 38: 24195–24228, 2026b.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1702–1713. IEEE, 2025.

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

Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 60580–60606, 2026.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *International Conference on Learning Representations*, volume 2025, pp. 54277–54296, 2025.

A Related Work

A.1 Vision-Language-Action Models

Vision-language-action (VLA) models combine pretrained vision-language models with robot demonstrations to connect multimodal understanding with action generation. Related research provides complementary foundations for generalist robot policy learning (Brohan et al., 2022; Li et al., 2024a; Wen et al., 2025; Qu et al., 2025; Li et al., 2024d; Team et al., 2024; Bousmalis et al., 2023; Li et al., 2024c; Bu et al., 2025a; Deng et al., 2025; Bu et al., 2025b; Wu et al., 2026; Shukor et al., 2025; Intelligence et al., 2026). RT-2 (Brohan et al., 2023) encodes actions as discrete tokens and jointly trains on robot trajectories and vision-language tasks. This formulation enables semantic knowledge from visual and linguistic pretraining to support robot control. OpenVLA (Kim et al., 2024) trains an open generalist policy on large-scale robot demonstrations, demonstrating the value of vision-language pretraining for multitask manipulation and policy adaptation. For continuous control, π_0 and $\pi_{0.5}$ (Black et al., 2024; Intelligence et al., 2025) combine vision-language pretraining with flow matching to generate continuous action chunks. GR00T N1 (Bjorck et al., 2025) combines a vision-language backbone with a flow-matching action expert, providing the architectural basis for our model. Action representations and generative policy learning provide complementary foundations for continuous control (Peebles & Xie, 2023; Lipman et al., 2022; Chi et al., 2025; Zhao et al., 2023; Liu et al., 2025; Kim et al., 2025; Pertsch et al., 2025; Yang et al., 2026).

VLA-Δ focuses on explicit supervision of physical changes between current and future states to shape transition representations for continuous action generation. These supervision signals are used only during pretraining, while downstream fine-tuning relies on action supervision.

A.2 Embodied Reasoning

Embodied reasoning links high-level task semantics with action execution through intermediate plans, spatial reasoning, and grounded representations (Ahn et al., 2022; Belkhale et al., 2024; Huang et al., 2023; Yuan et al., 2024b; Driess et al., 2023; Li et al., 2024b; Huang et al., 2024; Duan et al., 2024; Fang et al., 2026; Zheng et al., 2025; Shi et al., 2026). ECoT (Zawalski et al., 2024) generates task plans, subgoals, and motion descriptions before action prediction, grounding

reasoning in the robot state. ThinkAct (Huang et al., 2025) uses goal-completion and trajectory-consistency rewards to learn embodied reasoning and compresses plans into latent representations that guide action policies. ZR-0 (Li et al., 2026) learns cross-embodiment representations through dense embodied chain-of-thought supervision and can skip reasoning generation during deployment. ThinkingVLA (Lu et al., 2026) interleaves forward textual reasoning, future-image prediction, and inverse reasoning within a unified autoregressive architecture. Forward reasoning selects a subgoal and guides visual forecasting, while inverse reasoning uses the predicted target state to infer spatial relations and action intent. Actions are generated from the combined reasoning context.

These methods show how explicit reasoning can relate task goals and spatial information to robot control. VLA-Δ focuses on task-relevant physical changes between current and future states, combining Future-State Difference Text, Structured Transition Slots, and optical flow to connect coherent semantic descriptions with dense visual motion supervision.

A.3 World Action Models

In robotic manipulation, world action models connect predictions of future observations or latent states with action generation. Broader work on predictive representations and policy learning provides complementary ways to condition actions on anticipated outcomes (Zhen et al., 2024; Cheang et al., 2024; Bharadhwaj et al., 2024a; Wu et al., 2024; Zhao et al., 2025; Zhang et al., 2025; Chen et al., 2026; Lin et al., 2026; Wen et al., 2023; Bharadhwaj et al., 2024b; Yuan et al., 2024a). WorldVLA (Cen et al., 2025) unifies action generation and future-image generation within an autoregressive framework, allowing information sharing between policy learning and world modeling. VLA-JEPA (Sun et al., 2026) predicts latent representations of future observations, learning transition information while reducing sensitivity to appearance and background variations. DreamVLA (Zhang et al., 2026b) forecasts dynamic, spatial, and semantic information to guide action prediction. Motus (Bi et al., 2026) unifies understanding, video generation, and action experts, using optical flow to learn latent actions. These approaches differ in whether future information is represented by reconstructed images, latent features, or complementary prediction targets, but all use predictive learning to support action generation beyond the current observation. Generating future images online, however, adds visual-token decoding or video-denoising costs.

