ROBOACT-WORLDVLA: REINFORCING ACTION GENERATION WITH A LATENT ACTION-UNDERSTANDING WORLD MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Although Vision–Language–Action (VLA) models have shown strong potential in robotic manipulation, their imitation-driven nature still limits robust deployment under domain shifts. The core issue is not merely insufficient demonstrations, but the lack of inherent physical action understanding and the absence of a framework that uses such understanding to guide action generation. As a result, a VLA policy may produce actions that resemble demonstrations without knowing whether these actions move the system towards the goal or not. In this regard, we propose **RoboAct-WorldVLA**, a framework that reinforces action generation with a latent action-understanding world model. Given candidate actions from a flow-based generator or an off-the-shelf VLA policy, the world model predicts their future action representations in a pretrained disentangled latent space. A language-aligned action-value evaluator assesses how well the predicted future representation aligns with the prior action representations, suppressing uncertain or off-trajectory actions. The prior action representations are generated by an existing action representation encoder RoboAct-CLIP, which disentangles the inherent action dynamics from subject/object features for better generalizability. RoboAct-WorldVLA is first pretrained offline from demonstrations, failed trajectories, and counterfactual transitions. It is then fine-tuned online with environment rewards, which refine the action generator and calibrate the evaluator while preserving imitation, latent dynamics, semantic alignment, and uncertainty regularization. We evaluate RoboAct-WorldVLA on RLBench, ManiSkill3, and real-world manipulation tasks. Extensive experiments demonstrate that RoboAct-WorldVLA can consistently boost the performance of VLA backbones by 3.8–9.0 percentage points. Notably, the ablation study reveals the proposed method's active capability to determine required causal actions and evaluate current progress, guided by prior action representations.

Keywords: Vision–Language–Action, Robot Manipulation, Action Representation, Latent World Models, Reinforcement Learning

1 INTRODUCTION

Vision–Language–Action (VLA) models have shown strong potential in language-conditioned robotic manipulation by grounding visual observations and task instructions into executable robot actions (Cai et al., 2026; Physical Intelligence et al., 2025b; Kim et al., 2025). However, robust deployment remains difficult under domain shifts, partially completed states, unseen object configurations, or execution failures. One primary reason is the lack of inherent physical action understanding in the imitation learning paradigm, which can lead to learning shortcuts under limited demonstration trajectories. Consequently, achieving generalizable action understanding and leveraging it to guide physical action generation remains a valuable yet largely underexplored problem.

More specifically, current VLA policies are still trained primarily through imitation learning (Cai et al., 2026; Physical Intelligence et al., 2025b; Kim et al., 2025; Reuss et al., 2025). Like other high-capacity robotic action generators (Wen et al., 2025; Dong et al., 2026), these policies are optimized to reproduce demonstrated actions, but their training objectives do not explicitly require them to

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.

assess what changes those actions would cause or whether these changes would help fulfill the language instruction. Consequently, a policy may generate an action that resembles a demonstration yet fails to advance the current task. This limitation is particularly pronounced in long-horizon manipulation and failure recovery, where the robot must determine whether candidate actions satisfy execution preconditions, can correct prior errors, or would further exacerbate a failure (Feng et al., 2025; Chen et al., 2025b).

A key challenge in enabling such action understanding is learning action representations that generalize across embodiments and scenes. Such representations should capture manipulation dynamics while reducing entanglement with the executor, object appearance, and environmental context. We therefore adopt RoboAct-CLIP (Zhang et al., 2025) as the action representation encoder. Pretrained on human and robot manipulation videos curated from RH20T (Fang et al., 2023) and BridgeData V2 (Walke et al., 2023), RoboAct-CLIP models temporal feature changes, disentangles subject, action, and object information, and aligns their recombined semantic representations with language. Its temporal modeling and heterogeneous pretraining encourage shared action semantics across different executors and manipulated objects. These representations provide temporal and semantic priors for downstream manipulation, helping policies generalize across embodiments, objects, and scenes. Building on these pretrained representations, we keep the encoder frozen and learn to predict the changes induced by candidate actions in its representation space, enabling subsequent evaluation of progress toward the task goal.

