OBJDREAM: GUIDING VISION-LANGUAGE-ACTION MODELS WITH OBJECT-LEVEL UNDERSTANDING AND FUTURE STATE PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models have emerged as a powerful paradigm for robotic manipulation, enabling robots to generate actions from visual observations and language instructions. However, most existing VLA models still operate on image-level visual representations inherited from pretrained vision-language backbones, while lacking fine-grained object understanding for manipulation. As a result, their robustness can degrade under out-of-distribution object layouts, clutter, and background variations. To address these limitations, we introduce ObjDream, an object-centric VLA framework centered on object-level understanding and future object state prediction. ObjDream uses a spatio-temporal object encoder to organize visual observations into temporal object slots, where persistent object identity and dynamic object state are represented through separate channels. These structured object representations are inserted into the token sequence of the vision-language model (VLM), where identity-guided routing modulates VLM attention to the corresponding object tokens. In addition, an action-conditioned object dynamics module predicts compact future object states relevant to manipulation during training, explicitly supervising object state evolution under robot actions without requiring future prediction at inference. Experiments on LIBERO, LIBERO-Plus, and RoboTwin 2.0 show that ObjDream improves generalization under diverse distribution shifts and scene perturbations. Real-world robot experiments further demonstrate that ObjDream transfers to physical manipulation settings with novel object arrangements and background changes.

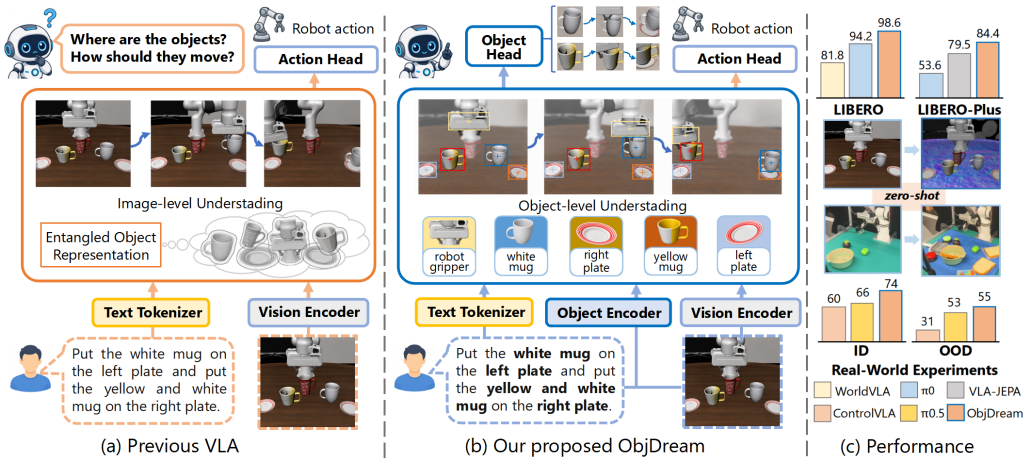


Figure 1: **Overview of ObjDream.** We introduce ObjDream, an object-centric VLA framework for robust manipulation. Unlike prior VLA models (a), which primarily rely on image-level understanding and entangled object representations, ObjDream (b) introduces explicit object-level representations and future object state prediction. This object-centric formulation improves manipulation robustness, leading to stronger performance than baseline methods (c).

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.

1 INTRODUCTION

Robust robot policies must reliably manipulate objects in out-of-distribution environments, where their appearance, position, and surrounding context may differ from those in the training data. VLA models have made rapid progress in generalist robot control by scaling pretrained vision-language backbones, robot data, and action decoders (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; NVIDIA et al., 2025; Wang et al., 2026b). To improve generalization, recent work has sought to preserve knowledge acquired during pretraining (Guo et al., 2026; Hancock et al., 2026), use high-level plans to guide low-level control (Li et al., 2025c), or anticipate scene changes through future visual or latent-state prediction (Zhang et al., 2025; Sun et al., 2026). Although these approaches improve generalization to some extent, most VLA models still lack fine-grained object-level understanding. Their visual representations, inherited from VLMs, remain largely image-level, leaving the identity and spatial state of task-relevant objects entangled with appearance and surrounding context. This entanglement can weaken out-of-distribution robustness under shifts in object appearance, spatial layout, and background context.

Object-centric approaches offer a natural way to strengthen object-level understanding by explicitly representing individual objects and their properties. Prior work incorporates object-aware visual representations to help policies focus on task-relevant regions (Zhu et al., 2023; Li et al., 2025a; Liang et al., 2026). Other methods extend object representations to relational reasoning (Hanyu et al., 2025), object-level memory (Chung et al., 2026), and long-horizon manipulation (Yang et al., 2026). Although these approaches demonstrate the value of object-level representations, they often encode persistent object identity and time-varying object state in the same slot or token. As appearance and geometry change, this coupling can obscure persistent identity cues by mixing them with dynamic object state. Recent dual-state slot models show that separating identity from dynamic appearance can improve temporal consistency in video perception by keeping identity information distinct from time-varying visual content (Tran et al., 2026). Building on this insight, we investigate whether persistent identity references can be represented alongside dynamic object states and further used to guide VLM attention for language-conditioned manipulation.

Beyond modeling the current visual observation, predictive models support manipulation by anticipating future observations or object states (Ferraro et al., 2025; Chen et al., 2025a; Villar-Corrales & Behnke, 2025). The choice of prediction target is important: reconstructing complete images requires modeling texture and background details that may be irrelevant to control (Zhang et al., 2025), while goal-image methods such as Goal-VLA introduce an additional synthesis stage before estimating the target object pose (Chen et al., 2025a). In contrast, object-centric world models show that scene evolution can be modeled using compact object representations (Jeong et al., 2025). Motivated by this trade-off, we use action-conditioned prediction of compact future object states, including geometry and visibility, as auxiliary supervision for VLA training without requiring future object state prediction at inference.

To this end, as illustrated in Fig. 1, we introduce ObjDream, an object-centric vision-language-action framework for improving manipulation generalization. The architecture of ObjDream builds on a pretrained VLM backbone and integrates a spatio-temporal object encoder, an object-level forward dynamics module, and a flow-matching action expert. Specifically, the spatio-temporal object encoder transforms visual observations into structured object slots, where persistent object identity and dynamic object state are represented through separate channels. These factorized object representations are mapped into the VLM hidden space and inserted into the token sequence, where identity-guided routing modulates attention to the corresponding object tokens. The forward dynamics module predicts future object states during training, providing auxiliary supervision for future object state evolution under robot actions without requiring future prediction at inference. The object encoder is pretrained with automatically generated offline object supervision and frozen during VLA training. We optimize ObjDream with a joint objective combining action imitation with current and future object state supervision. Extensive evaluations on LIBERO, LIBERO-Plus, RoboTwin 2.0, and real-world robot experiments demonstrate competitive manipulation performance across diverse scene perturbations.

The main contributions of this paper are listed below:

- We present ObjDream, an object-centric VLA framework centered on object-level understanding and future object state prediction to improve generalization under perturbations. ObjDream

introduces a spatio-temporal object encoder that explicitly represents persistent object identity and dynamic object state through separate channels. The resulting representations are integrated into the VLM hidden space with a novel identity-guided routing mechanism for robust language-conditioned manipulation.

- We introduce an action-conditioned object-level forward dynamics module for modeling object evolution during manipulation. It predicts compact future object states during training to provide direct auxiliary object-level supervision without requiring future object state prediction at inference. We further model inter-object relations from object features and pairwise geometry to provide structured context for action generation.
- Extensive evaluations on LIBERO, LIBERO-Plus, RoboTwin 2.0, and real-world robot experiments demonstrate competitive manipulation performance across diverse scene and environmental perturbations. Ablations further assess the effects of the key architectural components.

2 Related Work

Vision-language-action models. VLA models combine pretrained language or vision-language backbones with action decoders to learn general-purpose manipulation from large-scale robot demonstrations (Brohan et al., 2022; 2023; O'Neill et al., 2024; Team et al., 2024; Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025). Recent work improves VLA performance through stronger action decoders and fine-tuning strategies (Chi et al., 2025; Zhao et al., 2023; Kim et al., 2025; Hou et al., 2025; NVIDIA et al., 2025; Qu et al., 2025), larger-scale cross-embodiment training (Cai et al., 2026; Team et al., 2026; Wu et al., 2026a;b; Wang et al., 2026b), and better preservation or exploitation of pretrained visual-semantic and spatial priors (Guo et al., 2026; Li et al., 2026a; Wang et al., 2026c). Predictive objectives further use future images, video tokens, subgoal images, or latent states to provide temporal supervision for manipulation (Zhao et al., 2025; Zhang et al., 2025; Cen et al., 2025b;a; Li et al., 2026b). Despite these advances, existing VLA models still largely operate on dense image-level visual representations, where object-level structure is not explicitly organized for manipulation generalization. ObjDream addresses this limitation by enhancing VLA policies with object-level understanding and future object state supervision.

World models for robot control. World models learn action-conditioned environment dynamics and support robot control by planning or learning policies from predicted futures in observation or latent spaces (Hafner et al., 2019; Wu et al., 2023). Recent work extends this paradigm through language-conditioned video generation (Du et al., 2023; Zhou et al., 2024b), pretrained visual latent dynamics (Zhou et al., 2024a; Assran et al., 2025), and world-action or unified video-action models that couple future prediction with executable actions (Ye et al., 2026; Li et al., 2026c). These approaches provide dynamics supervision by modeling anticipated scene evolution. However, prediction in pixel or video space requires modeling high-dimensional visual details and can introduce training or inference overhead. Recent approaches therefore explore more efficient formulations, including avoiding explicit future imagination at inference or exposing compact representations of scene changes (Yuan et al., 2026; Bai et al., 2026). ObjDream follows this direction by using compact future object state prediction as supervision during training, while directly generating actions at inference without explicit future object state decoding.

Object-centric representations for manipulation. Object-centric representation learning decomposes visual scenes into entity-level latent variables and has been widely studied in scene decomposition, temporal tracking, and structured dynamics (Locatello et al., 2020; Kipf et al., 2021; Wu et al., 2022). In robotic manipulation, object-level representations have been used to improve sample efficiency and enhance visuomotor attention and robust policy conditioning through object proposals, grounded geometry, object-centric prompting, and physically grounded object concepts (Zhu et al., 2023; Vo et al., 2025; Li et al., 2025b; Wei et al., 2026), enable efficient VLA learning through compact object-centric tokenization (Bendikas et al., 2025), and support structured control through object-centric dynamics and relational representations (Ferraro et al., 2025; Mosbach et al., 2024; Hanyu et al., 2025; Yang et al., 2026). Complementary approaches use structured affordance representations to provide action-relevant object grounding, interaction localization, and geometric guidance for VLA policies (Yu et al., 2026b). Recent predictive approaches further incorporate object-aware future representations, masks, or addressable slot states into world-action learning (Wang et al.,

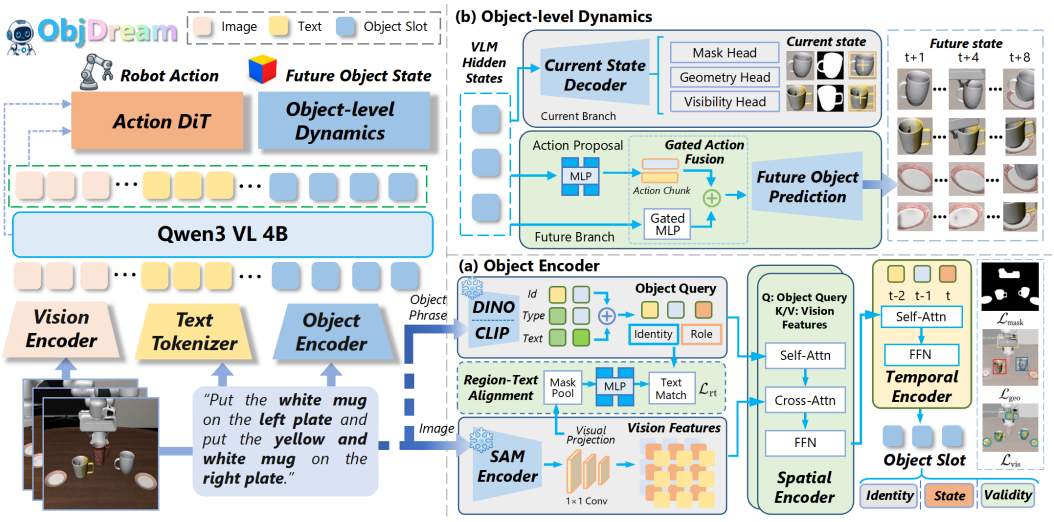


Figure 2: **Overview of the ObjDream architecture.** The model integrates three main components: (1) a spatio-temporal object encoder that constructs factorized object slots by separately representing persistent object identity and time-varying object state; (2) an object-level dynamics module that decodes current object states and predicts action-conditioned future object states; and (3) a flow-matching action expert that predicts robot action chunks. ObjDream enhances object-level understanding through factorized object representations and future object state modeling, enabling more generalizable language-conditioned manipulation under object and scene variations.

2026a; Yu et al., 2026a; Liu et al., 2026). Recent dual-state slot models separate per-frame appearance from persistent object identity, highlighting the value of disentangling identity from changing visual content in object-centric representations (Tran et al., 2026). From this perspective, ObjDream combines persistent identity references with dynamic object states and uses identity-guided routing to provide object-specific context for language-conditioned manipulation.

3 Method

Following the VLA policy formulation in Sec. 3.1, Fig. 2 illustrates the architecture of ObjDream. ObjDream combines a spatio-temporal object encoder, an object dynamics module, and a flow-matching action expert. The encoder produces structured object slots that are mapped to VLM object tokens, while identity-guided routing uses cached identity references to modulate attention to these tokens (Sec. 3.2). The object dynamics module provides future object state supervision during training (Sec. 3.3), and the action expert generates action chunks.

3.1 VLA Policy Formulation

We formulate ObjDream as an object-centric VLA policy that predicts continuous action chunks from language instructions, visual observations, proprioceptive states, and object-level embeddings. At control step t, the policy receives a language instruction x, a current multi-view observation I_t, and proprioceptive state s_t. The VLM consumes all policy views in I_t, while the object encoder operates on a designated external or third-person view with causal history $\mathbf{I}_t = (I^{\text{enc}}_{t-T+1}, \ldots, I^{\text{enc}}_t)$. At episode initialization, a phrase parser extracts a task-relevant phrase set P_0 from the instruction x and initial observation I_0. A spatio-temporal object encoder E_ω then produces structured object slots $O_t = E_\omega(\mathbf{I}_t, P_0)$. The VLA backbone f_ψ encodes these inputs into a multimodal context C_t, and the action expert parameterizes the conditional distribution over an action chunk $A_t = [a_t, \ldots, a_{t+H_a-1}]$:

$$C_t = f_\psi(x, I_t, s_t, O_t), \qquad p_\theta(A_t \mid x, I_t, s_t, O_t) = p_\theta(A_t \mid C_t). \tag{1}$$

Here, H_a denotes the action horizon. In our implementation, p_θ is instantiated by a flow-matching action expert (Lipman et al., 2022; Black et al., 2024). Given a demonstrated action chunk $A^\star_t$,

Gaussian noise $Z_t \sim \mathcal{N}(0, I)$, and flow time $\tau \sim \mathcal{U}[0, 1]$, the action loss is

$$A_t^\tau = (1-\tau)Z_t + \tau A_t^\star, \qquad \mathcal{L}_{\text{act}} = \mathbb{E}\Big[\|v_\theta(A_t^\tau, \tau; C_t) - (A_t^\star - Z_t)\|_2^2\Big]. \tag{2}$$

At inference, the learned velocity field is integrated to generate an action chunk, which is executed in a receding-horizon manner.

3.2 Object-Centric Tokenization and VLM Routing

Spatio-temporal object encoder. ObjDream uses a spatio-temporal object encoder E_ω to construct factorized object slots from the causal image window $\mathbf{I}_t$ and the cached phrase set P_0. At episode initialization, Qwen3-VL-8B (Bai et al., 2025) processes the instruction x and initial observation I_0 once to extract task-relevant object phrases, which are cached for all subsequent control steps. Query ordering is fixed across frames, with reserved robot queries and padding to N_s slots. For each phrase p_i, a frozen CLIP text encoder (Radford et al., 2021) produces f_i, and a learnable projection gives $\iota_i = \text{Normalize}(P_{\text{sem}}(f_i))$. The object query is initialized as

$$q_i^{(0)} = P_{\text{id}}(\iota_i) + P_{\text{text}}(f_i) + E_{\text{type}}(\chi_i), \tag{3}$$

where $P_{\text{id}}, P_{\text{text}}$ are learnable projections and E_{type} is a query type embedding. A frozen DINOv2 encoder (Oquab et al., 2023) extracts a visual reference d_i from the initial object region predicted by the encoder, which is cached throughout the episode. Robot queries use learned identities and type priors without the direct text term.

A spatial query module applies self-attention and cross-attention to learnably projected features from a frozen SAM2 image backbone (Ravi et al., 2025), followed by causal temporal encoding. The encoder outputs

$$O_t = E_\omega(\mathbf{I}_t, P_0) = \Big\{u_{i,t}^{\text{obj}}\Big\}_{i=1}^{N_s}, \qquad u_{i,t}^{\text{obj}} = (e_i^{\text{id}}, e_{i,t}^{\text{state}}, \nu_{i,t}). \tag{4}$$

The persistent identity $e_i^{\text{id}} = [\iota_i; d_i; r_i]$ combines semantic and visual references with a role embedding r_i, while $e_{i,t}^{\text{state}} = [c_{i,t}^{\text{app}}; g_{i,t}^{\text{geo}}; \tau_t]$ encodes dynamic appearance, geometry, and temporal information. Slot validity $\nu_{i,t}$ masks inactive or padded slots and is distinct from predicted visibility. During pretraining, regional features obtained by mask pooling from the projected SAM2 features, extracted before query cross-attention, are aligned with frozen CLIP text features. The objective is described in Sec. 3.4.

Factorized object injection. We map identity and dynamic state components separately into the VLM hidden space:

$$h_{i,t}^{\text{obj}} = \Big[P_{\text{id}}^{\text{inj}}(e_i^{\text{id}}); P_{\text{obj}}\left([P_{\text{state}}(e_{i,t}^{\text{state}}); \phi_\nu(\nu_{i,t})]\right)\Big]. \tag{5}$$

Here, $P_{\text{id}}^{\text{inj}}$, P_{state}, and P_{obj} are lightweight projections, and ϕ_ν embeds slot validity. The factorization is imposed at the encoder output and initial object-token construction; subsequent VLM contextualization is unrestricted, while the cached identity reference e_i^{id} remains separate for routing. Attention keys and values are computed from the full contextualized token.