VLA-Δ combines readable descriptions of task progress, spatial and contact changes, and physical constraints with structured transition slots and dense optical flow supervision. This joint supervision shapes a shared transition representation for action generation without requiring future-image or difference-text generation at deployment.

B Pretraining Dataset and Data Annotations

B.1 Pretraining Dataset

To learn visual, language, and physical state-transition representations for robot manipulation, we construct a pretraining dataset by selecting demonstrations from DROID, MolmoAct, and RH20T. We first convert all data sources into a unified trajectory format, then filter samples based on the completeness of visual observations, language instructions, robot states, and action records. The resulting dataset covers diverse tasks, scene layouts, robot configurations, and interaction types.

Table 4 summarizes the scale and collection characteristics of the three source datasets.

DROID contains approximately 76,000 demonstration trajectories across 564 scenes and 84 task categories, totaling approximately 350 hours of interaction. We retain demonstrations with complete timestamps, valid visual frames, and continuous action records, prioritizing trajectories that include approaching, grasping, transporting, placing, and releasing objects. These demonstrations increase the diversity of environments, objects, and tasks in the pretraining data.

MolmoAct contains over 10,000 high-quality manipulation trajectories covering household and tabletop settings. We extract visual observations, language instructions, robot states, and actions, retaining trajectories that clearly reflect task progress and subgoal changes. These data provide additional information about language conditioning, spatial relations, and multistage manipulation.

Table 4: Overview of the datasets used to construct the pretraining corpus.

Dataset	Frames	Hours	No. of Tasks	Introduction
MolmoAct (Household + Tabletop)	1,104,942	30.693	96	Everyday household and tabletop manipulation with a Franka robot, covering organizing, cleaning, opening and closing, grasping, and placement tasks; includes a fixed external view, a second external view, and a wrist-camera view.
DROID	27,630,375	511.674	49,630	Open-ended everyday manipulation with Franka robots across different data collectors, buildings, and real-world scenes; includes two external views and a wrist-camera view.
RH20T	3,898,056	108.279	141	Multi-robot real-world manipulation. The Stage05 release subset covers Flexiv, Franka, KUKA, and UR5 across seven configurations, 19 users, and 21 scene IDs; includes two external views, a wrist-camera view, and force/torque fields.

RH20T contains over 110,000 contact-rich manipulation sequences spanning multiple robot platforms, sensor configurations, and manipulation skills, with modalities including vision, force sensing, and audio. We use its visual observations, proprioceptive states, and continuous action records; the remaining modalities are not directly provided to the policy network. Grasping, pushing, insertion, and release sequences in this dataset provide additional examples of contact interactions, object state changes, and fine-grained manipulation.

The three data sources complement one another in scene diversity, task semantics, and physical interaction. After format standardization and filtering, the pretraining data cover diverse spatial layouts, motion patterns, and manipulation constraints, providing a foundation for learning multilevel physical state-transition representations across tasks and robot configurations.

B.2 Physical State-Transition Annotations

Figure 8 presents examples of physical state-transition annotations.



Figure 8: Physical state-transition annotation examples: a current–future observation pair with Future-State Difference Text and Structured Transition Slots (top), and slot annotations for loading a plate into a metal rack (middle) and hanging a mug (bottom).

B.3 Automated Data Annotation Pipeline

(1) Data Selection Strategy. We select demonstrations with clear goals, valid observations, complete state and action records, and reliable temporal alignment, considering diversity in tasks, objects, scenes, robot configurations, and viewpoints. Operations cover approaching, aligning, grasping, transporting, placing, releasing, pushing, and insertion. For partially corrupted trajectories, we retain continuous, semantically coherent segments and discard records with ambiguous target identities, action meanings, or temporal correspondences.