Leveraging action understanding during action generation is another challenge. Existing methods use world models to generate imagined rollouts for policy evaluation and refinement (Yin et al., 2026; Guo et al., 2025a; Jiang et al., 2025), infer actions from predicted visual futures (Li et al., 2026b), or support policy learning and planning through latent or geometric prediction (Zheng et al., 2025; Huang et al., 2026). However, image-generative approaches can incur substantial computational overhead and devote model capacity to appearance details unnecessary for control (Zheng et al., 2025). Furthermore, when future predictions are used only for action generation or auxiliary training, the model does not explicitly assess whether executing the proposed action would move the system toward the language-specified goal (Li et al., 2026b; Zheng et al., 2025). We therefore predict candidate-conditioned changes in pretrained, disentangled action representations and learn an evaluator to assess whether these changes advance the task, guiding action selection and policy improvement without reconstructing future images.

To address these challenges, we propose **RoboAct-WorldVLA**, a framework that reinforces VLA action generation with a latent action-understanding world model. RoboAct-WorldVLA consists of an action generator, a latent world model, and an action-value alignment evaluator. The action generator proposes candidate actions; the world model predicts the future action representation after executing each candidate; and the evaluator judges whether the predicted future representation better follows the task direction, i.e. aligning the action value based on action understanding. Candidate actions with higher action-value alignment are selected or reinforced, while uncertain or off-trajectory actions are suppressed.

This formulation also reshapes the role of reinforcement learning. Instead of applying RL as isolated online trial-and-error after imitation learning, RoboAct-WorldVLA places reward-guided improvement inside the latent world-model and evaluator framework (Xu et al., 2025; Jiang et al., 2025; Peng et al., 2026). The pretrained RoboAct-CLIP encoder (Zhang et al., 2025) remains frozen during offline training. The action generator is trained from demonstrations, the world model from successful and failed transitions, and the evaluator from successful, failed, and counterfactual candidates. Online, environment rewards refine the generator and calibrate the evaluator, while imitation, latent dynamics, semantic alignment, uncertainty, and action-regularization losses are retained to preserve stable action understanding (Xu et al., 2025; Jiang et al., 2025; Peng et al., 2026; Bu et al., 2025).

We evaluate RoboAct-WorldVLA on RLBench, ManiSkill3, and real-world manipulation (Chen et al., 2025b; Peng et al., 2026). RLBench is used to study offline pretraining followed by online reinforcement learning of a flow-based action generator (Chen et al., 2025b; Reuss et al., 2025). ManiSkill3 is used to test whether the same framework can improve multiple open-source VLA policies when they serve as action generators (Peng et al., 2026; Kim et al., 2025). JAKA and NexArm experiments further examine transfer to real-world manipulation with multi-view observations. Our contributions are summarized as follows:

- **The RoboAct-WorldVLA framework.** We propose RoboAct-WorldVLA, a framework that integrates action generation, latent consequence prediction, and action-value evaluation to guide language-conditioned manipulation and policy improvement.
- **A latent action-understanding world model.** We develop a world model that predicts candidate action consequences directly in a pretrained, disentangled latent space. By predicting future action representations and visual features, it avoids pixel-level reconstruction and provides compact, task-relevant predictions for action evaluation.
- **Action-Value-Guided Reinforcement Learning.** We develop a reinforcement learning scheme that updates the action-value evaluator with environment rewards and optimizes the action generator using the resulting value estimates. By evaluating predicted consequences against the language-specified goal, it reinforces actions that contribute to task completion.
- **Evaluation across simulation and real-world manipulation.** Experiments on RLBench, ManiSkill3, and selected real-world tasks show that RoboAct-WorldVLA improves both flow-based action generation and multiple open-source VLA backbones, with consistent gains supported by world-model and evaluator diagnostics.

2 Related Work

Vision–Language–Action Models. Recent VLA methods improve robotic manipulation through broader training data, spatial modeling, and efficient execution. $\pi_{0.5}$ combines heterogeneous training data with hierarchical subtask prediction (Physical Intelligence et al., 2025b), while SpatialVLA introduces explicit spatial representations for observation encoding and action prediction (Qu et al., 2025). OpenVLA-OFT improves action decoding and fine-tuning efficiency (Kim et al., 2025), and Xiaomi-Robotics-0 supports continuous execution through asynchronous action generation (Cai et al., 2026). While these advances strengthen action generation, their reported execution procedures do not explicitly compare alternative actions according to their predicted task progress. RoboAct-WorldVLA complements these policies by predicting and evaluating the consequences of executable action candidates before selection.