Relation and prospective object tokens. Beyond individual object tokens, ObjDream introduces a relation token and $N_p = 2$ prospective object tokens:

$$h_t^{\text{rel}} = \Phi_{\text{rel}}\Big([\text{Pool}_i(h_{i,t}^{\text{obj}}); \text{Pool}_{i,j}(\Delta g_{ij,t})]\Big), \qquad h_{k,t}^{\text{pro}} = \Phi_{\text{pro}}\Big([h_{i_k,t}^{\text{obj}}; h_t^{\text{rel}}]\Big). \tag{6}$$

Here, Φ_{rel} aggregates object features and pairwise geometry into a compact global relational context, while Φ_{pro} combines this context with individual object features. Pool denotes aggregation over valid slots or valid slot pairs, and $\Delta g_{ij,t}$ encodes the relative geometry between slots i and j. The indices i_k correspond to the top-N_p valid slots ranked by a learned slot importance gate. These prospective tokens are optimized through the action objective to provide object-specific context for action generation. Together, the tokens form the object block

$$X_t^{\text{obj}} = [h_{1,t}^{\text{obj}}, \ldots, h_{N_s,t}^{\text{obj}}, h_t^{\text{rel}}, h_{1,t}^{\text{pro}}, \ldots, h_{N_p,t}^{\text{pro}}]. \tag{7}$$

The block is inserted between the multimodal prefix and the action-generation prompt. The attention mask retains causal attention and additionally permits bidirectional attention within the object block, excluding invalid and padded slots as keys.

Identity-guided object routing. For the base object token associated with slot i at VLM position j_i, query position p, layer ℓ, and attention head m, let $\bar{h}_p^\ell$ denote the normalized query hidden state. Using the cached identity reference e_i^{id}, we compute an identity matching score b_{pi}^ℓ and a head-specific gate $\eta_p^{\ell,m}$:

$$b_{pi}^\ell = \cos\big(U^\ell \bar{h}_p^\ell, A^\ell e_i^{\mathrm{id}}\big), \qquad \eta_p^{\ell,m} = \eta_{\max}\sigma\big((w_g^{\ell,m})^\top \bar{h}_p^\ell + c_g^{\ell,m}\big). \tag{8}$$

Here, U^ℓ and A^ℓ are learnable projections shared across heads, and $\cos(\cdot,\cdot)$ denotes cosine similarity between their ℓ_2-normalized outputs. The parameters $w_g^{\ell,m}$ and $c_g^{\ell,m}$ define the head-specific gate $\eta_p^{\ell,m} \in (0, \eta_{\max})$. For each valid base slot ($\nu_{i,t} = 1$), the backbone attention logit is augmented as

$$s_{pj_i}^{\ell,m} = \frac{(q_p^{\ell,m})^\top k_{j_i}^{\ell,m}}{\sqrt{d_k}} + \eta_p^{\ell,m} b_{pi}^\ell + M_{pj_i}, \tag{9}$$

where $q_p^{\ell,m}$ and $k_{j_i}^{\ell,m}$ are the standard backbone query and key, d_k is the key dimension, and M is the attention mask. The cached identity affects only the additive bias; standard keys and values are computed from the full contextualized tokens. All other logits remain unchanged and are normalized by the same softmax.

3.3 Object-Level Forward Dynamics Module

Action-conditioned object forward dynamics. Let $o_{i,t}$ denote the final VLM hidden state of the object token associated with slot i. An auxiliary head predicts an H_a-step action chunk $\tilde{A}_t$ from mean-pooled, stop-gradient features of valid slots, without using the action expert's output. We construct a position-aware action embedding e_t^a and action conditioning gates $\beta_{i,t}$:

$$e_t^a = \frac{1}{H_a}\sum_{h=1}^{H_a} P_a([\mathrm{sg}(\tilde{A}_{t,h}); p_h]), \qquad \beta_{i,t} = \sigma(P_\beta(\mathrm{sg}(o_{i,t}))). \tag{10}$$

Here, p_h is a learned temporal embedding, P_a and P_β are trainable networks, and sg denotes stop-gradient. The action conditioning gate $\beta_{i,t}$ is distinct from the slot importance gate. During warmup, demonstrated actions $A_t^\star$ replace $\tilde{A}_t$ in the conditioning path. For valid slots, current states are decoded directly, while future prediction is restricted to non-robot object slots:

$$\hat{Y}_{i,t}^{\mathrm{cur}} = D_{\mathrm{cur}}(o_{i,t}), \qquad \hat{Y}_{i,t+1:t+H_f}^{\mathrm{fut}} = D_{\mathrm{fut}}(o_{i,t} + \gamma\beta_{i,t}e_t^a). \tag{11}$$

Here, D_{fut} predicts all H_f future horizons in parallel, producing horizon-indexed geometry, visibility, and mask logits; robot slots are excluded since their motion is represented by the action chunk. The auxiliary branch is skipped at inference, when the policy directly generates action chunks.

3.4 Training Objectives

Object encoder pretraining. We pretrain E_ω using structured object-level supervision:

$$\mathcal{L}_{\mathrm{enc}} = \lambda_{\mathrm{mask}}\mathcal{L}_{\mathrm{mask}} + \lambda_{\mathrm{geo}}\mathcal{L}_{\mathrm{geo}} + \lambda_{\mathrm{vis}}\mathcal{L}_{\mathrm{vis}} + \lambda_{\mathrm{rt}}\mathcal{L}_{\mathrm{rt}}. \tag{12}$$

Here, $\mathcal{L}_{\mathrm{mask}}$ combines binary cross-entropy and Dice losses for object segmentation. $\mathcal{L}_{\mathrm{geo}}$ supervises object geometry, including boxes, centers, areas, mask–geometry consistency, and auxiliary slot attributes. $\mathcal{L}_{\mathrm{vis}}$ supervises visibility labels derived from offline annotations; visibility is distinct from slot validity. $\mathcal{L}_{\mathrm{rt}}$ is a bidirectional, multi-positive region–text contrastive loss between projected regional visual features and frozen CLIP text features.

VLA training. During VLA training, we keep the pretrained object encoder E_ω frozen and optimize the remaining trainable components with action imitation and auxiliary losses:

$$\mathcal{L} = w_{\mathrm{act}}\mathcal{L}_{\mathrm{act}} + \mathcal{L}_{\mathrm{aux}}, \qquad \mathcal{L}_{\mathrm{aux}} = \mathcal{L}_{\mathrm{cur}} + \mathcal{L}_{\mathrm{fut}} + \mathcal{L}_{\mathrm{prop}} + \mathcal{L}_{\mathrm{gate}}. \tag{13}$$

Here, $\mathcal{L}_{\mathrm{act}}$ is the flow-matching action loss, which also optimizes the prospective object tokens through the action objective. $\mathcal{L}_{\mathrm{cur}}$ supervises current object geometry, visibility, and mask logits, while $\mathcal{L}_{\mathrm{fut}}$ supervises future object states predicted by the forward dynamics module. $\mathcal{L}_{\mathrm{prop}}$ trains the auxiliary coarse action proposal against the demonstrated action chunk, while $\mathcal{L}_{\mathrm{gate}}$ regularizes the

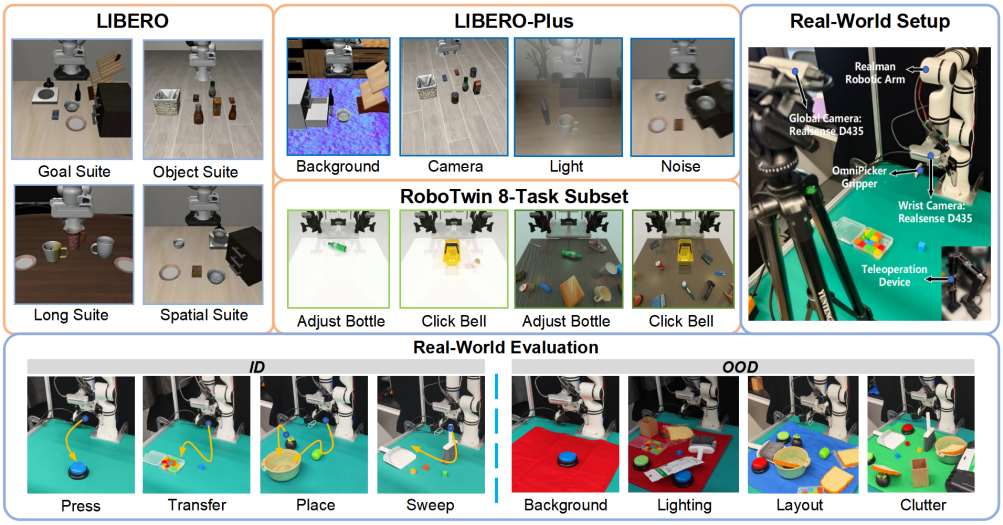


Figure 3: **Experimental overview.** We evaluate ObjDream on LIBERO, LIBERO-Plus, a fixed 8-task subset of RoboTwin 2.0, and four real-world manipulation tasks, covering standard and perturbed evaluation settings.

learned slot-importance gates used for object-slot prioritization. The routing projections and gates are optimized through the existing objectives without an additional routing loss. $\mathcal{L}_{\text{fut}}$ is computed only for valid non-robot object slots, excluding robot, invalid, and padded slots. Additional implementation details, training objectives, and analyses of the object encoder, identity-guided routing, and future object state prediction are provided in Appendices A–C.

4 Experiments

Benchmarks. We evaluate ObjDream on LIBERO, LIBERO-Plus, and RoboTwin 2.0, covering both single-arm and bimanual manipulation under diverse distribution shifts. For LIBERO, we follow the standard protocol over the Spatial, Object, Goal, and Long suites (Liu et al., 2023). We further evaluate zero-shot robustness on LIBERO-Plus under seven controlled shifts in camera viewpoint, robot initialization, object layout, language, lighting, background, and sensor noise (Fei et al., 2026). For RoboTwin 2.0, we evaluate on a fixed 8-task subset: *adjust bottle*, *click bell*, *grab roller*, *move can pot*, *open microwave*, *place can basket*, *scan object*, and *turn switch*. These tasks cover diverse manipulation behaviors and object-state transitions. We report results under both Easy and Hard settings (Chen et al., 2025b).

Implementation, training, and evaluation protocol. We use a Qwen3-VL-4B (Bai et al., 2025) backbone and a layerwise flow-matching action expert, following the StarVLA training recipe (Community, 2026). Training proceeds in three stages: (1) we pretrain the spatio-temporal object encoder using offline object-level supervision; (2) with the encoder frozen, we jointly optimize the VLM, action expert, and policy-side object modules using Eq. 13; and (3) we freeze the VLM and object-related modules and refine only the action expert with $\mathcal{L}_{\text{act}}$. Training is conducted on 8 NVIDIA H100 GPUs, while policy inference during evaluation is performed on NVIDIA RTX 4090 GPUs. Each LIBERO task is evaluated over 50 episodes, with the same checkpoint evaluated on LIBERO-Plus without additional fine-tuning. For RoboTwin 2.0, the policy is trained on 50 clean demonstrations per task and evaluated over 100 rollouts per task under both Easy and Hard settings. Further evaluation details are provided in Appendices D and E.

4.1 Main Results

LIBERO and LIBERO-Plus. Standard LIBERO measures nominal manipulation performance, while LIBERO-Plus evaluates the same LIBERO-trained policy under controlled distribution shifts without additional fine-tuning. As shown in Table 1, ObjDream achieves the highest average success

Table 1: **Performance on standard LIBERO and zero-shot robustness on LIBERO-Plus.** All LIBERO-Plus results are evaluated without fine-tuning on perturbed environments. Best results are shown in bold, and second-best results are underlined.

Method	LIBERO					LIBERO-Plus							
	Spatial	Object	Goal	Long	Avg	Cam.	Robot	Layout	Lang.	Light	BG	Noise	Overall
General VLA models													
OpenVLA (Kim et al., 2024)	84.7	88.4	79.2	53.7	76.5	0.8	3.5	28.5	23.0	8.1	34.8	15.2	15.6
OpenVLA-OFT (Kim et al., 2025)	97.6	98.4	97.9	94.5	97.1	56.4	31.9	74.2	79.5	88.7	93.3	75.8	69.6
UniVLA (Bu et al., 2025)	96.5	96.8	95.6	92.0	95.2	1.8	46.2	31.9	69.6	69.0	81.0	21.2	42.9
π_0 (Black et al., 2024)	96.8	98.8	95.8	85.2	94.2	13.8	6.0	68.9	58.8	85.0	81.4	79.0	53.6
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	98.8	98.2	98.0	92.4	96.9	75.4	77.5	**85.7**	85.6	96.9	94.6	**89.7**	**85.7**
AVA-VLA (Xiao et al., 2026)	97.4	99.4	97.4	**97.6**	98.0	55.5	25.9	74.1	85.6	95.5	88.9	78.0	70.1
X-VLA (Zheng et al., 2026)	98.2	98.6	97.8	**97.6**	98.1	23.4	**89.7**	71.8	75.7	88.2	96.0	62.7	71.4
Object-centric and predictive models													
WorldVLA (Cen et al., 2025b)	87.6	96.2	83.4	60.0	81.8	0.1	27.9	38.0	41.6	43.7	17.1	10.9	25.0
VLA-JEPA (Sun et al., 2026)	96.2	**99.6**	97.2	95.8	97.2	63.3	67.1	85.1	85.4	95.6	93.6	66.3	79.5
OA-WAM (Liu et al., 2026)	98.9	99.0	97.4	95.9	97.8	**80.5**	89.6	82.8	85.3	96.5	95.9	75.6	83.9
ObjDream (Ours)	**99.2**	**99.6**	**98.2**	97.4	**98.6**	66.2	75.8	81.2	**90.6**	**98.4**	**98.5**	88.6	84.4

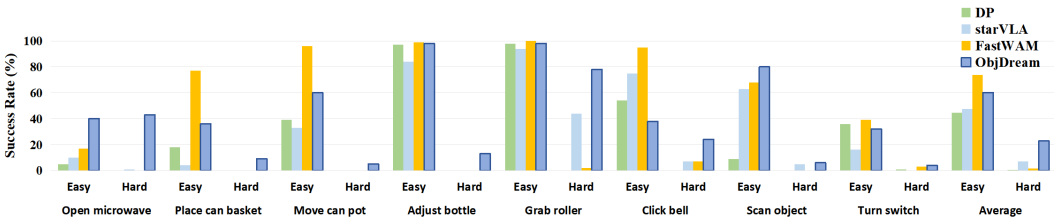


Figure 4: Success rates on the fixed 8-task RoboTwin 2.0 subset under Easy and Hard settings.

rate on LIBERO (98.6%) and ranks second overall on LIBERO-Plus (84.4%), with the best results under language (90.6%), lighting (98.4%), and background (98.5%) perturbations and second-best performance under sensor noise (88.6%). These results indicate strong robustness to semantic and appearance variations, while camera viewpoint and robot initialization remain the main challenges.

RoboTwin 2.0. As shown in Fig. 4, we evaluate ObjDream on the fixed eight-task RoboTwin 2.0 subset under Easy and Hard settings. ObjDream achieves average success rates of 60.25% on Easy and 22.75% on Hard, demonstrating competitive robustness under randomized scene variations without large-scale pretraining on robot data. Compared with StarVLA, ObjDream improves the average success rate by 12.87 percentage points on Easy and 15.62 percentage points on Hard, with a larger gain under the more challenging Hard setting. The improvement is particularly evident on *grab roller*, *open microwave*, and *click bell* under Hard conditions. These results indicate improved robustness to substantial distribution shifts in bimanual manipulation.

4.2 Ablation and Analysis

We conduct ablations on LIBERO-Long and LIBERO-Plus to assess the object interface, object relation modeling (ObjRel), object dynamics (ObjDyn), factorized object injection (Fact. Injection), and identity-guided routing (Id. Routing). Table 2 includes a backbone baseline without the object interface, while "w/o ObjRel & ObjDyn" retains the object tokens, factorized injection, identity-guided routing, and current state supervision.

The full model outperforms the backbone baseline by 2.0 and 3.2 percentage points on LIBERO-Long and LIBERO-Plus, respectively. The object interface without ObjRel or ObjDyn provides a modest improvement over the backbone on LIBERO-Plus and comparable performance on LIBERO-Long. Among single-component removals, disabling factorized injection or identity-guided routing produces the largest decreases in LIBERO-Plus performance, from 84.4% to 82.5% and 82.6%, respectively. Removing ObjDyn reduces the overall success rate to 83.2%, while removing ObjRel results in a smaller decrease to 84.1%. Overall, factorized injection, identity-guided routing, and future object

Table 2: **Ablation study of object-centric components.** Success rates (%) on LIBERO-Long and LIBERO-Plus. Overall is the trial-weighted LIBERO-Plus average. Best results are shown in bold, and second-best results are underlined.

Variant	LIBERO	LIBERO-Plus							
	Long	Cam.	Robot	Layout	Lang.	Light	BG	Noise	Overall
Backbone baseline	95.4	66.4	74.7	80.0	87.1	91.7	92.4	82.8	81.2
w/o ObjRel & ObjDyn	95.2	65.6	73.7	78.9	87.4	95.6	96.9	85.0	82.1
w/o ObjDyn	96.2	66.6	75.0	80.6	88.2	96.2	96.4	87.3	83.2
w/o ObjRel	96.6	**67.4**	**76.1**	80.9	88.2	97.6	97.8	**88.9**	84.1
w/o Fact. Injection	96.2	65.9	75.3	81.0	89.0	94.4	93.5	85.5	82.5
w/o Id. Routing	95.8	65.3	74.9	80.5	87.2	95.3	96.4	86.6	82.6
ObjDream (Full)	**97.4**	66.2	75.8	**81.2**	**90.6**	**98.4**	**98.5**	88.6	**84.4**

Table 3: **Real-robot manipulation results.** Success rates (%) on four real-world tasks under ID and OOD evaluation. Best and second-best results within each setting are bold and underlined.