(2) Unified Trajectory Format. We convert all sources to LeRobot v3, organizing instructions, multiview images, proprioceptive states, actions, and timestamps. Time-series data are stored in Parquet and visual observations in videos, with metadata specifying features, frame rates, and trajectory boundaries. We retain source and trajectory identifiers, robot and camera configurations, calibration information, and field dimensions, units, coordinate frames, and validity indicators. Modalities are aligned by timestamps; missing or unreliably aligned fields are marked invalid.

(3) Data Cleaning. Language instruction checks. We identify empty, garbled, truncated, or inconsistent descriptions, correcting recoverable instructions and excluding unrecoverable records from language-conditioned training. Language standardization preserves original text and verifies semantic consistency. Future-state difference descriptions remain supervision labels, separate from input instructions.

Temporal consistency checks. We inspect timestamp ordering, duplicates, missing frames, and cross-modal offsets. Different video and control frequencies are permitted when reliably aligned. Temporal gaps define segment boundaries.

Visual quality checks. We identify undecodable, black, overexposed, blurred, or uninformative views. Brief occlusions are retained when neighboring frames or other views provide evidence; otherwise, affected annotations or segments are excluded.

State and action checks. We inspect missing dimensions, non-finite values, jumps, and out-of-range records using sampling intervals and robot configurations. Unclear action types, rotation representations, units, or coordinate frames are not speculatively converted.

Trajectory integrity checks. We identify abnormal lengths, duplicates, and truncation, distinguishing valid waiting and sustained contact from recording stalls. When applicable, training and validation sets are split by trajectory before extracting windows to prevent leakage.

(4) Adaptive Keyframe Selection and Semantic Annotation. Qwen3.8-27B identifies operation phases from instructions and videos, selecting boundaries and surrounding frames. Sampling is denser for precise alignment, insertion, gripper changes, and contact transitions, and sparser for smooth transport, while preserving task-critical events.

Using keyframes and neighboring clips, the model annotates objects, phases, contact relations, interaction regions, state changes, constraints, and obstacles. Annotations distinguish current states, observed future changes, and goals, as well as explicit and inferred constraints. Fields retain object identifiers, timestamps, supporting frames, and uncertainty indicators.

(5) Visual Localization, Depth, and Optical Flow Annotation. SAM 3 produces bounding boxes and masks for targets, grippers, and obstacles. Neighboring frames and tracks maintain object identities through similar-instance ambiguity and occlusion.

UniDepth V2 estimates depth, summarized by the local median within valid target-mask regions near the two-dimensional center. This measures visible-surface depth, not the object's geometric center. Severe occlusion or unstable estimates invalidate the field.

MegaFlow estimates dense two-dimensional displacement offline between current and subsequent frames from the same viewpoint. Frame identifiers and intervals maintain correspondence with transition descriptions.

(6) Annotation Integration and Training Sample Construction. Shared records for current–future state pairs contain intervals, object identities, phases, positions, contacts, changes, and constraints. They generate Future-State Difference Text and Structured Transition Slots, providing semantic descriptions and structured categories, values, and validity indicators. Optical flow supplies dense motion information, with all annotations sharing temporal and object correspondences.

Keyframes locate events, while the model's prediction horizon determines future endpoints. Information derived from future observations serves only as pretraining supervision, never as input for current action prediction.

(7) Stratified Quality Review and Traceability. Programmatic checks verify completeness, timing, coordinates, identities, and cross-annotation consistency. An independent multimodal model reviews difficult or conflicting samples field by field against visual evidence, retaining uncertainty where necessary. Correctable errors are reannotated, invalid fields are masked, and entire samples are discarded only when essential timing or target identities cannot be established.

Manual inspections are stratified by source, task, phase, and quality level, with random checks of automatically accepted samples to estimate missed errors. Findings guide revisions to prompts,

Table 5: Counts and proportions from the audit of physical state-transition annotations.