World Models for Action Generation and Planning. World models support manipulation through future-conditioned action generation and explicit planning. MoLA infers latent actions from imagined visual transitions (Li et al., 2026b), requiring visual generation before action decoding. FLARE and LaWAM instead use latent future representations to guide policy learning and action generation without reconstructing future images (Zheng et al., 2025; Chen et al., 2026). However, conditioning action generation on a predicted future does not itself compare which executable candidate would better advance the task. Explicit planners address this distinction: Cosmos Policy and τ_0-WM evaluate candidate consequences using predicted values or task progress (Kim et al., 2026; Zhou et al., 2026). Such planning can introduce substantial inference costs, as reported by Cosmos Policy. GWM-MPC performs latent planning using language–future embedding similarity, with frozen semantic alignment serving as its selection criterion (Li et al., 2026a). Our method predicts candidate-conditioned futures in the latent space of a frozen RoboAct-CLIP encoder, whose disentangled action representations are supplemented by visual features (Zhang et al., 2025). It combines a learned outcome evaluator with language-aligned progress and prediction uncertainty, avoiding future-image reconstruction while using more than embedding similarity to assess task progress.

Reinforcement Learning for VLA Policy Improvement. Recent approaches improve VLA policies through online reinforcement learning (Guo et al., 2025b; Li et al., 2025a; Chen et al., 2025a), stage-aware feedback (Xu et al., 2025), and advantage-conditioned learning (Physical Intelligence et al., 2025a). VLA-RFT further uses a learned world simulator for policy optimization (Li et al., 2025b). These methods primarily incorporate outcome information into policy training, while their reported deployment procedures generate actions from the optimized policy without explicitly predicting and ranking the consequences of multiple executable candidates. RoboAct-WorldVLA uses consequence evaluation during both training and deployment. During training, a learned evaluator assesses candidate-conditioned latent future predictions and guides the generator toward actions with better estimated outcomes. During deployment, the world model and evaluator remain active to compare newly proposed candidates and select actions according to their predicted progress toward the language-specified goal.

3 METHOD

RoboAct-WorldVLA combines an action generator, a latent world model, and a language-aligned action-value evaluator. It predicts the consequences of candidate actions before execution and uses their estimated values for both action selection and policy improvement. Figure 1 summarizes the framework. **Detailed architectures and training objectives are provided in Appendix A.**

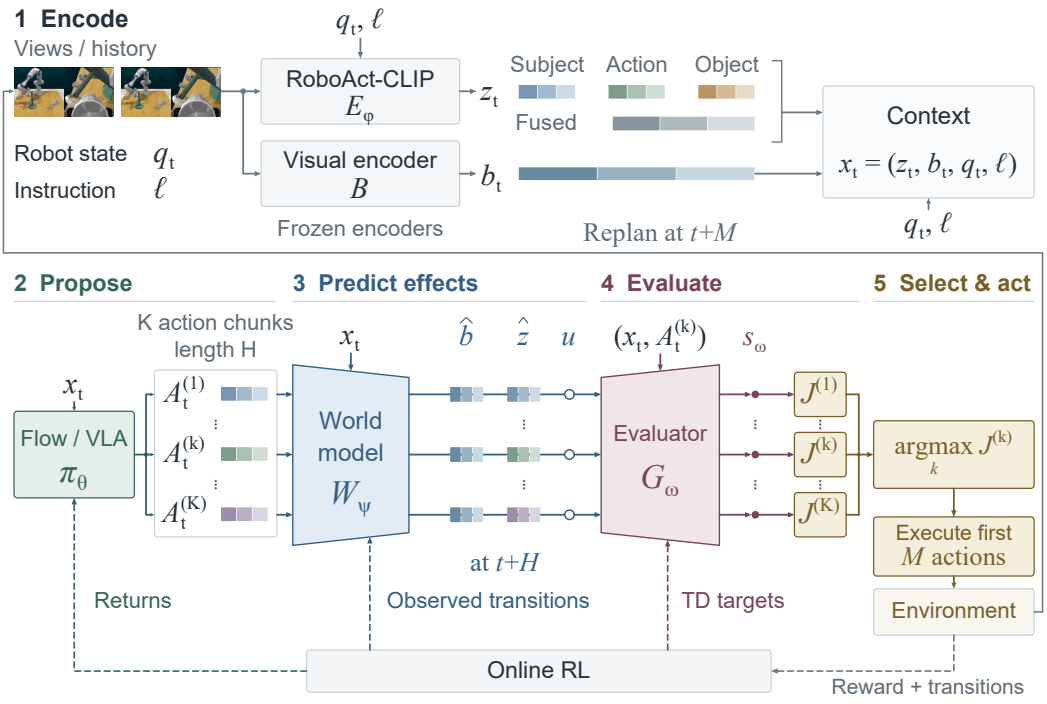


Figure 1: **Overview of RoboAct-WorldVLA.** Frozen encoders form the context. A flow policy or VLA proposes K action chunks of length H. The world model predicts terminal features and uncertainty; the evaluator ranks candidates. Only the first $M \leq H$ actions of the selected chunk are executed before replanning. Solid arrows denote inference; dashed arrows denote learning signals and parameter updates based on environment interaction during online fine-tuning. **The complete training procedure is summarized in Algorithm 1 in Appendix A.3.**