Eval.	Method	Button Pressing	Block Pick-and-Place	Two-Fruit Placement	Tabletop Sweeping	Avg. Success
ID	$\pi_{0.5}$ (Physical Intelligence et al., 2025)	**90**	**70**	40	65	66
	ControlVLA (Li et al., 2025a)	65	40	**70**	65	60
	ObjDream (Ours)	80	65	60	**90**	**74**
OOD	$\pi_{0.5}$ (Physical Intelligence et al., 2025)	**65**	**60**	40	45	53
	ControlVLA (Li et al., 2025a)	40	10	35	40	31
	ObjDream (Ours)	**65**	50	**50**	**55**	**55**

state supervision provide complementary gains, while Appendix C further shows that future object prediction captures nontrivial state changes and benefits from action conditioning.

4.3 Real-World Experiments

Real-world experiments. We evaluate ObjDream on four real-world tasks: *button pressing*, *block pick-and-place*, *two-fruit placement*, and *tabletop sweeping*, covering contact interaction, object manipulation, and long-horizon control. We compare against $\pi_{0.5}$ (Physical Intelligence et al., 2025) and ControlVLA (Li et al., 2025a). All methods use the same task demonstrations for adaptation and are evaluated under in-distribution (ID) and out-of-distribution (OOD) settings without additional fine-tuning.

Results. As shown in Table 3, ObjDream achieves average success rates of 74% and 55% under ID and OOD conditions, respectively, exceeding the strong $\pi_{0.5}$ baseline by 8 percentage points under ID while achieving comparable aggregate performance under OOD. Compared with ControlVLA, ObjDream improves average success rates by 14 and 24 percentage points under ID and OOD conditions, respectively. Notably, under OOD conditions, ObjDream outperforms the best baseline by 10 percentage points on both *two-fruit placement* and *tabletop sweeping*, two tasks that require precise manipulation. These results demonstrate competitive generalization in real-world manipulation without large-scale pretraining on robot data.

5 Conclusion

We present ObjDream, an object-centric VLA framework combining structured object representations with action-conditioned future object state supervision. ObjDream represents objects with persistent identity cues and time-varying state features, and leverages identity-guided routing to direct VLM attention toward task-relevant object tokens. It further supervises object state evolution during training without requiring the decoding of future object states at inference. Experiments on LIBERO, LIBERO-Plus, a fixed 8-task RoboTwin 2.0 subset, and four real-world tasks demonstrate strong manipulation performance and robustness to scene variations.

AI USE STATEMENT

Generative AI tools were used for language polishing and editorial refinement of the manuscript, including improving its clarity, grammar, and presentation. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the final content of the paper.

REFERENCES

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Yongjie Bai, Hanting Wang, Mingtong Dai, Qijun Zhong, Yang Liu, and Liang Lin. Bridge-wa: Predicting where and how the world changes for robotic action. *arXiv preprint arXiv:2607.02195*, 2026.

Rokas Bendikas, Daniel Dijkman, Markus Peschl, Sanjay Haresh, and Pietro Mazzaglia. Focusing on What Matters: Object-Agent-Centric Tokenization for Vision Language Action Models. In *Proceedings of The 9th Conference on Robot Learning*, volume 305, pp. 3869–3887, 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.

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

Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, et al. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026.

Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Bohan Hou, Yuming Jiang, Jiayan Guo, Xin Li, et al. Rynnvla-002: A unified vision-language-action and world model. *arXiv preprint arXiv:2511.17502*, 2025a.

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*, 2025b.

Haonan Chen, Jingxiang Guo, Bangjun Wang, Tianrui Zhang, Xuchuan Huang, Boren Zheng, Yiwen Hou, Chenrui Tie, Jiajun Deng, and Lin Shao. Goal-vla: Image-generative vlms as object-centric world models empowering zero-shot robot manipulation. *arXiv preprint arXiv:2506.23919*, 2025a.

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*, 2025b.

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.

Nhat Chung, Taisei Hanyu, Toan Nguyen, Huy Le, Frederick Bumgarner, Duy Minh Ho Nguyen, Khoa Vo, Kashu Yamazaki, Chase Rainwater, Tung Kieu, Anh Nguyen, and Ngan Le. Rethinking progression of memory state in robotic manipulation: An object-centric perspective. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(5):3407–3415, 2026. doi: 10.1609/aaai.v40i5.37337.

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

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

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 (CVPR)*, pp. 38574–38583, June 2026.

Stefano Ferraro, Pietro Mazzaglia, Tim Verbelen, and Bart Dhoedt. Focus: object-centric world models for robotic manipulation. *Frontiers in Neurorobotics*, 19:1585386, 2025.

Xinyu Guo, Bin Xie, Wei Chai, Xianchi Deng, Tiancai Wang, Zhengxing Wu, and Xingyu Chen. Priorvla: Prior-preserving adaptation for vision-language-action models. *arXiv preprint arXiv:2605.10925*, 2026.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Asher Hancock, Xindi Wu, Lihan Zha, Olga Russakovsky, and Anirudha Majumdar. Actions as language: Fine-tuning vlms into vlas without catastrophic forgetting. In *International Conference on Learning Representations*, volume 2026, pp. 75333–75360, 2026.

Taisei Hanyu, Nhat Chung, Huy Le, Toan Nguyen, Yuki Ikebe, Anthony Gunderman, Duy Nguyen Ho Minh, Khoa Vo, Tung Kieu, Kashu Yamazaki, et al. Slotvla: Towards modeling of object-relation representations in robotic manipulation. *arXiv preprint arXiv:2511.06754*, 2025.

Zhi Hou, Tianyi Zhang, Yuwen Xiong, Haonan Duan, Hengjun Pu, Ronglei Tong, Chengyang Zhao, Xizhou Zhu, Yu Qiao, Jifeng Dai, et al. Dita: Scaling diffusion transformer for generalist vision-language-action policy. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7686–7697. IEEE, 2025.

Youngjoon Jeong, Junha Chun, Soonwoo Cha, and Taesup Kim. Object-centric world model for language-guided manipulation. *arXiv preprint arXiv:2503.06170*, 2025.

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.

Thomas Kipf, Gamaleldin F Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. *arXiv preprint arXiv:2111.12594*, 2021.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 132324–132345, 2026a.

Puhao Li, Yingying Wu, Ziheng Xi, Wanlin Li, Yuzhe Huang, Zhiyuan Zhang, Yinghan Chen, Jianan Wang, Song-Chun Zhu, Tengyu Liu, and Siyuan Huang. ControlVLA: Few-Shot Object-Centric Adaptation for Pre-Trained Vision-Language-Action Models. In *Proceedings of The 9th Conference on Robot Learning*, volume 305, pp. 1898–1913, 2025a.

Wei Li, Peijin Jia, Yuan Ma, Xuefeng Jiang, Titong Jiang, Sheng Sun, Yujian Li, Xin Wen, Han Hong, Zhikang Liu, et al. Fomovla: Bridging visual foresight and motion guidance for vision-language-action models. *arXiv preprint arXiv:2607.14739*, 2026b.

Xiang Li, Yupeng Zheng, Songen Gu, Huailiang Ma, Feng Yu, Xian Nie, Shanshuai Yuan, Yujie Zang, Weize Li, Shuai Tian, et al. Latent action as intention enables efficient future imagination for world action models. *arXiv preprint arXiv:2608.24882*, 2026c.

Xiaoqi Li, Jingyun Xu, Mingxu Zhang, Jiaming Liu, Yan Shen, Iaroslav Ponomarenko, Jiahui Xu, Liang Heng, Siyuan Huang, Shanghang Zhang, et al. Object-centric prompt-driven vision-language-action model for robotic manipulation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27638–27648. IEEE, 2025b.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, et al. Hamster: Hierarchical action models for open-world robot manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 24040–24068, 2025c.

Wenqi Liang, Gan Sun, Yao He, Jiahua Dong, Suyan Dai, Ivan Laptev, Salman Khan, and Yang Cong. Pixelvla: Advancing pixel-level understanding in vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 110464–110481, 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.

Yushan Liu, Peibo Sun, Shoujie Li, Yifan Xie, Lingfeng Zhang, Xintao Chao, Shiyuan Dong, Fang Chen, Xiao-Ping Zhang, and Wenbo Ding. Oa-wam: Object-addressable world action model for robust robot manipulation. *arXiv preprint arXiv:2605.06481*, 2026.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Malte Mosbach, Jan Niklas Ewertz, Angel Villar-Corrales, and Sven Behnke. Sold: Slot object-centric latent dynamics models for relational manipulation learning from pixels. *arXiv preprint arXiv:2410.08822*, 2024.

NVIDIA, Johan Bjorck, Fernando Castaneda, 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.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 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.

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.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 8748–8763. PMLR, 2021.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, volume 2025, pp. 28085–28128, 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. *arXiv preprint arXiv:2602.10098*, 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.

Xiaomi Robotics Team, Jun Guo, Piaopiao Jin, Jason Li, Peiyan Li, Yingyan Li, Futeng Liu, Wanli Peng, Optimus Qin, Yifei Su, et al. Xiaomi-robotics-1: Scaling vision-language-action models with over 100k hours of real-world trajectories. *arXiv preprint arXiv:2607.15330*, 2026.

Sieu Tran, Duc Nguyen, Hao Vo, Khoa Vo, and Ngan Le. Dual-state slot attention: Decoupling appearance and identity for video object-centric learning. *arXiv preprint arXiv:2606.12601*, 2026.

Angel Villar-Corrales and Sven Behnke. Playslot: Learning inverse latent dynamics for controllable object-centric video prediction and planning. *arXiv preprint arXiv:2502.07600*, 2025.

Khoa Vo, Taisei Hanyu, Yuki Ikebe, Trong Thang Pham, Nhat Chung, Minh Nhat Vu, Duy Nguyen Ho Minh, Anh Nguyen, Anthony Gunderman, Chase Rainwater, et al. Clutter-robust vision-language-action models through object-centric and geometry grounding. *arXiv preprint arXiv:2512.22519*, 2025.

Kuanning Wang, Ke Fan, Chenhao Qiu, Zeyu Shangguan, Yuqian Fu, Yanwei Fu, Daniel Seita, and Xiangyang Xue. Oflow: Injecting object-aware temporal flow matching for robust robotic manipulation. *arXiv preprint arXiv:2604.17876*, 2026a.

Qiuyue Wang, Mingsheng Li, Jian Guan, Jinhui Ye, Sicheng Xie, Yitao Liu, Junhao Chen, Zhixuan Liang, Jie Zhang, Xintong Hu, et al. Qwen-vla: Unifying vision-language-action modeling across tasks, environments, and robot embodiments. *arXiv preprint arXiv:2605.30280*, 2026b.

Xinkai Wang, Chenyi Wang, Yifu Xu, Mingzhe Ye, Fu-Cheng Zhang, Jialin Tian, Xinyu Zhan, Lifeng Zhu, Cewu Lu, and Lixin Yang. Lamp: Learning vision-language-action policies with 3d scene flow as latent motion prior. *arXiv preprint arXiv:2603.25399*, 2026c.

Jiude Wei, Yuxuan Li, Cewu Lu, and Jianhua Sun. Physically ground commonsense knowledge for articulated object manipulation with analytic concepts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13519–13528, 2026.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pp. 2226–2240. PMLR, 2023.

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*, 2026a.

Wei Wu, Fangjing Wang, Fan Lu, He Sun, Shi Liu, Yunnan Wang, Yibin Yan, Yong Wang, Shuailei Ma, Xinyang Wang, et al. From foundation to application: Improving vla models in practice. *arXiv preprint arXiv:2607.06403*, 2026b.

Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. *arXiv preprint arXiv:2210.05861*, 2022.

Lei Xiao, Jifeng Li, Juntao Gao, Feiyang Ye, Yan Jin, Jingjing Qian, Jing Zhang, Yong Wu, and Xiaoyuan Yu. Ava-vla: Improving vision-language-action models with active visual attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13453–13463, 2026.

Yue Yang, Shuo Cheng, Yu Fang, Homanga Bharadhwaj, Mingyu Ding, Gedas Bertasius, and Daniel Szafir. Lilo-vla: Compositional long-horizon manipulation via linked object-centric policies. *arXiv preprint arXiv:2602.21531*, 2026.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Hanyang Yu, Haitao Lin, Jingbo Zhang, Wenyao Zhang, Chenghao Gu, Heng Li, and Ping Tan. Maskwam: Unifying mask prompting and prediction for world-action models. *arXiv preprint arXiv:2606.13515*, 2026a.

Qize Yu, Jiadi You, Yuran Wang, Jiaqi Liang, Bowen Ping, Yang Tian, Yue Chen, Minghong Cai, Zeying Gong, Ruihai Wu, et al. Affordancevla: A vision-language-action model empowering action generation through affordance-aware understanding. *arXiv preprint arXiv:2606.06155*, 2026b.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

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, pp. 24195–24228, 2025.

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.

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.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024a.

Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. Robodreamer: Learning compositional world models for robot imagination. *arXiv preprint arXiv:2404.12377*, 2024b.

Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. In *Conference on Robot Learning*, pp. 1199–1210. PMLR, 2023.

CONTENTS OF THE APPENDIX

A Object Encoder Pretraining and Analysis 15
A.1 Offline Annotation and Pretraining Data 15
A.2 Object Encoder Architecture and Pretraining 17
A.3 Object Encoder Evaluation in Simulation 19

B VLA Integration and Training Details 20
B.1 Object Representation Integration 20
B.2 Training Objectives 23
B.3 Training and Inference Configuration 24

C Future Object State Prediction Analysis 27

D Simulation Evaluation Protocols 29
D.1 LIBERO and LIBERO-Plus 29
D.2 RoboTwin 2.0 Evaluation 30

E Real-World Experimental Setup and Evaluation 31
E.1 Robot Platform and Policy Interface 31
E.2 Demonstration Data and Training 33
E.3 Evaluation Protocol 34
E.4 Real-World Result Analysis 34

F Limitations and Broader Impact 35

G Qualitative Rollout Visualizations 35

H Appendix References 42

A OBJECT ENCODER PRETRAINING AND ANALYSIS

A.1 OFFLINE ANNOTATION AND PRETRAINING DATA

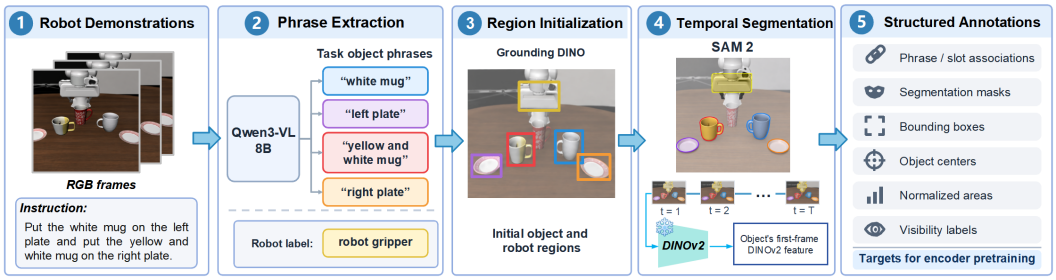


Figure 5: **Automatic offline annotation pipeline for object encoder pretraining.** Qwen3-VL-8B extracts task-relevant object phrases from demonstration instructions and initial observations, while a separate *robot gripper* label is used for robot annotation. Grounding DINO initializes the corresponding object regions, SAM2 propagates segmentation masks across frames, and DINOv2 extracts object-specific features from the initial grounded regions. Masks, geometry, visibility, and phrase–region correspondences are automatically constructed from this pipeline without manual frame-wise object annotation. The resulting annotations supervise encoder pretraining, while the DINOv2 features provide visual identity references.

The spatio-temporal object encoder is pretrained using the robot demonstrations summarized in Table 4. Importantly, the object-level supervision is automatically generated from the demonstration trajectories and does not require manual frame-wise annotation of masks, boxes, geometry, or visibility. For each trajectory with instruction x and visual observations, we construct object-level supervision through phrase extraction, grounded localization, mask propagation, and automatic object-state annotation, yielding temporally aligned masks, geometry, visibility, and phrase–region correspondences. The offline annotation pipeline is illustrated in Fig. 5.

Table 4: **Data used for object encoder pretraining.**

Evaluation setting	Pretraining data	Tasks	Demos / task	Total demos
LIBERO	LIBERO	40	50	2,000
LIBERO-Plus	LIBERO	40	50	2,000
RoboTwin 2.0 (8-task subset)	RoboTwin 2.0	8	50	400
Real-world experiments	Real-world demonstrations	4	100	400

LIBERO-Plus uses the same encoder pretraining data as LIBERO. For RoboTwin 2.0, encoder pretraining uses 50 clean demonstrations per task.

Instruction-conditioned object phrases. During automatic offline annotation, Qwen3-VL-8B (Bai et al., 2025) takes the language instruction x together with the initial visual observation I_0 and extracts the task-relevant phrase set P_0. The extracted phrases are fixed for the entire trajectory and used as consistent semantic references across frames. During policy rollout, the same parser is invoked once at episode initialization to construct P_0, which is cached for all subsequent control steps. When multiple instances share the same category, coarse spatial cues, such as relative left/right positions, are included when needed to distinguish them. During offline annotation, the label *robot gripper* is used only to localize robot regions and assign their robot role; it is not used to construct robot queries. Learnable robot queries retrieve robot visual features without a direct text term, as described in Sec. A.2. Unused slot positions are assigned the padding role and excluded through the validity mask.

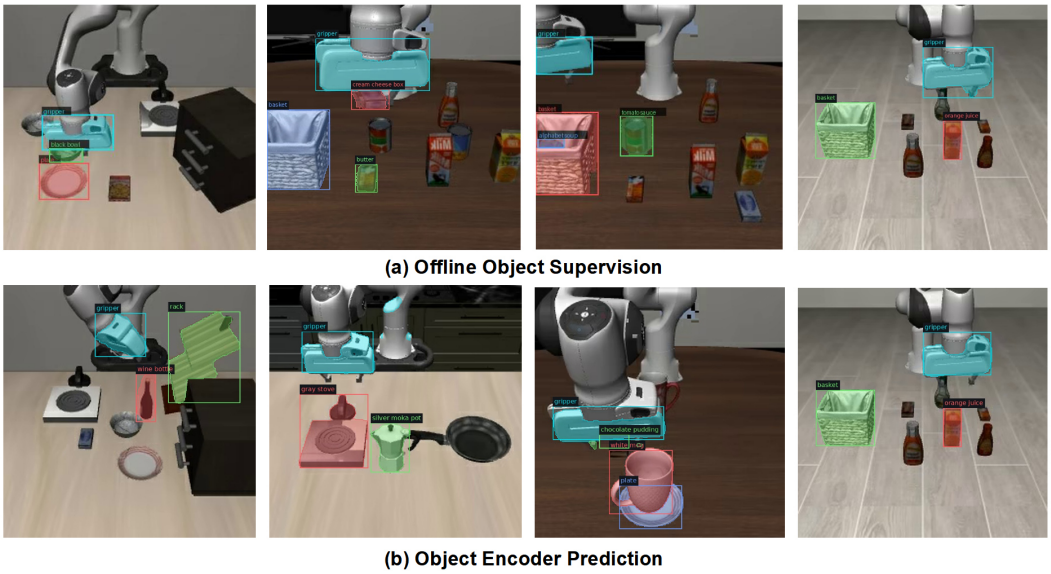


Figure 6: **Visualization of automatically generated object supervision and object encoder predictions. (a)** Automatically generated offline supervision associates task-relevant phrases with object regions and includes dedicated robot annotations. **(b)** Predictions from the pretrained object encoder conditioned on the cached phrase set P_0.