Evaluation target	Reference criteria	Valid count	Percentage
Text: Task Temporal	Human-verified task phases, task progress, and subgoal changes	290	96.67%
Text: Spatial Motion	Human-verified object and gripper positions, relative relations, and their changes	289	96.33%
Text: Contact Interaction	Gripper opening and closing, contact establishment, maintenance, and release confirmed in video	293	97.67%
Text: Object Constraints	Manipulation constraints specified by task instructions or supported by visual evidence	298	99.33%
Slot 1: Task progress	Human reference values in $[0, 1]$ based on a consistent definition of progress	292	97.33%
Slot 2: Task phase	Human labels based on predefined phase categories	293	97.67%
Slot 3: Target position	Target-object bounding boxes annotated by GPT-5.6	298	99.33%
Slot 4: Gripper position	Gripper bounding boxes annotated by GPT-5.6	288	96.00%
Slot 5: Motion and relative displacement	Reference displacements from manually matched points or reliable calibration, in the same coordinate frame as the predictions	279	93.00%
Slot 6: Interactable position	Human-annotated regions for grasping, contact, or insertion	294	98.00%
Slot 7: Gripper state transitions	Human-verified gripper opening and closing transition labels	298	99.33%
Slot 8: Contact transitions	Human-verified labels for contact establishment, maintenance, and release	296	98.67%
Slot 9: Obstacles and risk	Human-annotated obstacle boxes and risk labels based on consistent risk criteria	287	95.67%

Note: The audit includes 300 current–future state pairs, with 100 each from DROID, MolmoAct, and RH20T. Percentages equal the valid count divided by 300, rounded to two decimal places. Rows assess different aspects of the same samples, so their counts are not additive. Reference boxes for Slots 3 and 4 are generated by GPT-5.6 and do not constitute independent human ground truth. These percentages do not represent bounding-box intersection over union or displacement error.

rules, and guidelines and reprocessing of affected data. Sources, model versions, evidence frames, review decisions, and revision histories support traceability and reproducibility.

(8) Annotation Audit Statistics. We sample 300 current–future state pairs, with 100 each from DROID, MolmoAct, and RH20T, and audit the four aspects of Future-State Difference Text and the nine Structured Transition Slot categories. Table 5 summarizes the reference criteria, valid counts, and corresponding proportions of the sampled pairs for each category.

C Evaluation Protocol

C.1 LIBERO and LIBERO-Plus

Benchmark overview. LIBERO is a simulation benchmark for evaluating knowledge transfer and multitask manipulation in robotics. It includes four task suites, Spatial, Object, Goal, and Long, which assess spatial understanding, object manipulation, goal-conditioned task completion, and long-horizon execution, respectively. The benchmark provides diverse tabletop manipulation tasks with language instructions, covering grasping, placing, moving, stacking, opening, and combinations of these operations.

The four suites emphasize different capabilities.

- Spatial evaluates the model's understanding of relative positions, orientations, and spatial relations, such as placing an object to the left or right of another.

- Object evaluates target identification and object recognition, requiring the policy to select the correct object and perform the specified operation.
- Goal assesses whether the model achieves the target state specified by a language instruction, such as placing an object inside a container or moving it to a designated location.
- Long comprises sequences of subgoals, requiring the model to track task progress, execution history, and intermediate states over extended horizons.

Training data and fine-tuning. We perform multitask training using demonstrations from all four LIBERO task suites, with a total of 1,693 training trajectories.

Standard LIBERO evaluation. We evaluate the model on each of the four suites and report both suite-level and overall average success rates. Each task is evaluated over 50 independent episodes, yielding 2,000 evaluation episodes in total. Each episode uses a predefined initial scene configuration and random state. The model performs closed-loop control using only the current observations, task instruction, and robot state.

LIBERO-Plus generalization evaluation. To assess adaptation to out-of-distribution environmental changes, we further evaluate the model on LIBERO-Plus. We select four perturbation types and evaluate them across all four task suites, running 50 episodes per task under each perturbation type for a total of 8,000 episodes. These perturbations preserve the task goals and underlying manipulation logic while modifying the instructions or environment.