3.1 PROBLEM FORMULATION AND OVERVIEW

At time t, the robot receives external-view and wrist-view observations, its proprioceptive state q_t, and a language instruction ℓ. A frozen RoboAct-CLIP encoder (Zhang et al., 2025) extracts representations from the observation history: $z_t = \{z_t^{\text{sub}}, z_t^{\text{act}}, z_t^{\text{obj}}, z_t^{\text{fuse}}\}$. The first three representations describe the executor, action, and object; the fused representation combines visual and language information. Their disentanglement and action–text alignment are inherited from RoboAct-CLIP pretraining. A separate frozen visual encoder extracts b_t from the two camera views. These features provide the representation space for learning candidate-conditioned consequence prediction and evaluation.

At each decision step, the generator proposes candidate action chunks, the world model predicts their consequences, and the evaluator ranks them against the language-specified goal. The robot executes the selected action prefix and replans after receiving new observations.

3.2 ACTION GENERATOR

The generator proposes a fixed budget of K candidate action chunks, $A_t^{(k)} = (a_t^{(k)}, \ldots, a_{t+H-1}^{(k)}) \in \mathbb{R}^{H \times d_a}$. Every candidate is evaluated under the same budget.

We instantiate the generator as either a flow-based policy or an open-source VLA policy. The flow-based policy is trained from demonstrations. For VLA adaptation, we update LoRA (Hu et al., 2021) modules, lightweight adapters, and the action head while keeping the main backbone frozen. Each VLA uses its native observation and language inputs, while the world model and evaluator assess its action candidates in the shared latent representation space. **Generator training and adaptation objectives are detailed in Appendix A.3.**

3.3 Latent Action-Understanding World Model

The world model learns what changes may result from executing a candidate action, providing predicted consequences for evaluating progress toward the task goal. For an action prefix $A_{t,h}^{(k)} = (a_t^{(k)}, \ldots, a_{t+h-1}^{(k)})$, it predicts

$$(\hat{b}_{t+h}^{(k)}, \hat{z}_{t+h}^{(k)}, u_h^{(k)}) = W_\psi(z_t, b_t, q_t, A_{t,h}^{(k)}, \ell, h), \tag{1}$$

where $\hat{b}_{t+h}^{(k)}$ and $\hat{z}_{t+h}^{(k)}$ are future visual and RoboAct-CLIP features, and $u_h^{(k)}$ represents prediction uncertainty. Each query predicts one endpoint. Planning uses $h = H$ to evaluate the complete candidate chunk. Optional online updates use the actually executed prefix, with $h = m \leq M$ for a maximum execution-prefix length $M \leq H$, and targets from the corresponding observed endpoint.

Prediction targets are extracted from future observations using the frozen encoders. Training includes both successful and failed transitions to cover task progress and unsuccessful action effects. Predicting in this feature space provides candidate-specific consequences without reconstructing future images. **The architecture, uncertainty estimation, and prediction losses are described in Appendix A.1.**

3.4 Action-Value-Guided Reinforcement Learning

For each candidate, the trainable evaluator G_ω uses the current and predicted features, candidate action, robot state, language instruction, prediction horizon, and uncertainty to produce a base score $s_\omega^{(k)}$. The final action value is

$$J^{(k)} = s_\omega^{(k)} + \lambda_{\text{sem}} \widetilde{\Delta}_{\text{sem}}^{(k)} + \lambda_{\text{feas}} s_{\text{feas}}^{(k)} - \lambda_u u^{(k)} - \lambda_c C(A_t^{(k)}), \tag{2}$$

where $u^{(k)} = u_H^{(k)}$ during planning. The semantic term measures the change in alignment with the language goal from the current to the predicted future representation, with uncertain predictions downweighted. The remaining terms account for action feasibility, prediction uncertainty, and action magnitude or smoothness. The robot selects $k^\star = \arg\max_k J^{(k)}$, executes a prefix of at most $M \leq H$ actions, and replans. **The evaluator architecture, semantic alignment, and score components are detailed in Appendix A.2.**

Offline training uses successful, failed, and counterfactual candidates to learn action evaluation. Online training uses rewards from environment interaction to refine the evaluator and improve the policy. Let m denote the number of actions actually executed. For a nonterminal transition ending at $t + m$, the evaluator uses the temporal-difference target

$$y_t = r_t + \gamma \max_j J_{t+m}^{(j)}, \qquad \mathcal{L}_{\text{eval}}^{\text{RL}} = \left(J_t^{\text{sel}} - \text{sg}(y_t)\right)^2, \tag{3}$$

where J_t^{sel} is the selected candidate's value and sg denotes stop-gradient.