Phrase-grounded localization and temporal segmentation. Given P_0, Grounding DINO (Liu et al., 2024) provides initial phrase-conditioned localization, and SAM2 (Ravi et al., 2025) propagates the corresponding masks across the demonstration trajectory. The phrase associated with each object remains fixed while its spatial support changes with motion, occlusion, or visibility. Boxes, centers, areas, and visibility labels are automatically derived from the propagated masks and normalized with respect to the image dimensions where applicable. DINOv2 (Oquab et al., 2023) features are pooled over the initial grounded object region to construct the visual identity reference during encoder pretraining. Thus, the masks, geometry, and visibility labels serve as automatically generated offline supervision and are not provided as policy inputs during rollout. Grounding DINO localization and

Table 5: **Supervision for object encoder pretraining.** All targets are automatically constructed offline from the demonstration trajectories without manual frame-wise object annotation.

Objective	Offline target	Supervision
$\mathcal{L}_{\text{mask}}$	Binary masks of automatically localized object and robot regions	BCE + Dice segmentation loss
$\mathcal{L}_{\text{geo}}$	Boxes, centers, areas, and auxiliary slot attributes derived from the propagated regions	Geometry regression and mask–geometry consistency
$\mathcal{L}_{\text{vis}}$	Binary visibility labels automatically derived for each slot and frame	Per-slot visibility prediction
$\mathcal{L}_{\text{rt}}$	Aligned visual regions and frozen CLIP phrase features	Bidirectional multi-positive region–text contrastive learning

SAM2 video mask propagation are used only for offline annotation and are not executed during rollout.

Object-level supervision. The encoder is pretrained with four objectives using the automatically generated offline targets introduced above:

$$\mathcal{L}_{\text{enc}} = \lambda_{\text{mask}}\mathcal{L}_{\text{mask}} + \lambda_{\text{geo}}\mathcal{L}_{\text{geo}} + \lambda_{\text{vis}}\mathcal{L}_{\text{vis}} + \lambda_{\text{rt}}\mathcal{L}_{\text{rt}}. \tag{14}$$

$\mathcal{L}_{\text{mask}}$ combines binary cross-entropy (BCE) and Dice losses for object segmentation. $\mathcal{L}_{\text{geo}}$ groups box, center, and area supervision together with mask–geometry consistency and auxiliary slot attribute prediction. $\mathcal{L}_{\text{vis}}$ supervises binary visibility labels for each slot; visibility is distinct from slot validity. $\mathcal{L}_{\text{rt}}$ provides bidirectional multi-positive region–text contrastive supervision between projected visual regions and frozen CLIP phrase features. The DINOv2 reference is used as an identity feature and introduces no additional supervision term.

A.2 Object Encoder Architecture and Pretraining

Inputs during policy training and rollout. During policy training and rollout, the encoder receives the causal image window $\mathbf{I}_t$ from the designated external or third-person view and the cached phrase set P_0 constructed from the instruction x and initial observation I_0. A frozen SAM 2.1 Hiera-Tiny image backbone (Ravi et al., 2025) extracts per-frame visual features, while a frozen CLIP text encoder (Radford et al., 2021) provides phrase features for object queries. At episode initialization, a frozen DINOv2 encoder (Oquab et al., 2023) extracts a visual reference from each initial object region predicted by the object encoder, which is cached throughout the episode. Unlike offline SAM2 video mask propagation, the frozen SAM2 image backbone remains active at each rollout step, whereas DINOv2 is invoked only during episode initialization.

Query initialization. For each object phrase $p_i \in P_0$, the frozen CLIP text encoder produces a feature $f_i \in \mathbb{R}^{512}$. A learnable semantic projection produces the compact identity embedding

$$\iota_i = \text{Normalize}(P_{\text{sem}}(f_i)) \in \mathbb{R}^{32}. \tag{15}$$

The corresponding object query is initialized as

$$q_i^{(0)} = P_{\text{id}}(\iota_i) + P_{\text{text}}(f_i) + E_{\text{type}}(\chi_i), \tag{16}$$

where P_{id} maps the compact identity into the query space, P_{text} projects the full CLIP phrase feature, and E_{type} provides a learned query type prior. The compact identity branch provides a stable semantic reference, while the direct text branch preserves richer phrase semantics for visual grounding.

Features from the frozen DINOv2 encoder are pooled over the corresponding initial object region and projected into a compact visual reference $d_i \in \mathbb{R}^{32}$. During encoder pretraining, pooling uses the aligned annotated region in the initial encoder view; during policy training and rollout, it uses the region predicted by the phrase-conditioned object query. Because query initialization does not depend on d_i, the initial region can be predicted before constructing the visual reference. During policy training and rollout, d_i and ι_i are produced by the frozen encoder and cached for the episode. Robot queries instead use learned identity embeddings and robot type priors without the direct text term, with a learned reference vector occupying the visual reference channel.

Table 6: **Object encoder architecture and outputs.** T denotes the causal observation window length. Tensor shapes omit the batch dimension.

Component	Description	Output
Visual encoding	Frozen SAM 2.1 Hiera-Tiny backbone followed by learnable visual projection	Per-frame visual features
Phrase encoding	Frozen CLIP text encoder	Phrase feature $f_i \in \mathbb{R}^{512}$
Semantic identity	Learnable projection and normalization of f_i	$\iota_i \in \mathbb{R}^{32}$
DINOv2 visual reference	Pooling over initial object regions from frozen DINOv2 features followed by learnable projection	$d_i \in \mathbb{R}^{32}$
Object query initialization	Compact identity projection + full CLIP text projection + learned query type prior	Query in $\mathbb{R}^{336}$
Robot query initialization	Learned identity embedding + robot type prior	Query in $\mathbb{R}^{336}$
Spatial query module	Query self-attention, visual cross-attention, and feed-forward transformations	$T \times N_s \times 336$
Temporal encoder	Sinusoidal temporal encoding, causal self-attention, and gated residual refinement	$T \times N_s \times 336$
Region–text alignment	Mask-weighted region pooling followed by semantic projection to frozen CLIP space	Region embedding in $\mathbb{R}^{512}$
Identity component e_i^{id}	Semantic identity ι_i, visual reference d_i, and role embedding r_i	$N_s \times 80$
Dynamic state $e_{i,t}^{\mathrm{state}}$	Appearance, geometry, and temporal features	$T \times N_s \times 256$
Flattened slot feature $s_{i,t}$	Identity, appearance, geometry, temporal, and role components	$T \times N_s \times 336$
Validity mask $\nu_{i,t}$	Binary mask derived from phrase validity and query type	$T \times N_s$
Visibility prediction	Binary per-slot prediction supervised by offline visibility labels	$T \times N_s$

Spatial and temporal encoding. A learnable visual projection maps frozen SAM2 features into the encoder hidden space for each frame. The same initial query set is reused across the causal observation window. Within each frame, the spatial query module applies query self-attention, cross-attention to the projected visual features, and feed-forward transformations to obtain image-conditioned object features. For each fixed query index, the temporal encoder integrates the resulting feature sequence using sinusoidal temporal encodings, causal self-attention, and gated residual refinement. Temporal attention accesses only the current and preceding observations, while the gated residual combines temporal information with the per-frame object features. The fixed query index associates each temporal feature sequence with its cached phrase and identity reference.

For LIBERO, we use $N_s = 5$ slots, including one robot slot and up to four phrase-conditioned object slots. For the fixed RoboTwin 2.0 subset, we use $N_s = 6$ slots, including one robot slot per arm and up to four object slots. Unused positions are assigned to padding and excluded through the validity mask.

Factorized slot readout. Following the main text, each output slot is $u_{i,t}^{\mathrm{obj}} = (e_i^{\mathrm{id}}, e_{i,t}^{\mathrm{state}}, \nu_{i,t})$. Its flattened feature vector is

$$s_{i,t} = [\iota_i; d_i; c_{i,t}^{\mathrm{app}}; g_{i,t}^{\mathrm{geo}}; \tau_t; r_i] \in \mathbb{R}^{336}, \tag{17}$$

with

$$e_i^{\mathrm{id}} = [\iota_i; d_i; r_i] \in \mathbb{R}^{80}, \qquad e_{i,t}^{\mathrm{state}} = [c_{i,t}^{\mathrm{app}}; g_{i,t}^{\mathrm{geo}}; \tau_t] \in \mathbb{R}^{256}. \tag{18}$$

Here, ι_i, d_i, $c_{i,t}^{\mathrm{app}}$, $g_{i,t}^{\mathrm{geo}}$, τ_t, and r_i have dimensions 32, 32, 224, 16, 16, and 16, respectively. Persistent object identity and dynamic object state are represented through separate channels: e_i^{id} remains fixed across the episode, whereas $e_{i,t}^{\mathrm{state}}$ captures time-varying appearance, geometry, and temporal information. The validity mask $\nu_{i,t}$ is stored separately from the slot feature and is distinct from predicted object visibility.

Interface to identity-guided object routing. At control step t, the encoder supplies the object slots $(e_i^{\mathrm{id}}, e_{i,t}^{\mathrm{state}}, \nu_{i,t})$. The policy separately projects identity and dynamic state into object tokens

Table 7: **Object encoder pretraining hyperparameters.** The encoder is pretrained for 80k steps and frozen during subsequent VLA training.

Hyperparameter	Value
Optimization	
Optimizer	AdamW
Training steps	80,000
Learning rate	$5 \times 10^{-5} \rightarrow 5 \times 10^{-6}$
LR schedule	1,000-step warmup + cosine decay
Weight decay	0.01
Global batch size	32 sequences (4/GPU $\times$ 8 GPUs)
Gradient clipping	1.0 (global norm)
Mixed precision	FP16 AMP
Random seed	42
Input configuration	
Temporal window	4 frames
Image / mask resolution	256×256 / 256×256
SAM input resolution	1024×1024
Maximum text length	48 tokens
DINOv2 reference dimension	32
Loss weights	
Mask / box	2.0 / 1.0
Auxiliary box supervision	0.25
Center / area	1.0 / 0.5
Mask-center / mask-area consistency	0.3 / 0.1
Region–text / visibility / auxiliary slot attributes	0.2 / 0.2 / 0.2

in the hidden space of the vision-language model (VLM) and inserts them into its token sequence. The identity reference e_i^{id} is cached separately to guide attention to the corresponding object token. All object token channels undergo contextual updates, and standard queries, keys, and values are computed from the full contextualized tokens. The cached reference supplies only an additive attention bias; identity channels are not reset after each VLM layer. The identity matching projections and routing gates are policy-side modules, as detailed in Sec. B.1.

Region–text alignment. To provide explicit visual-semantic supervision, region representations are extracted from the projected SAM2 feature map F_t before interaction with phrase-conditioned object queries. For region mask $m_{i,t}$, we compute

$$\rho_{i,t} = \mathrm{Normalize}(P_{\mathrm{reg}}(\mathrm{Pool}(F_t, \mathrm{sg}(m_{i,t})))) \,, \tag{19}$$

where Pool denotes mask-weighted pooling, P_{reg} maps the pooled visual feature into the frozen CLIP text feature space, and sg denotes stop-gradient. During encoder pretraining, aligned teacher masks are used for pooling; when evaluating region–text matching with the pretrained encoder, predicted masks are used instead. The bidirectional multi-positive contrastive objective treats all occurrences of the same normalized phrase within a batch as positives.

After pretraining, the complete object encoder is frozen during subsequent VLA optimization. During policy training and rollout, it aggregates visual evidence over the causal image window $\mathbf{I}_t$ conditioned on P_0 while preserving the cached identity references. For object slots, the DINOv2 visual reference d_i is constructed once from the initial region predicted by the encoder and reused throughout the episode.

A.3 Object Encoder Evaluation in Simulation

We evaluate the pretrained object encoder against offline annotations on the 40 LIBERO tasks and the eight RoboTwin tasks used in our benchmark. For RoboTwin 2.0, the encoder is pretrained using the clean demonstrations summarized in Table 4. The evaluation includes 203 LIBERO trajectories,

Table 8: **Object encoder performance in simulation.** Results are averaged across tasks using a causal window of $T = 4$. RoboTwin results are reported separately for clean and randomized evaluation trajectories. All values are percentages.

Dataset	Mask mIoU ↑	Box IoU ↑	Center error ↓	Visibility F1 ↑	Slot IDF1 ↑
LIBERO	95.31	88.35	0.42	98.49	95.92
RoboTwin (clean)	94.67	83.34	0.32	87.17	83.11
RoboTwin (randomized)	92.54	82.82	0.38	84.49	80.02

Table 9: **Effect of the temporal window length at inference.** All rows use the same encoder pretrained with $T = 4$; only the number of input frames changes. Results are averaged over the 40 LIBERO tasks and the eight RoboTwin tasks. Latency measures encoder forward passes after episode-level identity initialization on an NVIDIA RTX 4090 with batch size 1 and BF16 autocast, averaging 200 synchronized forward passes after 50 warm-up iterations. Both RoboTwin evaluation settings use the same pretrained encoder and latency measurement. All quality metrics are reported as percentages.

Dataset	T	Mask mIoU ↑	Box IoU ↑	Center error ↓	Visibility F1 ↑	Slot IDF1 ↑	Latency (ms) ↓
LIBERO	1	95.22	87.90	0.46	98.48	95.80	21.77
	2	95.23	88.37	0.43	98.49	95.90	27.46
	3	95.27	88.40	0.43	98.49	95.91	33.55
	4	95.31	88.35	0.42	98.49	95.92	40.29
RoboTwin (clean)	1	94.83	83.62	0.32	87.31	83.29	21.56
	2	94.86	83.67	0.31	87.35	83.18	27.35
	3	94.80	83.57	0.30	87.22	83.31	33.84
	4	94.67	83.34	0.32	87.17	83.11	39.88
RoboTwin (randomized)	1	91.59	80.93	0.41	83.50	80.11	21.56
	2	91.59	81.96	0.39	84.52	80.16	27.35
	3	92.58	82.82	0.38	84.50	80.07	33.84
	4	92.54	82.82	0.38	84.49	80.02	39.88

40 clean RoboTwin trajectories, and 151 randomized RoboTwin trajectories, all using a causal observation window of $T = 4$ frames. Metrics are first averaged within each trajectory and task and then averaged across tasks.

Table 8 reports mask mean intersection over union (mIoU), box intersection over union (IoU), center error, visibility F1, and Slot IDF1. Spatial metrics are computed on annotated visible objects with nonempty masks covering at least 0.3% of the image. Center error is the Euclidean distance between predicted and annotated centers, normalized by the image diagonal. Slot IDF1 (Ristani et al., 2016) measures temporal identity consistency by associating predicted slot trajectories with phrase-aligned annotated object trajectories using box-based matching. Table 9 further compares inference windows of one to four frames using the same encoder pretrained with $T = 4$. Longer temporal windows provide small gains on LIBERO, while RoboTwin shows limited sensitivity to the window length. These measurements evaluate the object encoder; the reported latency excludes VLM computation, identity-guided routing, and action generation.

B VLA Integration and Training Details

B.1 Object Representation Integration

We use a Qwen3-VL-4B backbone and a layerwise flow-matching action expert across simulation and real-world experiments. The pretrained object encoder provides factorized object slots, which are projected into object tokens and inserted into the VLM token sequence. Identity-guided routing uses

the cached identity references to guide attention to these tokens. Base object tokens correspond to the encoder slots, including robot slots. The resulting multimodal context is used by the action expert to predict continuous action chunks.

Factorized object injection. Following the main text, each object slot $u_{i,t}^{\mathrm{obj}}$ contains an identity component $e_i^{\mathrm{id}} = [\iota_i; d_i; r_i]$, a dynamic state component $e_{i,t}^{\mathrm{state}}$, and slot validity $\nu_{i,t}$. We separately project identity and state into complementary parts of the initial object token:

$$a_i = P_{\mathrm{id}}^{\mathrm{inj}}(e_i^{\mathrm{id}}), \qquad z_{i,t} = P_{\mathrm{obj}}\left([P_{\mathrm{state}}(e_{i,t}^{\mathrm{state}}); \phi_\nu(\nu_{i,t})]\right). \tag{20}$$

We then construct

$$h_{i,t}^{\mathrm{obj}} = [a_i; z_{i,t}]. \tag{21}$$

Here, $P_{\mathrm{id}}^{\mathrm{inj}}$, P_{state}, and P_{obj} are learnable projections, and ϕ_ν embeds slot validity. The concatenated token matches the VLM hidden dimension D. This factorization specifies the input representation; all token channels subsequently undergo contextual updates. Keys and values are computed from the full contextualized token, without resetting the identity channels after each layer. The original identity reference e_i^{id} is retained separately for the routing mechanism below and occupies no additional sequence position.

Relation and prospective object tokens. In addition to the base object tokens, we use one relation token and $N_p = 2$ prospective object tokens. Let $g_{i,t}^{\mathrm{slot}} \in (0, 1)$ denote the learned slot importance gate for slot i. The indices i_k correspond to the top-N_p valid slots ranked by $g_{i,t}^{\mathrm{slot}}$; predicted visibility is used as the fallback ranking score when the learned gate is unavailable. The k-th prospective token combines the object token associated with the selected slot and the shared relational context, following Φ_{rel} and Φ_{pro} in the main text. We fix $N_p = 2$ in all experiments to maintain a compact token budget.