- Language varies instruction wording, sentence structure, or phrasing.
- Light changes lighting direction, brightness, or color.
- Background modifies background objects, tabletop textures, or scene appearance.
- Layout changes initial object configurations, relative positions, or scene arrangements.

We use no additional demonstrations from perturbed environments and perform no perturbation-specific fine-tuning for LIBERO-Plus. The model retains the parameters learned on standard LIBERO and uses the same input interface, action space, and maximum execution steps.

C.2 RoboTwin 2.0

Benchmark overview. RoboTwin 2.0 is a simulation benchmark for dual-arm robot manipulation, focusing on bimanual coordination, object interaction, spatial manipulation, and policy generalization under environmental changes. It includes 50 dual-arm manipulation tasks and supports multiple robot configurations and scene randomization. Scene generation covers diverse object categories and instances, tabletop layouts, backgrounds, lighting conditions, and language instructions. The official dataset provides a large object library and multiple structured domain randomization settings.

RoboTwin 2.0 provides two main settings.

- Clean uses relatively fixed scene appearances and object layouts to evaluate basic dual-arm manipulation in standard environments.
- Random introduces variations in object instances, initial positions, lighting, backgrounds, tabletop layouts, and instruction phrasing to assess generalization under distribution shifts.

The tasks involve bimanual grasping, coordinated transport, object arrangement, container manipulation, tool use, and multi-object organization, placing substantial demands on bimanual coordination, temporal planning, and contact control.

Training data. We train on 27,500 demonstration trajectories across all 50 tasks. Each task contributes 50 Clean demonstrations and 500 Random demonstrations.

Evaluation protocol. We evaluate all 50 tasks under both Clean and Random settings, running 100 episodes per task in each setting for a total of 10,000 evaluation episodes. During inference, the action chunk length is set to 16. The policy executes the first eight actions of each predicted chunk before generating a new chunk from updated observations, enabling closed-loop control.

Table 6 reports the complete per-task results under the Clean and Random settings of RoboTwin 2.0.

Table 6: Per-task success rates (%) on RoboTwin 2.0 under the Clean and Random settings.