Pairwise relation features. The pairwise geometry feature $\Delta g_{ij,t}$ is constructed from explicit spatial and slot-level attributes of two valid slots, including their relative center displacement, spatial distance, relative object extent, and pairwise visibility or importance cues. It is therefore distinct from a direct difference between the learned geometry embeddings $g_{i,t}^{\mathrm{geo}}$ and $g_{j,t}^{\mathrm{geo}}$. Self-pairs ($i = j$) are excluded. The resulting pairwise features are aggregated over valid slot pairs using an importance-weighted mean, where the weights are determined by the corresponding slot importance scores. Together with pooled object features, this aggregated pairwise representation is processed by Φ_{rel} to construct the relation token h_t^{rel}. The same slot importance scores are used to rank valid slots when selecting the $N_p = 2$ source slots for the prospective object tokens.

Object block and attention mask. The object block X_t^{obj} is inserted between the multimodal prefix and the action-generation prompt, as illustrated in Fig. 7(b). Attention is causal except for bidirectional connections within the object block. Let $\mathcal{B}$ denote the set of object block positions and $m_j \in \{0, 1\}$ indicate whether position j is a valid key. The attention mask is

$$M_{pj} = \begin{cases} 0, & m_j = 1,\ j \le p, \\ 0, & m_j = 1,\ p < j,\ p, j \in \mathcal{B}, \\ -\infty, & \text{otherwise.} \end{cases} \tag{22}$$

For the base object token associated with slot i at sequence position j_i, we set $m_{j_i} = \nu_{i,t}$, excluding invalid and padded slots independently of predicted visibility. Valid relation and prospective tokens retain $m_j = 1$. The identity bias modifies attention logits without changing the set of connections permitted by M.

Identity-guided object routing. Figure 7 illustrates the routing mechanism. We detail the implementation of Eqs. 8 and 9, using the same layer index ℓ and attention head index m as in the main text. Let $\bar{h}_p^\ell \in \mathbb{R}^D$ denote the normalized hidden state at query position p supplied to the backbone attention module. For each layer, $U^\ell \in \mathbb{R}^{d_r \times D}$ and $A^\ell \in \mathbb{R}^{d_r \times d_{\mathrm{id}}}$ project the contextual query representation and cached identity reference into a shared matching space, where $d_{\mathrm{id}} = 80$. The identity matching score is

$$b_{pi}^\ell = \left[\mathrm{Normalize}\left(U^\ell \bar{h}_p^\ell\right)\right]^\top \mathrm{Normalize}\left(A^\ell e_i^{\mathrm{id}}\right). \tag{23}$$

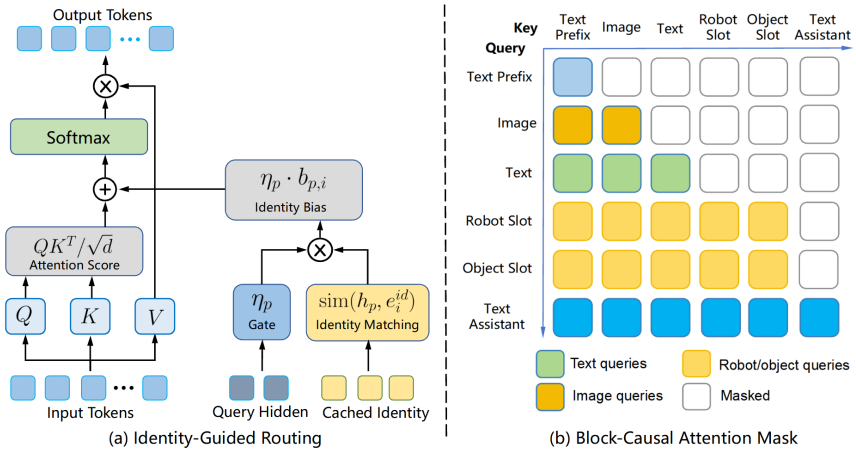


Figure 7: **Identity-guided routing and block-causal attention. (a) Identity-guided routing.** Cached identity references are matched with contextual query representations to produce a gated identity bias for valid base object-token keys. The bias is added to the standard attention logits before a shared softmax, while keys and values are computed from the full contextualized tokens. **(b) Block-causal attention mask.** The mask preserves causal attention across the sequence while allowing bidirectional attention within the valid object block. Invalid and padded slots remain masked, and identity-guided routing only modifies connections permitted by this mask.

We use $d_r = 32$ and $\mathrm{Normalize}(z) = z/\max(\|z\|_2, 10^{-6})$, yielding $b^{\ell}_{pi} \in [-1, 1]$. The projections U^{ℓ} and A^{ℓ} are shared across attention heads within each layer. Because e^{id}_i is retained independently of the contextualized object token, contextual updates to the token do not overwrite the cached identity reference. For attention head m, a query-dependent gate controls the contribution of the identity score:

$$\eta^{\ell,m}_p = \eta_{\max}\sigma\big((w^{\ell,m}_g)^{\top}\bar{h}^{\ell}_p + c^{\ell,m}_g\big), \qquad 0 < \eta^{\ell,m}_p < \eta_{\max}. \tag{24}$$

Here, $w^{\ell,m}_g \in \mathbb{R}^D$ and $c^{\ell,m}_g \in \mathbb{R}$ parameterize the head-specific gate, and σ denotes sigmoid. We use $\eta_{\max} = 2$ and initialize the routing gate to $\eta_0 = 0.05$. For the base object token associated with valid slot i at sequence position j_i, the attention logit is augmented as

$$s^{\ell,m}_{pj_i} = \frac{(q^{\ell,m}_p)^{\top} k^{\ell,m}_{j_i}}{\sqrt{d_k}} + \eta^{\ell,m}_p b^{\ell}_{pi} + M_{pj_i}, \qquad \nu_{i,t} = 1. \tag{25}$$

Here, $q^{\ell,m}_p$ and $k^{\ell,m}_{j_i}$ are the standard backbone query and key vectors, and d_k is the per-head key dimension. The identity term is applied only to keys of base object tokens whose source slots are valid; all other logits retain the standard backbone attention score and the same mask. Standard queries, keys, and values are computed from the full contextualized tokens. A single softmax is then applied over all permitted key positions:

$$\alpha^{\ell,m}_{pj} = \mathrm{softmax}_j\left(s^{\ell,m}_{pj}\right), \qquad y^{\ell,m}_p = \sum_j \alpha^{\ell,m}_{pj} v^{\ell,m}_j. \tag{26}$$

Here, $v^{\ell,m}_j$ is the value vector for key position j and attention head m, and $y^{\ell,m}_p$ is the head output. The cached identity modifies only the additive bias before softmax and does not replace the standard keys or values. Connections masked by $M = -\infty$ remain inaccessible.

Identity caching and gradient flow. The cached identity reference e^{id}_i is produced by the frozen object encoder and remains associated with slot i throughout the episode. In contrast, contextualized object tokens, query representations, and routing gate values depend on the current observation. During Stage II, U^{ℓ}, A^{ℓ}, and the routing gates are optimized jointly with the policy through the existing training objectives, without an additional routing loss. These routing parameters are frozen during Stage III. At inference, the cached identity references remain fixed for the episode, while contextualized object keys and values are recomputed from the current observation window.

Policy conditioning and auxiliary supervision. The contextualized object block contributes to the multimodal context C_t used by the flow-matching action expert. Let $o_{i,t}$ denote the final VLM hidden state of the object token associated with slot i. During Stage II, current states are decoded for all valid slots, while future states are decoded only for valid non-robot object slots using gated action conditioning as defined in Sec. 3.3. The auxiliary action proposal is used only by the future object state prediction branch. Prospective tokens are optimized through the action objective and are distinct from the decoded future object states. During rollout, the base object tokens, relation token, and prospective object tokens remain active together with identity-guided routing, while the auxiliary action proposal and the heads for current and future object state prediction are omitted.

B.2 Training Objectives

We expand the Stage II training objective in Eq. 13 by making the auxiliary loss coefficients explicit:

$$\mathcal{L} = w_{\text{act}}\mathcal{L}_{\text{act}} + \lambda_{\text{cur}}\ell_{\text{cur}} + \lambda_{\text{fut}}\ell_{\text{fut}} + \lambda_{\text{prop}}\ell_{\text{prop}} + \mathcal{L}_{\text{gate}}. \tag{27}$$

Here, $\mathcal{L}_{\text{act}}$ is the flow-matching objective defined in Eq. 2. The lowercase terms ℓ_{cur}, ℓ_{fut}, and ℓ_{prop} denote the auxiliary losses before applying their overall coefficients. They correspond to the weighted terms in the main text through

$$\mathcal{L}_{\text{cur}} = \lambda_{\text{cur}}\ell_{\text{cur}}, \qquad \mathcal{L}_{\text{fut}} = \lambda_{\text{fut}}\ell_{\text{fut}}, \qquad \mathcal{L}_{\text{prop}} = \lambda_{\text{prop}}\ell_{\text{prop}}. \tag{28}$$

Thus, Eq. 27 uses the same objective as the main text, with each overall coefficient applied once. The pretrained object encoder E_ω remains frozen throughout policy training. The identity matching projections and routing gates are learned through the existing objectives during Stage II, without a separate routing loss.

Slot importance gate regularization. The term $\mathcal{L}_{\text{gate}}$ regularizes the learned slot importance gates $g_{i,t}^{\text{slot}}$ used to rank slots, not the action conditioning gates $\beta_{i,t}$ or routing gates $\eta_p^{\ell,m}$. Let $\bar{g}$ denote the mean slot importance score over valid slots. We define

$$\mathcal{L}_{\text{gate}} = \lambda_{\text{upper}}\,\text{ReLU}(\bar{g} - g_{\text{ceil}})^2 + \lambda_{\text{lower}}\,\text{ReLU}(g_{\text{floor}} - \bar{g})^2. \tag{29}$$

Here, g_{floor} and g_{ceil} are the lower and upper thresholds for the mean gate score, and λ_{lower} and λ_{upper} weight the corresponding penalties. Their values are summarized in Table 10.

Current and future object supervision. The current-state objective supervises all valid slots, while future state supervision is restricted to valid non-robot object slots. Both objectives supervise bounding boxes, object centers, normalized areas, visibility, and segmentation masks. Let

$$\mathcal{S} = \{\text{box}, \text{center}, \text{area}, \text{vis}, \text{mask}\} \tag{30}$$

denote the set of supervised state components. We define

$$\ell_{\text{cur}} = \sum_{j\in\mathcal{S}} \lambda_j\, \ell_j^{\text{cur}}, \qquad \ell_{\text{fut}} = \sum_{j\in\mathcal{S}} \lambda_j\, \ell_j^{\text{fut}}, \tag{31}$$

where ℓ_j^{cur} and ℓ_j^{fut} denote the corresponding losses for current and future object state predictions. Bounding box, center, and area regression use mean squared error on targets normalized in image space. Visibility uses binary cross-entropy, while segmentation supervision combines binary cross-entropy and Dice loss. The component coefficients λ_j are shared by the current and future objectives. Each component is evaluated only when its corresponding target is available. Current state supervision is applied to all valid slots, including robot slots, whereas future state supervision excludes robot, invalid, and padded slots.

Auxiliary action proposal and gated conditioning. Let $o_{i,t}$ denote the final VLM hidden state of the object token associated with valid slot i. The auxiliary proposal head first applies mean pooling to the contextualized object tokens over valid slots:

$$c_t^{\text{prop}} = \frac{\sum_{i=1}^{N_s} \nu_{i,t}\,\text{sg}(o_{i,t})}{\sum_{i=1}^{N_s} \nu_{i,t} + \epsilon}, \tag{32}$$

where sg denotes stop-gradient and ϵ is a small constant for numerical stability. A lightweight proposal head then predicts a coarse action chunk

$$\tilde{A}_t = \pi_{\text{prop}}(c_t^{\text{prop}}) \in \mathbb{R}^{H_a \times D_a}. \tag{33}$$

The proposal is supervised by the demonstrated action chunk $A_t^\star$ through

$$\ell_{\text{prop}} = \frac{1}{H_a D_a} \left\| \tilde{A}_t - A_t^\star \right\|_2^2. \tag{34}$$

The proposal conditions future object state prediction and is produced by a separate head without using the output of the flow-matching action expert.

To condition future object state prediction, each proposed action is combined with a learned temporal embedding, projected into the same feature space as $o_{i,t}$, and averaged over the action horizon:

$$e_t^a = \frac{1}{H_a} \sum_{h=1}^{H_a} P_a\Big([\text{sg}(\tilde{A}_{t,h}); p_h]\Big), \tag{35}$$

where p_h is a learned temporal embedding, P_a is a learnable action projector, and h indexes the position within the action chunk. For each valid non-robot object slot, a scalar action conditioning gate controls the contribution of the position-aware action embedding:

$$\beta_{i,t} = \sigma(P_\beta(\text{sg}(o_{i,t}))), \tag{36}$$

where P_β is a lightweight gate network and σ denotes the sigmoid function. The representation for future object state prediction is then

$$o_{i,t}^{\text{cond}} = o_{i,t} + \gamma\, \beta_{i,t}\, e_t^a, \tag{37}$$

where γ is a fixed residual scale; we use $\gamma = 0.25$ in all experiments. For valid slots, current states are decoded from $o_{i,t}$, while for valid non-robot object slots, future states are decoded from $o_{i,t}^{\text{cond}}$:

$$\hat{Y}_{i,t}^{\text{cur}} = D_{\text{cur}}(o_{i,t}), \qquad \hat{Y}_{i,t+1:t+H_f}^{\text{fut}} = D_{\text{fut}}(o_{i,t}^{\text{cond}}). \tag{38}$$

During the first 10,000 Stage II updates, the demonstrated action chunk $A_t^\star$ replaces $\tilde{A}_t$ in the conditioning path with a probability that decreases linearly from 1 to 0. After this warmup period, the conditioning path uses the predicted proposal $\tilde{A}_t$ exclusively. The stop-gradient operations block gradients from the future object state objective to the proposal head through the action conditioning path and to $o_{i,t}$ through the gate input. The direct path from $o_{i,t}$ to D_{fut} remains differentiable. The action conditioning gates $\beta_{i,t}$ modulate the residual in the future object state branch and are not regularized by $\mathcal{L}_{\text{gate}}$. The proposal loss $\lambda_{\text{prop}}\ell_{\text{prop}}$ directly supervises the auxiliary action proposal.

Table 10 summarizes the Stage II loss coefficients and slot importance gate regularization parameters. LIBERO, RoboTwin 2.0, and the real-world experiments use the same values during joint policy adaptation. For state component j, the effective coefficient is $\lambda_{\text{cur}}\lambda_j$ for current predictions and $\lambda_{\text{fut}}\lambda_j$ for future predictions. The slot importance gate regularizer is added directly to the Stage II objective without an additional global coefficient.

B.3 Training and Inference Configuration

Training stages. Training proceeds in three stages across simulation and real-world experiments, as summarized in Table 11. Stage I pretrains the object encoder using the configuration and objectives described in Sec. A.2. Stage II freezes the pretrained encoder and jointly optimizes the VLM, action expert, and policy-side object modules using the action and auxiliary objectives. Stage III freezes the VLM and all object-related modules and updates only the action expert with $\mathcal{L}_{\text{act}}$.

Stage III action refinement. Stage III refines the flow-matching action expert for 40,000 additional optimizer updates after Stage II. During this stage, the VLM, object encoder, object token projections, identity-guided routing modules, relation and prospective token modules, and auxiliary object prediction modules are frozen. Only the action expert is updated using $\mathcal{L}_{\text{act}}$. The optimizer and learning-rate scheduler are continued from Stage II rather than reinitialized at the stage boundary.

Table 10: **Loss coefficients used during Stage II.** Overall coefficients appear in Eq. 27, state loss coefficients appear in Eq. 31, and slot importance gate regularization parameters correspond to Eq. 29.

Loss term / parameter	Symbol	Value
Overall coefficients		
Action flow matching	w_{act}	1.0
Current state supervision	λ_{cur}	0.5
Future object state supervision	λ_{fut}	0.1
Action proposal	λ_{prop}	0.05
Components within each state loss		
Bounding box	λ_{box}	0.25
Object center	λ_{center}	1.0
Normalized area	λ_{area}	0.1
Visibility	λ_{vis}	0.2
Segmentation mask (BCE + Dice)	λ_{mask}	0.05
Slot importance gate regularization		
Upper penalty weight	λ_{upper}	0.002
Lower penalty weight	λ_{lower}	0.01
Gate mean ceiling	g_{ceil}	0.40
Gate mean floor	g_{floor}	0.12

Table 11: **Three-stage training schedule.** The schedule is shared across simulation and real-world experiments. Step counts denote optimizer updates within each stage. Policy-side object modules include object token projections, identity matching projections and routing gates, relation and prospective token modules, and auxiliary prediction heads.

Stage	Steps	Trainable modules	Frozen modules
I: Encoder pretraining	80,000	Learnable encoder modules and prediction heads	SAM 2.1 image backbone, CLIP text encoder, and DINOv2 image encoder
II: Joint policy adaptation	80,000	VLM, action expert, and policy-side object modules	Pretrained object encoder
III: Action refinement	40,000	Action expert	VLM, object encoder, and policy-side object modules

Training of ObjDream. Training follows the three-stage optimization procedure described above, with the object encoder pretrained from offline annotations and subsequently frozen during policy optimization. The corresponding training schedule, loss coefficients, and optimization settings are summarized in Tables 11, 10, and 12.

Optimization. Table 12 summarizes the optimization configuration for Stage II joint policy adaptation. Separate learning rates are assigned to the VLM, action expert, and newly initialized policy-side object modules. The identity matching projections and routing gates use the learning rate assigned to policy-side object modules. Batch sizes count demonstration sequences, while repeated noise–time sampling reuses the encoded observation for flow-matching training.

Action expert optimization. The flow-matching action expert is optimized with AdamW using $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and weight decay 10^{-8}. Its learning rate follows the benchmark-specific values in Table 12 with cosine decay. The optimizer and scheduler states are continued across the transition from Stage II to Stage III without reinitialization.

Table 12: **VLA policy optimization configuration.** Learning rates refer to Stage II joint policy adaptation. Effective batch sizes assume eight data-parallel workers and one gradient accumulation step.

Parameter	LIBERO	RoboTwin	Real world
Optimizer	AdamW	AdamW	AdamW
Optimizer (β_1, β_2)	$(0.9, 0.95)$	$(0.9, 0.95)$	$(0.9, 0.95)$
Learning-rate schedule	Cosine	Cosine	Cosine
VLM learning rate	5×10^{-7}	10^{-5}	10^{-5}
Action expert learning rate	10^{-6}	10^{-4}	10^{-4}
Object module learning rate	10^{-4}	10^{-4}	10^{-4}
Sequences per device	2	4	4
Effective sequence batch	16	32	32
Global gradient-norm clipping	1.0	1.0	1.0
Noise–time pairs per sequence	2	8	2

Table 13: **Observation and action settings.** The full predicted action chunk is executed before replanning, so H_a denotes both the predicted and executed action horizon. The future prediction horizon H_f denotes the number of future object state prediction steps.

Parameter	LIBERO	RoboTwin	Real world
Policy image resolution	224×224	224×224	224×224
Action dimension D_a	7	14	8
Predicted / executed action horizon H_a	16	50	16
Action representation	End-effector + gripper	Joints + grippers	Seven joints + gripper
Future prediction horizon H_f	8	8	8
Auxiliary mask resolution	32×32	32×32	32×32
Flow integration steps	4	4	4

Observation and action settings. Table 13 summarizes the observation and action settings across simulation and real-world experiments. LIBERO uses external and wrist camera observations; the object encoder processes the external view. RoboTwin 2.0 uses a third-person camera and two wrist cameras; the object encoder processes the third-person view, while the VLM receives all three views. For both simulation benchmarks, the object encoder uses a causal observation window of four frames. LIBERO-Plus directly uses the corresponding LIBERO-trained policy without additional fine-tuning. The real-world policy receives RGB observations from a fixed external camera and a wrist-mounted camera.

Inference. At episode initialization, Qwen3-VL-8B (Bai et al., 2025) processes the instruction x and initial observation I_0 once to construct the phrase set P_0, which is cached for all subsequent control steps. For phrase-conditioned object slots, the frozen object encoder predicts the initial regions used for DINOv2 reference pooling. The resulting visual references d_i and complete identity references e_i^{id} are cached throughout the episode, following Sec. A.2. Grounding DINO localization and SAM2 video mask propagation are used only for offline annotation and are not executed during rollout; the frozen SAM2 image backbone remains active within the object encoder. At each replanning step, the frozen encoder processes the causal image window $\mathbf{I}_t$ using P_0 and the cached identity references to produce the object slots O_t. The slots are projected into base object tokens and combined with the relation and prospective object tokens to form the object block. The block is inserted into the VLM token sequence, and identity-guided routing guides attention to the base object tokens. The action expert generates an action chunk by integrating the learned velocity field from Gaussian noise. Actions are denormalized using training-set statistics, and the full predicted action chunk is executed before replanning from the latest observation. The policy-side auxiliary action proposal and the heads for current and future object state prediction are omitted during rollout.

Table 14: **Future object state prediction accuracy on LIBERO.** Current state persistence repeats the current object state at each future target. Bold indicates the better value within each comparison.

Subset	Method	Center error (%) ↓	Box IoU ↑	Mask IoU ↑
Unclamped	Current state persistence	6.37	0.634	0.564
	Future prediction	**1.60**	**0.690**	**0.616**
Displacement	Current state persistence	13.91	0.287	0.245
	Future prediction	**2.03**	**0.575**	**0.542**

C Future Object State Prediction Analysis

We analyze whether the auxiliary future object state decoder learns nontrivial object state evolution rather than simply preserving the current state. We evaluate three representative LIBERO manipulation tasks covering object placement and articulated-object interaction. Only valid non-robot object slots are evaluated; robot and padding slots are excluded. For each prediction origin, the decoder predicts object centers, bounding boxes, and masks at every consecutive future step $h \in \{1, \dots, H_f\}$, with $H_f = 8$.

We compare the learned decoder against a *current state persistence* baseline that repeats the current object estimate at each future step. We report an *Unclamped* subset containing only predictions whose future targets are directly available within the trajectory, and a *Displacement* subset containing object–target pairs whose centroid changes by more than 1% of the image diagonal. Center error is normalized by the image diagonal, while box and mask IoU measure predicted object extent.

The learned decoder consistently improves over current state persistence. On the Unclamped subset, the learned decoder has lower center error and higher box and mask IoU than current state persistence. The gains are more pronounced under object displacement, where center error decreases from 13.91% to 2.03% and box and mask IoU improve from 0.287 and 0.245 to 0.575 and 0.542. These results indicate that the auxiliary decoder captures nontrivial future object state changes rather than merely propagating the current state.

Qualitative future object state predictions. Figure 8 shows representative LIBERO future object state predictions. Although the decoder predicts all horizons $h = 1, \dots, H_f$, we display only $h = 1, 4, 8$ for clarity. Figures 10 and 9 show additional predictions on real-world tasks and RoboTwin 2.0, respectively.

Table 15: **Sensitivity of future object state prediction to action conditioning on LIBERO.** *Native Predicted Action* uses the original auxiliary action proposal; *No Action* removes the position-aware action embedding; and *Mismatched Action* replaces it with the position-aware embedding of a predicted action chunk from another sample within the same task. All other model components and prediction targets are kept fixed.

Subset	Conditioning	Center error (%) ↓	Box IoU ↑	Mask IoU ↑
Unclamped	No Action	1.90	0.659	0.609
	Mismatched Action	2.02	0.660	0.587
	Native Predicted Action	**1.60**	**0.690**	**0.616**
Displacement	No Action	2.35	0.546	0.534
	Mismatched Action	2.71	0.532	0.486
	Native Predicted Action	**2.03**	**0.575**	**0.542**

Effect of action conditioning on future object state prediction. To examine whether the decoder uses action information, we intervene on the action conditioning path using the same trained model and evaluation tasks. Model parameters, contextualized object tokens, and action conditioning gates are kept fixed. *Native Predicted Action* uses the original auxiliary action proposal; *No Action* sets the position-aware action embedding to zero; and *Mismatched Action* replaces it with the

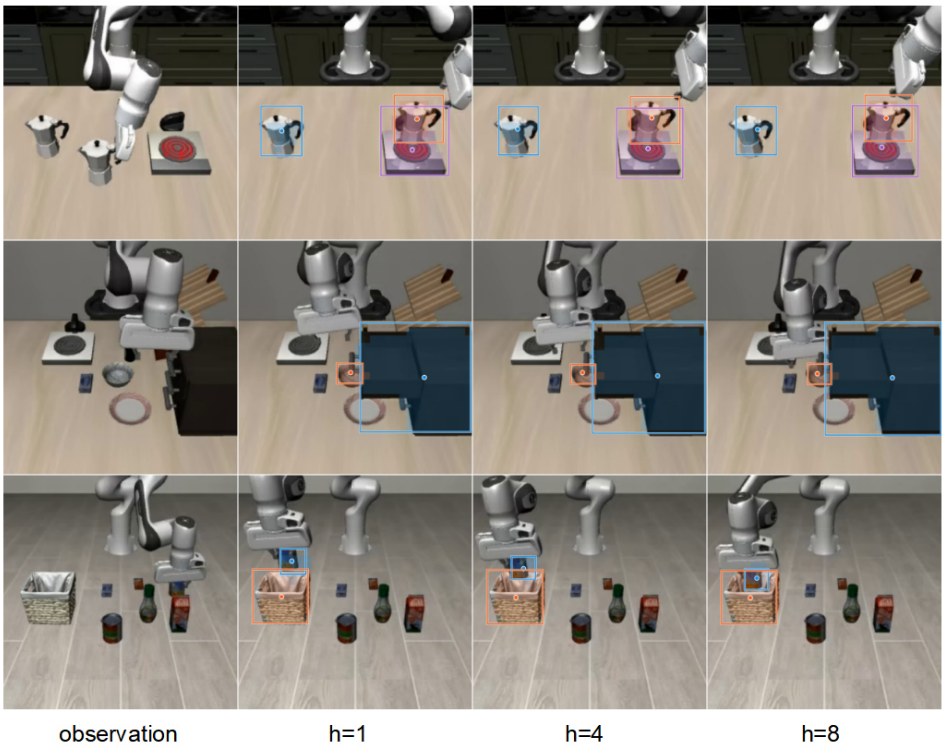


Figure 8: **Qualitative future object mask predictions on LIBERO.** Rows show three representative manipulation tasks. The decoder predicts all future steps $h = 1, \ldots, H_f$ for non-robot objects; for clarity, we visualize only $h = 1, 4, 8$. The predicted masks are overlaid on the corresponding future RGB frames, which are used only for visualization and are not provided to the decoder.



Figure 9: **Qualitative future object state predictions on RoboTwin 2.0.** Rows show representative tasks from the fixed eight-task subset. The decoder predicts all future steps for non-robot objects; for clarity, we visualize masks and bounding boxes only at $h = 1, 4, 8$. The predictions are overlaid on the corresponding future RGB frames, which are used only for visualization and are not provided to the decoder.

position-aware embedding of a predicted action chunk from another sample within the same task. As shown in Table 15, the native condition yields the lowest center error and the highest box and mask IoU on both subsets. On the Unclamped subset, center error is 1.60% with the native condition,

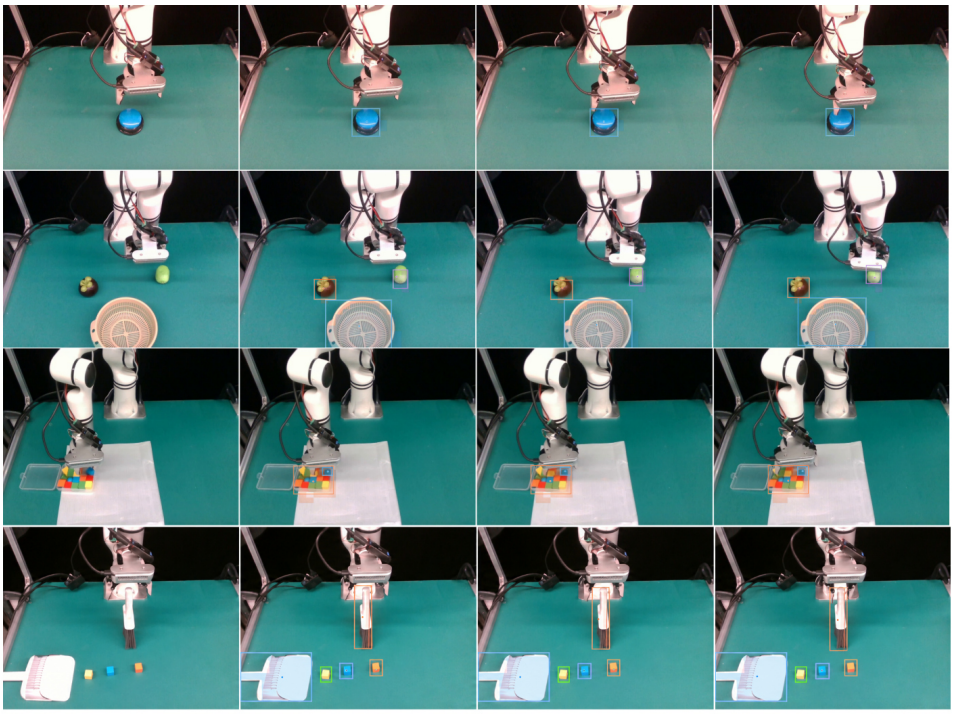


Figure 10: **Qualitative future object state predictions on real-world tasks.** Rows show button pressing, two-fruit placement, block pick-and-place, and tabletop sweeping. The decoder predicts all future steps for non-robot objects; for clarity, we visualize masks and bounding boxes only at $h = 1, 4, 8$. The predictions are overlaid on the corresponding future RGB frames, which are used only for visualization and are not provided to the decoder.

compared with 1.90% without action conditioning and 2.02% with mismatched actions. On the Displacement subset, center error increases from 2.03% to 2.35% and 2.71%, respectively. Box IoU decreases from 0.575 to 0.546 and 0.532, while mask IoU decreases from 0.542 to 0.534 and 0.486. These interventions indicate that the future object state decoder uses the action condition, while not establishing counterfactual dynamics under arbitrary action sequences.

D Simulation Evaluation Protocols

This section describes the evaluation protocols for LIBERO, LIBERO-Plus, and the fixed RoboTwin 2.0 task subset. For ObjDream, offline annotation and object encoder pretraining follow Sec. A, while model integration, training, and inference follow Sec. B. All simulation results use the task-completion criteria defined by the corresponding benchmark environments.

D.1 LIBERO and LIBERO-Plus

LIBERO. We evaluate the Spatial, Object, Goal, and Long suites of LIBERO (Liu et al., 2023). Each suite contains ten tasks. Each task is evaluated over 50 rollouts using the benchmark-provided initial states, yielding 500 rollouts per suite and 2,000 rollouts across the four suites. A rollout is considered successful when the environment reports task completion within the evaluation horizon. We report the mean success rate for each suite and the equally weighted average across the four suites.

LIBERO-Plus. LIBERO-Plus (Fei et al., 2026) evaluates robustness under controlled changes in camera viewpoint, robot initialization, object layout, language, lighting, background, and sensor noise. We directly evaluate the corresponding LIBERO-trained checkpoint under these perturbations without additional fine-tuning. For language perturbations, P_0 is constructed once at episode

initialization from the perturbed instruction and initial observation following Sec. B.3; the same perturbed instruction is provided to the VLA policy. For each perturbation category, success rate is computed over the benchmark evaluation instances across the four LIBERO suites. The overall score is computed over all evaluated instances, accounting for the different numbers of instances across perturbation categories.

Result analysis. On LIBERO-Long, ObjDream achieves a success rate of 97.4%, outperforming the evaluated object-centric and predictive baselines and demonstrating strong capability in long-horizon manipulation involving multiple task-relevant objects. On LIBERO-Plus, the same LIBERO-trained policy is evaluated under controlled distribution shifts without additional fine-tuning. ObjDream achieves an overall success rate of 84.4%, second only to $\pi_{0.5}$, which benefits from large-scale pretraining on robot data, while outperforming the other evaluated baselines. This result suggests that explicitly extracting and integrating task-relevant object information provides an effective source of robustness even without large-scale pretraining on robot data. The strongest gains appear under language, lighting, and background perturbations, where ObjDream achieves the best performance among the evaluated methods. In ObjDream, cached identity references guide attention to object tokens, which provide object-specific context alongside the original visual and language inputs. Overall, these results show competitive manipulation performance and robustness to semantic and appearance variations.

D.2 RoboTwin 2.0 Evaluation

Task subset and observations. We evaluate a fixed subset of eight RoboTwin 2.0 tasks (Chen et al., 2025b), listed in Table 16. The subset covers diverse bimanual manipulation behaviors, including articulated-object interaction, object relocation, scanning, placement, and switch or button interaction. The object encoder processes the benchmark's main `head_camera` view, while the VLA policy additionally receives the wrist camera observations as described in Sec. B.3. All tasks use the benchmark-defined observation interface and task-specific success criteria. Reported RoboTwin results and averages are computed over this fixed eight-task subset.

Training and Easy/Hard settings. For all eight selected tasks, both object encoder pretraining and policy training use 50 `demo_clean` demonstrations per task. The resulting policy checkpoint is evaluated under the benchmark's Easy (`demo_clean`) and Hard (`demo_randomized`) settings. The Hard setting changes scene appearance and configuration, including background, clutter, lighting, and table geometry, while preserving the task objective, robot embodiment, and observation interface. The same trained checkpoint is used for both Easy and Hard evaluation, without setting-specific fine-tuning.

Evaluation protocol. Each selected task is evaluated over 100 rollouts under both the Easy and Hard settings. Success is determined by the benchmark simulator's task-completion signal, and episodes that do not complete the task within the evaluation horizon are counted as failures. The final Easy and Hard scores are computed separately as arithmetic means over the eight tasks, with each task receiving equal weight.

Ablation configurations. The ablations in Table 2 use the same LIBERO training data and evaluation protocol as the full model. The *Backbone baseline* removes the complete object interface and retains only the Qwen3-VL backbone and flow-matching action expert. The *w/o ObjRel & ObjDyn* variant retains the base object tokens, factorized object injection, identity-guided routing, and current object state supervision, while removing both the relation modeling module and future object state prediction. The *w/o ObjDyn* variant retains object relation modeling but removes the future object state prediction branch and its associated supervision. Conversely, *w/o ObjRel* removes the relation token and relational aggregation while retaining future object state supervision. The *w/o Fact. Injection* variant replaces the separate identity and dynamic state projections with a unified projection for constructing each object token, while retaining the cached identity references, identity-guided routing, relation modeling, and future object state supervision. The *w/o Id. Routing* variant removes the additive identity-guided attention bias and uses standard VLM attention, while retaining the factorized object tokens, relation modeling, and future object state supervision. Unless otherwise specified, all remaining components and training settings are kept unchanged.

Table 16: **Performance on the fixed eight-task RoboTwin 2.0 subset.** Success rates (%) are reported under Easy and Hard settings. Tasks are ordered alphabetically, and the average is computed equally over the eight listed tasks. Within each pretraining group, the best results are shown in bold and the second-best results are underlined, including ties. Public leaderboard results are recomputed over the same eight-task evaluation subset.