Task	π_0		$\pi_{0.5}$		X-VLA		Motus		ZR-0		VLA-Δ	
	Cle	Ran.	Cle	Ran.	Cle	Ran.	Cle	Ran.	Cle	Ran.	Cle	Ran.
AdjustBottle	99	95	100	99	100	99	89	93	100	99	100	98
BeatBlockHammer	79	84	96	93	92	88	95	88	85	88	90	86
BlocksRankingRGB	80	63	92	85	83	83	99	97	92	91	95	94
BlocksRankingSize	14	5	49	26	67	74	75	63	70	81	80	74
ClickAlarmclock	77	68	98	89	99	99	100	100	96	82	92	94
ClickBell	71	48	99	66	100	100	100	100	90	83	100	100
DumpBinBigbin	88	83	92	97	79	77	95	91	94	93	97	96
GrabRoller	98	94	100	100	100	100	100	100	100	100	98	94
HandoverBlock	47	31	66	57	73	37	86	73	93	87	93	92
HandoverMic	97	97	98	97	0	0	78	63	100	99	95	98
HangingMug	14	11	18	17	23	27	38	38	35	33	40	34
LiftPot	80	72	96	85	99	100	96	99	96	98	96	100
MoveCanPot	68	48	51	55	89	86	34	74	85	81	94	78
MovePillbottlePad	67	46	84	61	73	71	93	96	98	99	97	100
MovePlayingcardAway	74	65	96	84	93	98	100	96	99	94	96	98
MoveStaplerPad	41	24	56	42	78	73	83	85	85	92	87	91
OpenLaptop	71	81	90	96	93	100	95	91	96	99	98	90
OpenMicrowave	4	32	34	77	79	71	95	91	94	92	95	94
PickDiverseBottles	69	31	81	71	58	36	90	91	90	88	93	92
PickDualBottles	59	37	93	63	47	36	96	90	97	98	92	92
PlaceA2BLeft	43	47	87	82	48	49	82	79	83	80	85	84
PlaceA2BRight	39	34	87	84	36	36	90	87	86	87	85	89
PlaceBreadBasket	62	46	77	64	81	71	91	94	90	93	91	95
PlaceBreadSkillet	66	49	85	66	77	67	86	83	92	85	90	89
PlaceBurgerFries	81	76	94	87	94	94	98	98	98	98	96	100
PlaceCanBasket	55	46	62	62	49	52	81	76	66	62	80	64
PlaceCansPlasticbox	63	45	94	84	97	98	98	94	86	85	94	96
PlaceContainerPlate	97	92	99	95	97	95	98	99	99	98	97	100
PlaceDualShoes	59	51	75	75	79	88	93	87	90	95	92	96
PlaceEmptyCup	91	85	100	99	100	98	99	98	97	97	96	99
PlaceFan	66	71	87	85	80	75	91	87	85	78	84	83
PlaceMousePad	20	20	60	39	70	70	66	68	85	83	87	86
PlaceObjectBasket	67	70	80	76	44	39	81	87	75	77	75	79
PlaceObjectScale	57	52	86	80	52	74	88	85	88	89	89	93
PlaceObjectStand	82	68	91	85	86	88	98	97	90	91	90	88
PlacePhoneStand	49	53	81	81	88	87	87	86	85	81	84	83
PlaceShoe	76	76	92	93	96	95	99	97	98	97	97	100
PressStapler	44	37	87	83	92	98	93	98	90	92	92	90
PutBottlesDustbin	65	56	84	79	74	77	81	79	82	79	82	81
PutObjectCabinet	73	60	80	79	46	48	88	71	82	76	81	80
RotateQRCode	74	70	89	87	34	33	89	73	78	85	83	87
ScanObject	55	42	72	65	14	36	67	66	86	85	88	87
ShakeBottleHorizontally	98	92	99	99	100	100	100	98	100	100	100	98
ShakeBottle	94	91	99	97	99	100	100	97	100	100	100	100
StackBlocksThree	72	52	91	76	6	10	91	95	86	88	88	92
StackBlocksTwo	93	79	97	100	92	87	100	98	95	91	94	93
StackBowlsThree	77	75	77	71	76	86	79	87	79	88	83	87
StackBowlsTwo	94	95	95	96	96	93	98	98	94	92	97	96
StampSeal	46	33	79	55	76	82	93	92	92	92	92	95
TurnSwitch	41	42	62	54	40	61	84	78	83	78	83	82
Average	**65.92**	**58.4**	**82.74**	**76.76**	**72.88**	**72.84**	**88.52**	**87.02**	**88.7**	**87.98**	**90.06**	**89.74**

C.3 Real-Robot Evaluation

Experimental platform. To evaluate the effectiveness of VLA-Δ in real-world environments, we conduct experiments on the AgileX PiPER 6-DoF dual-arm robot platform. The platform comprises left and right robotic arms with end effectors, three cameras (one head camera and two wrist cameras), and robot proprioceptive state feedback.

Task setup. We design five tasks to evaluate physical constraint compliance, long-horizon execution, fine-grained manipulation, spatial-relation reasoning, and generalization under environmental changes, respectively.

The first four tasks are performed in a standard tabletop environment. The fifth task keeps the language instruction and task goal unchanged while modifying only the tablecloth or tabletop appearance to evaluate generalization to visual background changes.

Table 7: Capability dimensions and task instructions for real-robot evaluation.

Capability Dimension	Task Instruction
Obstacle Avoidance and Physical Constraint Compliance	Place the carrots on a rack, making sure they do not touch the cups.
Long-Horizon Execution	Put all the items on the tabletop onto the tray.
Fine-Grained Manipulation	Put the spoon into the cup.
Spatial Reasoning	Place the item closest to the bowl onto the plate.
Generalization under Changed Tablecloth	Place the item closest to the bowl onto the plate.

Evaluation protocol. Each task is evaluated over 40 independent episodes, for a total of 200 real-robot trials. For each task, all episodes use the same task instruction and a predefined distribution of initial scene configurations, while specific object positions, poses, or background conditions may vary according to the experimental setup.