Method	RoboTwin 2.0																	
	Adjust Bottle		Click Bell		Grab Roller		Move Can Pot		Open Microwave		Place Can Basket		Scan Object		Turn Switch		Average	
	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard
Without large-scale pretraining on robot data																		
ACT (Zhao et al., 2023)[S]	97	23	58	3	94	25	22	4	**86**	0	1	0	2	0	5	2	45.63	7.13
DP (Chi et al., 2025)[S]	97	0	54	0	98	0	39	0	5	0	18	0	9	0	36	1	44.50	0.13
StarVLA (Community, 2026)[M]	84	0	75	7	94	44	33	0	10	1	4	0	63	5	16	0	47.38	7.13
EventVLA (Yang et al., 2026)[M,†]	**100**	**24**	94	**48**	97	48	62	0	55	**44**	76	7	27	3	24	**8**	66.88	**22.75**
FastWAM (Yuan et al., 2026)[M]	99	0	**95**	7	**100**	2	**96**	0	17	0	**77**	0	68	0	**39**	3	**73.88**	1.50
ObjDream (Ours)[M]	98	13	38	24	98	**78**	59	**5**	41	43	36	**9**	**81**	**6**	31	4	60.25	**22.75**
With large-scale pretraining on robot data																		
RDT (Liu et al., 2025)[S]	81	75	80	9	74	43	25	12	37	20	19	6	4	1	35	15	44.38	22.63
π_0 (Black et al., 2024)[S]	90	56	44	3	96	80	58	21	80	50	41	5	18	1	27	23	56.75	29.88
$\pi_{0.5}$ (Physical Intelligence et al., 2025)[M]	99	77	31	35	99	97	60	10	**86**	**59**	56	10	49	40	**52**	25	66.50	44.13
Xiaomi-Robotics-0 (Cai et al., 2026)[M]	**100**	9	**100**	57	99	12	63	7	45	16	52	7	46	8	38	14	67.88	16.25
X-VLA (Zheng et al., 2026)[M]	99	33	**100**	63	**100**	20	58	0	43	5	**70**	9	**57**	9	40	6	**70.88**	18.13
GigaBrain-0.7 (Team et al., 2026)[M]	98	**98**	98	**84**	**100**	**100**	**86**	**82**	82	50	20	**22**	32	**46**	42	**52**	69.75	**66.75**

Notes: [S] denotes single-task fine-tuning and [M] denotes multi-task joint training. Public leaderboard results are recomputed over the same fixed eight-task evaluation subset while retaining each method's publicly reported training and pretraining configuration. StarVLA and ObjDream are evaluated on the same fixed eight-task subset. ObjDream uses multi-task joint policy training over these eight tasks, with 50 `demo_clean` demonstrations per task. Large-scale pretraining on robot data refers to additional robot demonstration pretraining beyond benchmark-specific task data; general vision-language or video pretraining is not included in this grouping. [†]EventVLA is grouped according to its released simulation-training configuration. Leaderboard data accessed on September 14, 2026.

Baseline results. We compare ObjDream with publicly reported RoboTwin 2.0 results from the official leaderboard.[1] For each baseline, we extract the reported results on the same eight evaluation tasks and recompute the corresponding subset averages. Baseline methods retain their publicly reported training and pretraining configurations. Single-task and multi-task training settings are indicated in Table 16.

Result analysis. RoboTwin 2.0 provides a challenging evaluation of bimanual manipulation under substantial scene variation. Although ObjDream is trained only on clean demonstrations, the same checkpoint achieves 60.25% success on Easy and 22.75% on Hard. In the expanded comparison in Table 16, ObjDream matches EventVLA on the highest Hard average among the listed methods without large-scale pretraining on robot data. Overall, these results demonstrate competitive robustness under randomized scene variations without setting-specific fine-tuning.

E Real-World Experimental Setup and Evaluation

This section supplements Sec. 4.3 with details of the physical platform, demonstration data, baseline training, and evaluation protocol.

E.1 Robot Platform and Policy Interface

Robot platform and observations. The experiments use a seven-joint RealMan robot arm equipped with an OmniPicker electric gripper. Two Intel RealSense D435 cameras provide RGB observations, with one mounted at the wrist and the other providing a fixed external view of the tabletop workspace. For ObjDream, the VLM receives both camera views, while the object encoder processes the external view. VLM image inputs are resized to 224×224, while object encoder preprocessing follows Sec. A.2. The proprioceptive state contains seven joint coordinates and one gripper state, yielding an eight-dimensional representation.

[1] https://robotwin-platform.github.io/leaderboard

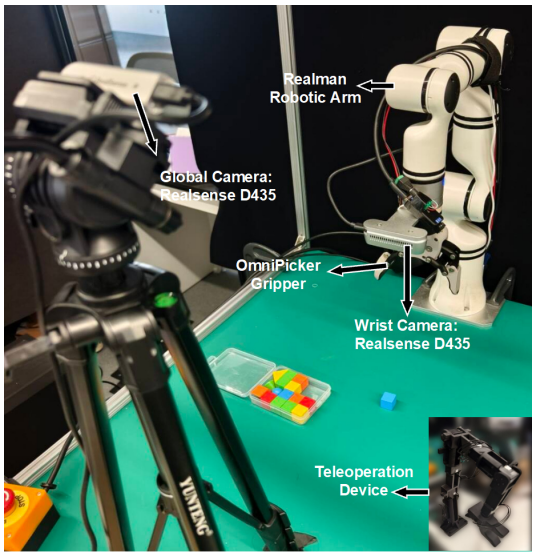


Figure 11: **Real-world experimental setup.** The platform consists of a RealMan robot arm equipped with an OmniPicker electric gripper and two Intel RealSense D435 cameras providing fixed external and wrist views. The bottom-right inset shows the teleoperation device used to collect demonstration trajectories.

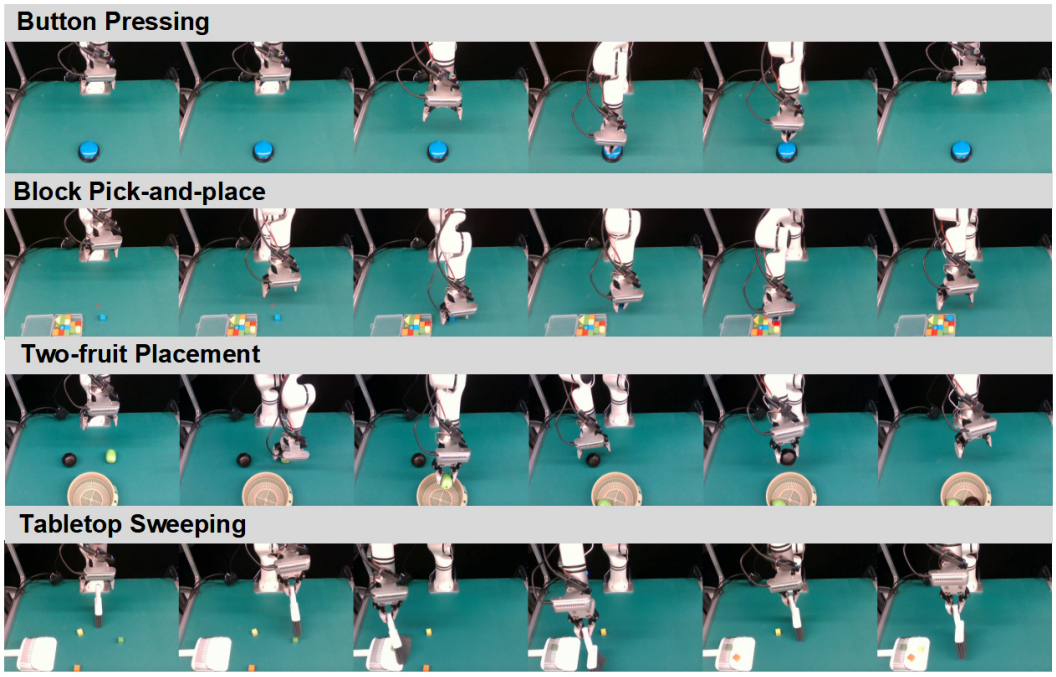


Figure 12: **Real-world manipulation tasks and representative rollouts.** We evaluate ObjDream on four physical manipulation tasks: *button pressing*, *block pick-and-place*, *two-fruit placement*, and *tabletop sweeping*. Each row shows a representative execution sequence progressing from left to right. The tasks cover contact-based interaction, object grasping and placement, sequential multi-object manipulation, and long-horizon tool use.

Policy action interface. The policy predicts and executes action chunks of length $H_a = 16$. Each action contains seven joint targets and one gripper command, giving an action dimension of $D_a = 8$.

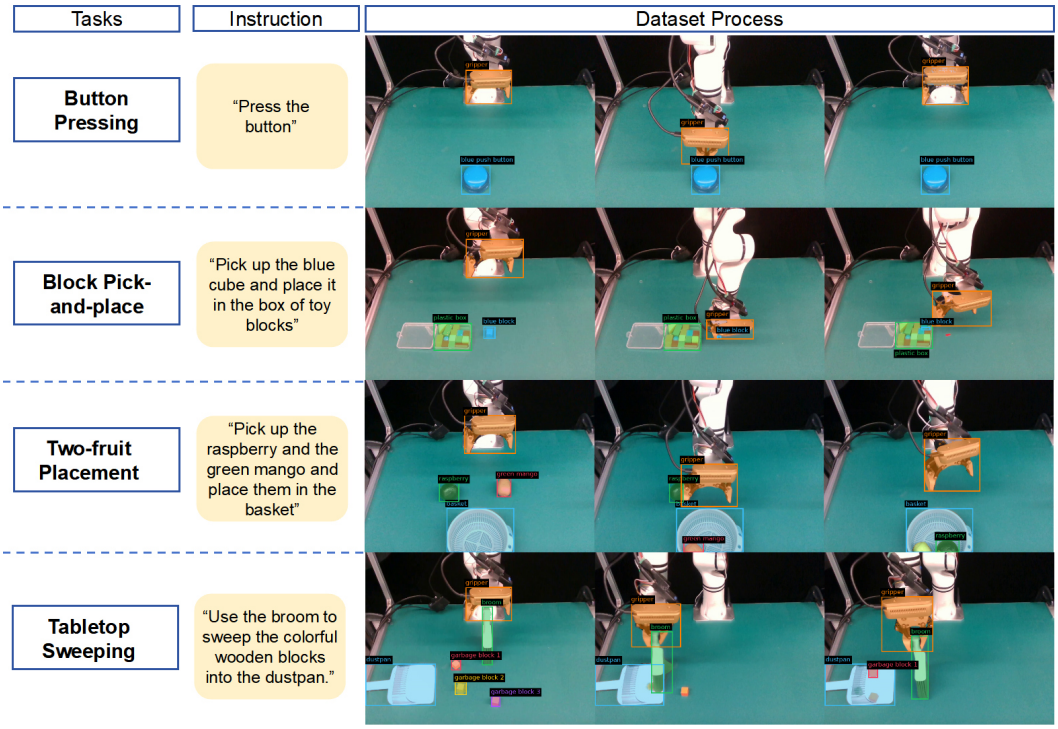


Figure 13: **Real-world demonstrations and offline object annotations.** Each row pairs a task instruction with representative demonstration frames arranged chronologically from left to right. Colored overlays, bounding boxes, and labels visualize annotations of task-relevant objects and the robot gripper. These annotations provide offline supervision for object encoder pretraining across the four manipulation tasks.

The flow-matching action expert generates each chunk using four integration steps. Actions are denormalized using training-set statistics, and the full predicted chunk is executed before replanning from the latest observation. Object tokens and identity-guided routing remain active during rollout, while the policy-side auxiliary action proposal and the heads for current and future object state prediction are omitted. Episode initialization and rollout follow Sec. B.3.

E.2 Demonstration Data and Training

Demonstration data. We collect 100 successful demonstration trajectories for each of the four tasks, yielding 400 trajectories in total. Demonstrations are recorded at 20 Hz in the training environment. Each trajectory contains external and wrist RGB observations, robot state and action data, and an associated language instruction. All methods use the same recorded task demonstrations for real-world adaptation. ObjDream and $\pi_{0.5}$ are jointly trained on the pooled four-task dataset, while ControlVLA uses the corresponding task-specific subsets described below. Each baseline retains its native observation and action interface when processing the shared demonstration data.

Training of ObjDream. We train a single instruction-conditioned policy jointly on all four real-world tasks. The language instruction specifies the task goal and provides the context used to construct the corresponding object phrases. Training follows the three-stage optimization procedure and auxiliary objectives described in Sec. B, with the object encoder pretrained from offline object annotations and subsequently frozen during policy optimization. The corresponding training schedule, loss coefficients, and optimization settings are summarized in Tables 11, 10, and 12.

Training of $\pi_{0.5}$. We reproduce $\pi_{0.5}$ (Physical Intelligence et al., 2025) using the official OpenPI implementation and initialize it from the released `pi05_base` checkpoint. A single policy is fine-tuned jointly on all 400 demonstrations for 40,000 optimizer updates. Fine-tuning uses AdamW with

Table 17: **Success criteria for real-world tasks.** Each method is evaluated over 20 trials per task under each of the ID and OOD settings. Success requires completion of the full task objective.

Task	Success criterion
Button pressing	The designated button is depressed.
Block pick-and-place	The target block is placed into the designated gap in the block box.
Two-fruit placement	Both target fruits are placed inside the fruit basket.
Tabletop sweeping	The robot picks up the brush and sweeps all target blocks into the collection box.

a global batch size of 32, a peak learning rate of 2.5×10^{-5}, and 1,000 warmup updates followed by cosine decay. The policy receives external and wrist RGB observations, proprioceptive state, and the task instruction while retaining its native state and action processing.

Training and deployment of ControlVLA. We reproduce ControlVLA (Li et al., 2025a) using the official implementation and initialize each policy from the released DROID-pretrained checkpoint. Following its task-specific training setup, we train a separate policy for each task using the same 100 demonstrations for that task. Each policy is fine-tuned for 80,000 optimizer updates with a global batch size of 64, using a policy learning rate of 10^{-4} and an observation encoder learning rate of 3×10^{-5}. We use 1,000 warmup updates followed by cosine decay. During rollout, ControlVLA retains its online object mask conditioning. SAM2 Hiera-Tiny provides object masks from the external camera stream, initialized from bounding-box prompts in the first frame and propagated over subsequent observations. If propagation fails after initialization, the most recent valid mask is reused. ControlVLA also retains its Tool Center Point (TCP) action interface. The policy predicts relative TCP action chunks with a horizon of 16, which are converted to absolute TCP poses and gripper commands and executed through the RealMan robot interface.

E.3 Evaluation Protocol

Evaluation settings. We compare ObjDream with $\pi_{0.5}$ and ControlVLA under the in-distribution (ID) and out-of-distribution (OOD) settings defined in the main text. The ID setting uses tabletop appearance, object configurations, and illumination similar to those encountered during training. The OOD setting introduces changes in tablecloth color, obstacle objects, object layouts, and lighting. The same OOD perturbation configuration is used for all methods within each task. No additional fine-tuning is performed for either evaluation setting. Each method is evaluated over 20 trials per task under each setting, yielding 80 ID trials and 80 OOD trials per method.

Task success criteria. Each trial receives a binary success label according to the criteria in Table 17. The same criteria are applied to all methods under both ID and OOD conditions. Success requires completion of the full task objective.

Evaluation metric. For a given method, let $S_{k,s}$ denote the number of successful trials for task k under setting s. The task success rate and the equally weighted average across the four tasks are

$$\mathrm{SR}_{k,s} = \frac{S_{k,s}}{20} \times 100\%, \qquad \overline{\mathrm{SR}}_s = \frac{1}{4} \sum_{k=1}^{4} \mathrm{SR}_{k,s}, \tag{39}$$

where $s \in \{\mathrm{ID}, \mathrm{OOD}\}$. ID and OOD results are reported separately.

E.4 Real-World Result Analysis

ObjDream achieves the highest average ID success rate among the evaluated methods and remains competitive with $\pi_{0.5}$ under OOD conditions. The real-world evaluation provides a complementary view of how different sources of prior knowledge and object-level conditioning affect generalization. $\pi_{0.5}$ (Physical Intelligence et al., 2025) serves as a strong baseline with competitive performance under both ID and OOD conditions. Despite not relying on large-scale pretraining on

robot data, ObjDream achieves a higher average success rate under ID evaluation and comparable aggregate performance to $\pi_{0.5}$ under OOD evaluation.

ObjDream shows notable OOD gains on two-fruit placement and tabletop sweeping. On *tabletop sweeping*, ObjDream outperforms $\pi_{0.5}$ by 10 percentage points under OOD conditions, with a similar gain on *two-fruit placement*. Qualitatively, ObjDream can still complete the sweeping task when target blocks are initialized in challenging locations, using the brush to progressively move them into the collection box. The object encoder produces structured object slots that are projected into VLM object tokens, while cached identity references guide attention to these tokens alongside the original visual and language context.

The comparison reflects complete policy systems with different perception and action interfaces. ControlVLA (Li et al., 2025a) uses object information through online object mask conditioning, whereas ObjDream integrates object tokens into the VLM context used by the flow-matching action expert. The compared methods use the same demonstration trajectories but retain their respective perception pipelines, action representations, and task training configurations. Together, these results demonstrate competitive real-world generalization without large-scale pretraining on robot data.

Inference latency. We measure steady-state policy inference latency on a single NVIDIA RTX 4090 with batch size 1, using 50 warm-up iterations followed by 200 synchronized runs. ObjDream requires approximately 177.4 ms to generate a complete action chunk with four flow-matching integration steps, compared with 84.6 ms for $\pi_{0.5}$ and 79.4 ms for ControlVLA, where the latter includes online SAM2 Hiera-Tiny mask prediction. Network communication, sensor acquisition, robot execution, and one-time episode initialization are excluded. The auxiliary future object state prediction branch of ObjDream is not executed during rollout.

F Limitations and Broader Impact

Our current evaluation covers a limited range of robot embodiments and manipulation settings. Future work may incorporate richer 3D spatial information into the object representation and evaluate ObjDream across broader scenarios. More broadly, while improved object-level reasoning can support more robust robotic manipulation, deployment in unconstrained physical environments should still incorporate appropriate safety mechanisms and human oversight.

G Qualitative Rollout Visualizations

We provide qualitative rollout examples to complement the quantitative evaluations in the main paper. The visualizations cover both real-world and simulation environments and illustrate representative policy executions under standard and perturbed conditions. For the real-world experiments, Figs. 14–17 compare ControlVLA, $\pi_{0.5}$, and ObjDream across the four manipulation tasks under both ID and OOD conditions. Each real-world figure shows temporally ordered rollout frames for all three methods; yellow annotations highlight execution events or failure modes, while the symbols on the right indicate the final success or failure of each rollout. These selected examples illustrate individual execution outcomes rather than aggregate success rates. Figures 18 and 19 present representative rollouts from the fixed eight-task RoboTwin 2.0 subset under Easy and Hard settings, respectively. Figures 20 and 21 show representative rollouts from standard LIBERO and LIBERO-Plus.