D Training Details

D.1 Structured Slot Heads and Supervision Objectives

Prediction head architecture and outputs. Each prediction branch consists of LayerNorm followed by a single linear projection:

$$u_{t,i} = W_i \, \mathrm{LN}_i\big(\bar{z}_{t,g(i)}\big) + b_i, \tag{3}$$

where $g(i)$ identifies the Query group assigned to target S_i, and $\bar{z}_{t,g(i)}$ is the mean-pooled final Query representation of that group. Depending on the target type, the output is then reshaped, passed through a sigmoid function, or transformed using a valid bounding-box parameterization.

S_1 and S_2 share the pooled representation but have separate LayerNorm and linear layers. S_7 and S_8 share both the pooled representation and LayerNorm, i.e., $\mathrm{LN}_7 = \mathrm{LN}_8$, while using two independent linear layers. The remaining branches each have their own normalization and linear projection parameters. The complete module contains nine linear layers and eight LayerNorm layers, with no additional activation function or dropout between normalization and linear projection.

Task progress (S_1) and task phase (S_2) describe task progression through continuous completion levels and discrete manipulation stages, respectively, and therefore share the Query representation of G12. Gripper state transitions (S_7) and contact transitions (S_8) both describe interaction events during manipulation and therefore share the Query representation of G78. By assigning semantically related supervision targets to the same Query group, we encourage the model to jointly encode task progression and interaction changes while retaining target-specific prediction capabilities through separate output branches. This sharing provides a soft semantic division of labor without requiring a deterministic one-to-one correspondence between related targets.

Table 8 summarizes the outputs of the nine target categories. All output shapes exclude the batch dimension.

Table 8: Structured Transition Slot outputs, Query groups, and supervision. Output shapes exclude the batch dimension.

Supervision Target	Query Group	Output Content	Output Shape	Output Constraints and Supervision
S_1: Task progress	G12	Task progress at the current and next keyframes	2	Sigmoid; Smooth L1 and a progress monotonicity penalty
S_2: Task phase	G12	Seven manipulation phases at two time steps	2×7	Classification logits; class-weighted cross-entropy with label smoothing
S_3: Target position	G3	Target-object boxes at two time steps	2×4	Valid box parameterization; Smooth L1 and GIoU
S_4: Gripper position	G4	Gripper boxes at two time steps	2×4	Valid box parameterization; Smooth L1 and GIoU
S_5: Motion and relative displacement	G5	Target, gripper, and relative 3D displacements	3×3	Standardized regression and displacement consistency in meters
S_6: Interactable position	G6	Two affordance contact points in the current frame	2×2	Sigmoid; pointwise Smooth L1
S_7: Gripper state transitions	G78	Four gripper opening/closing transition classes	4	Classification logits; class-weighted cross-entropy with label smoothing
S_8: Contact transitions	G78	Four contact-transition classes	4	Classification logits; class-weighted cross-entropy with label smoothing
S_9: Obstacles and risk	G9	Boxes, risk scores, and validity for three ordered annotation positions	3×6	Box loss, risk regression, and binary validity classification

Because Queries within each group are mean-pooled, all output components are jointly predicted from the same group representation. For example, the target-position branch predicts bounding boxes at two time steps, but no Query is exclusively assigned to the current box or the future box. Contact points, three-dimensional displacements, and obstacle positions follow the same readout scheme.

Task phase (S_2), gripper state transitions (S_7), and contact transitions (S_8) use fixed class vocabularies. Table 9 lists the classes in the order of their zero-based class IDs.

Table 9: Fixed class vocabularies for task phase, gripper state transitions, and contact transitions. Class IDs start at zero.

Target	Classes in ID Order and Meanings
S_2: Task phase	0: `align` (alignment) 1: `approach` (approaching) 2: `contact` (contact) 3: `manipulate` (object manipulation) 4: `release` (release) 5: `retract` (retraction) 6: `transport` (transport)
S_7: Gripper state transitions	0: `closed_to_open` (closed to open) 1: `maintain_closed` (remaining closed) 2: `maintain_open` (remaining open) 3: `open_to_closed` (open to closed)
S_8: Contact transitions	0: `break_contact` (breaking contact) 1: `establish_contact` (establishing contact) 2: `maintain_contact` (maintaining contact) 3: `remain_separated` (remaining separated)