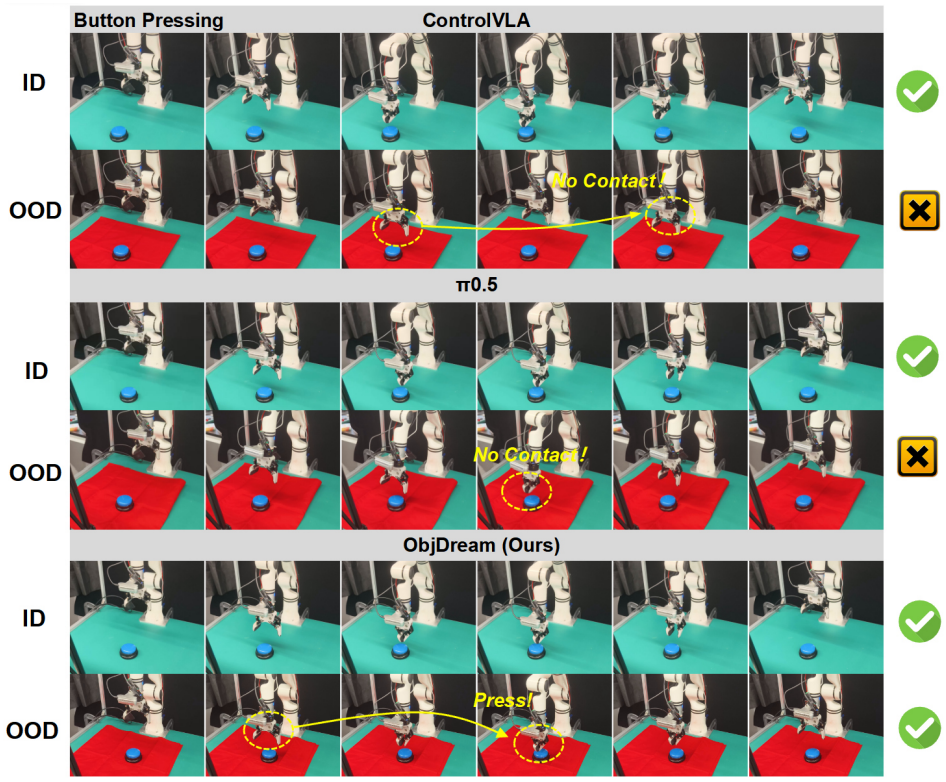


Figure 14: **Button pressing.** ControlVLA, $\pi_{0.5}$, and ObjDream under ID and OOD.

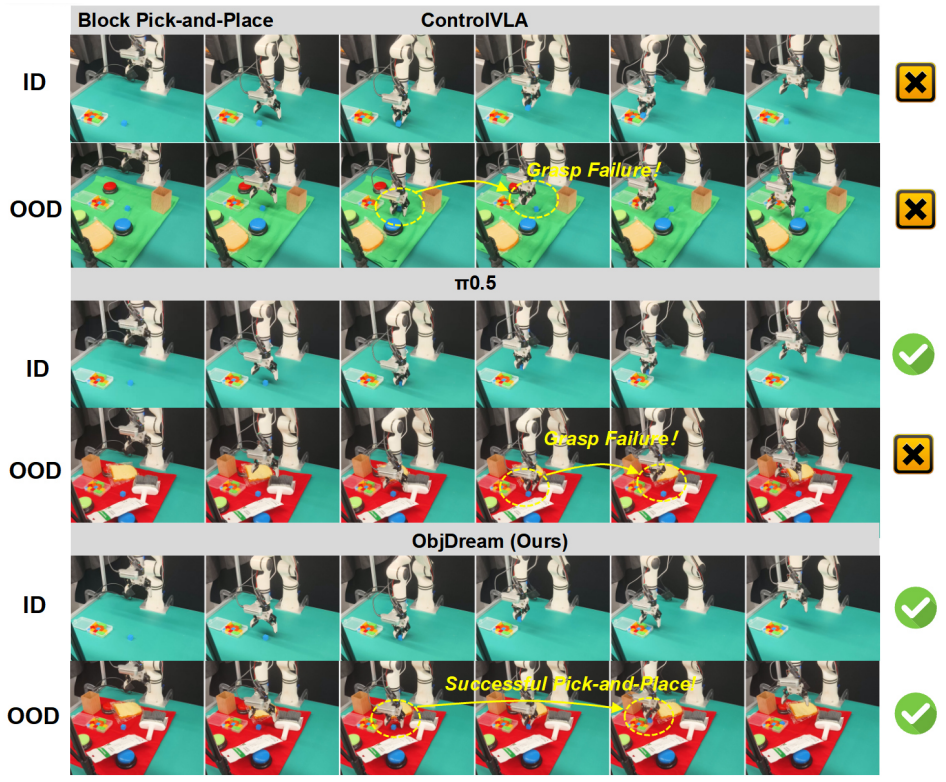


Figure 15: **Block pick-and-place.** ControlVLA, $\pi_{0.5}$, and ObjDream under ID and OOD.

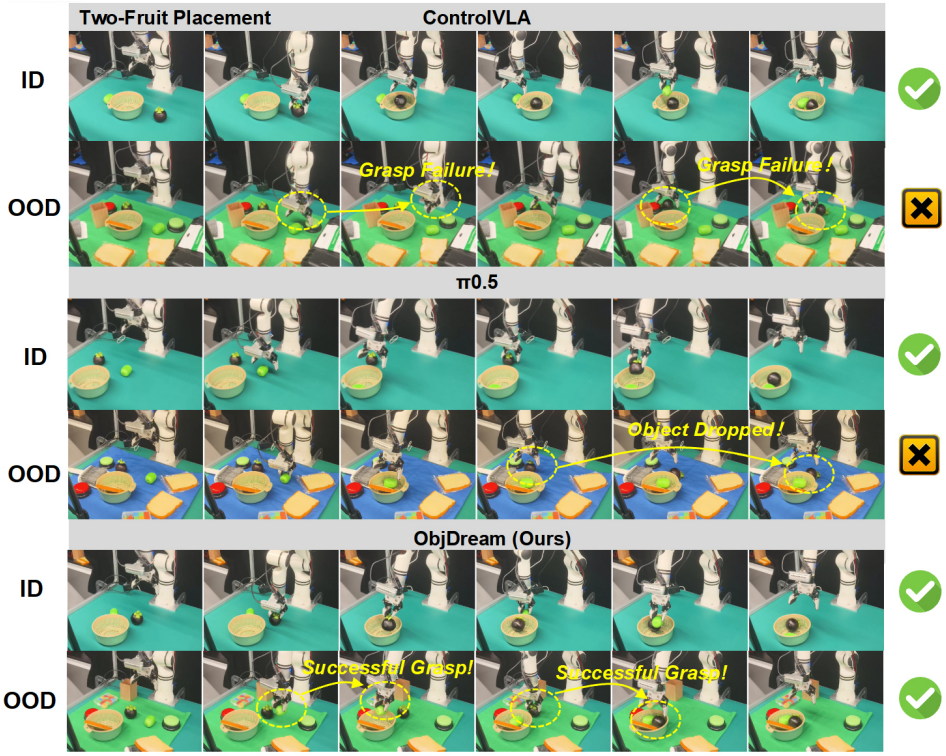


Figure 16: **Two-fruit placement.** ControlVLA, $\pi_{0.5}$, and ObjDream under ID and OOD.

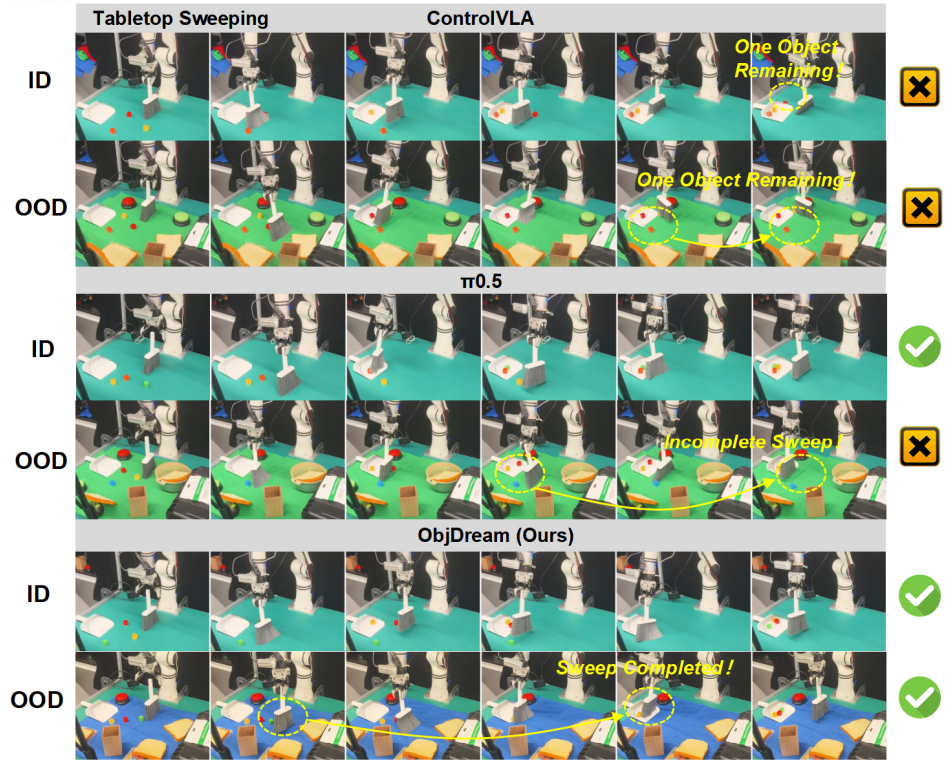


Figure 17: **Tabletop sweeping.** ControlVLA, $\pi_{0.5}$, and ObjDream under ID and OOD.



Figure 18: **Representative RoboTwin 2.0 rollouts under the Easy setting.** The visualization covers all eight tasks in the fixed evaluation subset using the benchmark's standard scene configuration. Each sequence shows representative stages of policy execution from the main camera view.



Figure 19: **Representative RoboTwin 2.0 rollouts under the Hard setting.** The same eight tasks are shown under randomized scene variations, including changes in background, clutter, lighting, and table configuration. Each sequence illustrates representative policy execution from the main camera view.

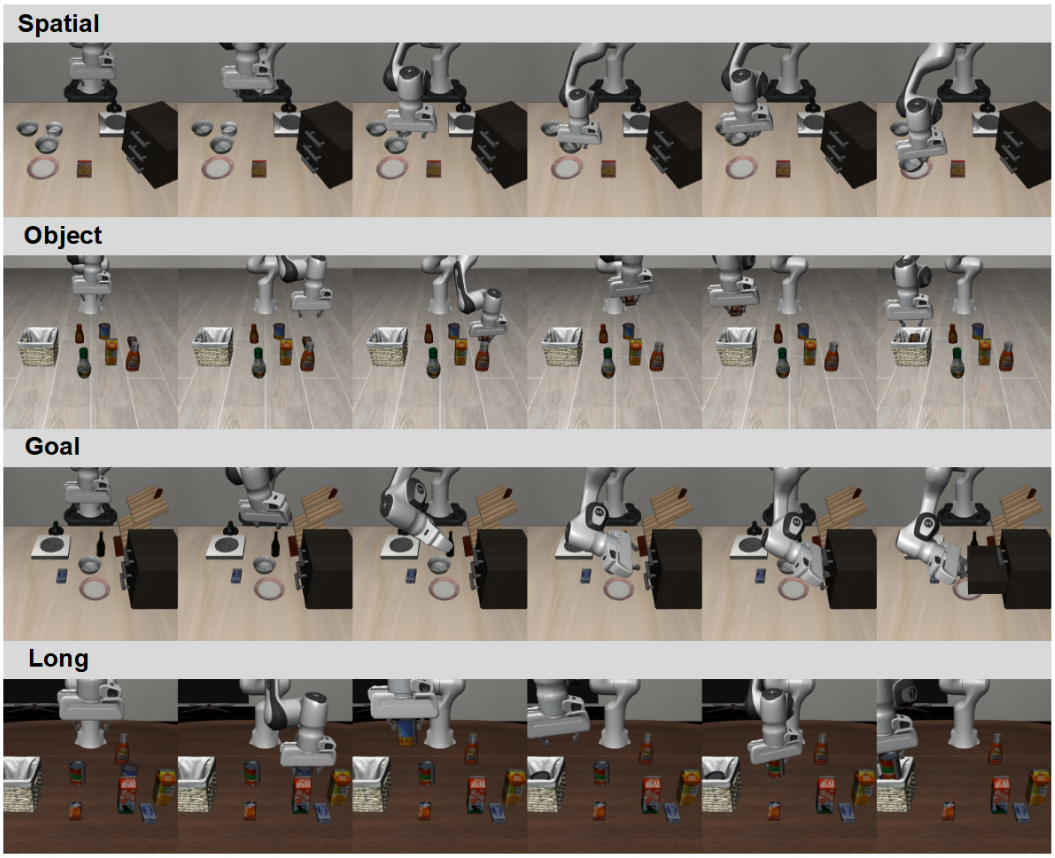


Figure 20: **Representative rollouts on standard LIBERO.** Examples are shown across the Spatial, Object, Goal, and Long suites, covering diverse language-conditioned manipulation behaviors under the standard benchmark setting.

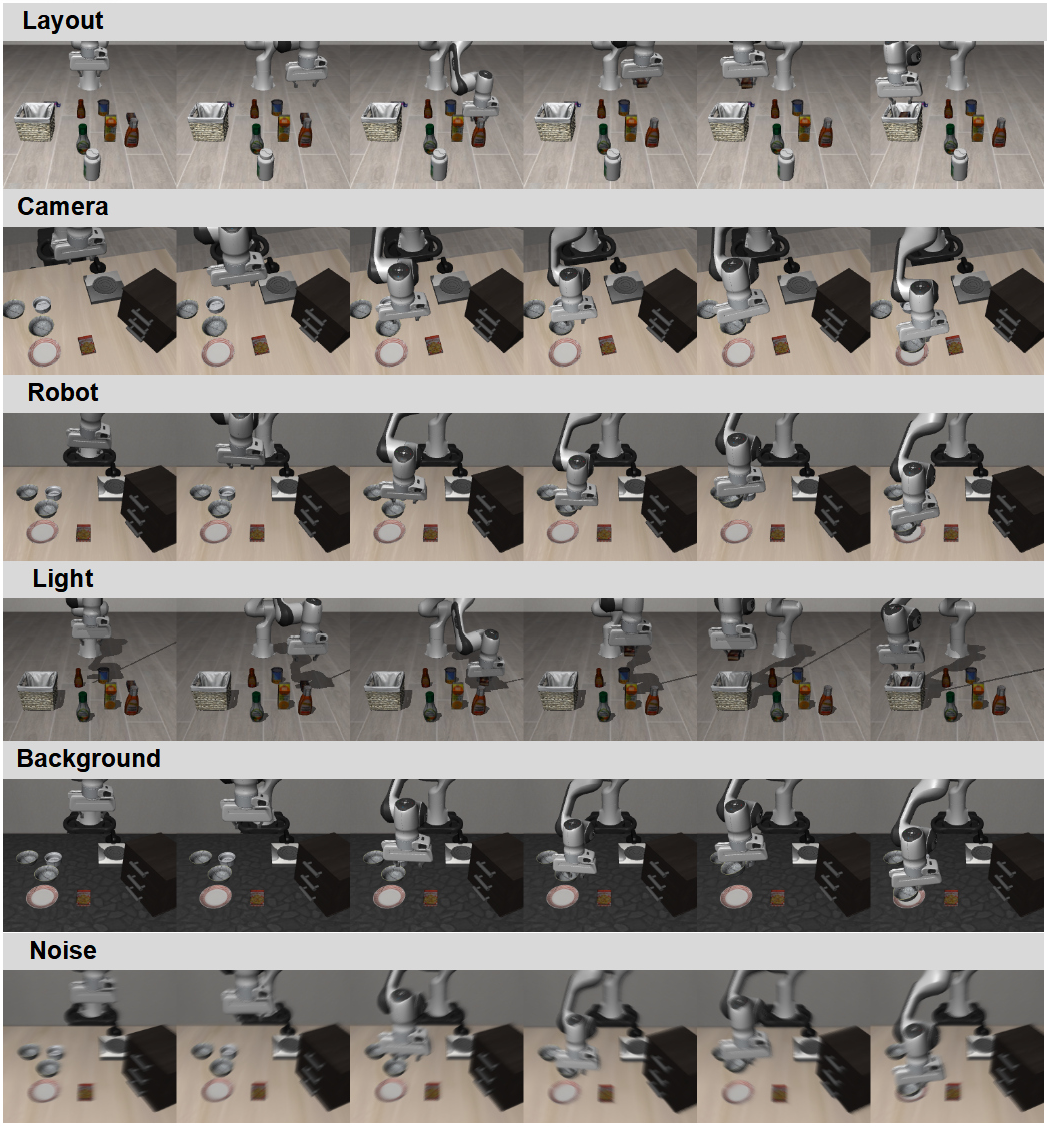


Figure 21: **Representative rollouts on LIBERO-Plus.** The examples illustrate policy execution under selected perturbations in LIBERO-Plus.

H Appendix References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 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.

Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, et al. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026.

Haonan Chen, Jingxiang Guo, Bangjun Wang, Tianrui Zhang, Xuchuan Huang, Boren Zheng, Yiwen Hou, Chenrui Tie, Jiajun Deng, and Lin Shao. Goal-vla: Image-generative vlms as object-centric world models empowering zero-shot robot manipulation. *arXiv preprint arXiv:2506.23919*, 2025a.

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*, 2025b.

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.

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 (CVPR)*, pp. 38574–38583, June 2026.

Puhao Li, Yingying Wu, Ziheng Xi, Wanlin Li, Yuzhe Huang, Zhiyuan Zhang, Yinghan Chen, Jianan Wang, Song-Chun Zhu, Tengyu Liu, and Siyuan Huang. ControlVLA: Few-Shot Object-Centric Adaptation for Pre-Trained Vision-Language-Action Models. In *Proceedings of The 9th Conference on Robot Learning*, volume 305, pp. 1898–1913, 2025a.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, et al. Hamster: Hierarchical action models for open-world robot manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 24040–24068, 2025b.

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.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pp. 38–55. Springer, 2024.

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.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Michael 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.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 8748–8763. PMLR, 2021.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, volume 2025, pp. 28085–28128, 2025.

Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*, pp. 17–35. Springer, 2016.

GigaBrain Team, Angen Ye, Axiang Sun, Can Jin, Chenxi Cheng, Chong Shi, Dengke Shang, Dingqian Zhang, Guan Huang, Guangqiang Wang, et al. Gigabrain-0.7: Scaling embodied foundation models to emergent capabilities with a three-system architecture. *arXiv preprint arXiv:2608.15875*, 2026.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, et al. Eventvla: Event-driven visual evidence memory for long-horizon vision-language-action policies. *arXiv preprint arXiv:2606.20092*, 2026.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

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.

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.