The resulting value estimates provide advantages for policy improvement. The flow generator uses advantage-weighted flow matching; VLA policies update the same LoRA modules, adapters, and action head through policy gradients when action log-probabilities are available, or advantage-weighted action regression otherwise. This connects policy updates to the evaluated consequences of candidate actions. The world model and evaluator remain active during deployment to assess newly generated candidates. **Reward construction and the complete policy optimization objectives are provided in Appendix A.3.**

Table 1: RLBench success rates (%). Avg. covers the ten listed tasks.

Model	Avg.	Insert Peg	Place Cups	Place Wine	Put in Cup.	Screw Bulb	Sort Shape	Stack Blocks	Stack Cups	Sweep Dustpan	Turn Tap
PerAct (Shridhar et al., 2023)	28.4	$5.6_{\pm 4.1}$	$2.4_{\pm 3.2}$	$44.8_{\pm 7.8}$	$28.0_{\pm 4.4}$	$17.6_{\pm 2.0}$	$16.8_{\pm 4.7}$	$26.4_{\pm 3.2}$	$2.4_{\pm 2.0}$	$52.0_{\pm 0.0}$	$88.0_{\pm 4.4}$
RVT (Goyal et al., 2023)	46.1	$11.2_{\pm 3.0}$	$4.0_{\pm 2.5}$	$91.0_{\pm 5.2}$	$49.6_{\pm 3.2}$	$48.0_{\pm 5.7}$	$36.0_{\pm 2.5}$	$28.8_{\pm 3.9}$	$26.4_{\pm 8.2}$	$72.0_{\pm 0.0}$	$93.6_{\pm 4.1}$
RVT-2 (Goyal et al., 2024)	71.0	$40.0_{\pm 0.0}$	$38.0_{\pm 4.5}$	$95.0_{\pm 3.3}$	$66.0_{\pm 4.5}$	$88.0_{\pm 4.9}$	$35.0_{\pm 7.1}$	$\mathbf{80.0}_{\pm 2.8}$	$69.0_{\pm 5.9}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{99.0}_{\pm 1.7}$
SAM2ACT (Fang et al., 2025)	80.1	$84.0_{\pm 5.7}$	$47.0_{\pm 6.0}$	$93.0_{\pm 3.8}$	$75.0_{\pm 3.8}$	$89.0_{\pm 2.0}$	$64.0_{\pm 4.6}$	$76.0_{\pm 8.6}$	$78.0_{\pm 4.0}$	$99.0_{\pm 2.0}$	$96.0_{\pm 5.7}$
BridgeVLA (Li et al., 2025c)	79.4	$88.0_{\pm 2.8}$	$58.4_{\pm 10.0}$	$88.0_{\pm 2.8}$	$73.6_{\pm 4.6}$	$87.2_{\pm 6.6}$	$60.8_{\pm 7.7}$	$76.8_{\pm 8.7}$	$81.6_{\pm 3.6}$	$87.2_{\pm 1.8}$	$92.8_{\pm 3.3}$
TGM-VLA (Pu et al., 2026)	83.8	$94.0_{\pm 2.3}$	$60.0_{\pm 7.3}$	$94.0_{\pm 2.3}$	$83.0_{\pm 2.0}$	$89.0_{\pm 3.8}$	$80.0_{\pm 5.7}$	$70.0_{\pm 4.0}$	$87.0_{\pm 2.0}$	$88.0_{\pm 4.6}$	$93.0_{\pm 3.8}$
RoboAct-WorldVLA	**86.0**	$\mathbf{95.0}_{\pm 0.9}$	$\mathbf{64.0}_{\pm 2.9}$	$\mathbf{95.3}_{\pm 2.6}$	$\mathbf{85.3}_{\pm 5.3}$	$\mathbf{90.3}_{\pm 3.1}$	$\mathbf{83.0}_{\pm 1.4}$	$73.3_{\pm 3.1}$	$\mathbf{89.0}_{\pm 4.1}$	$90.3_{\pm 2.8}$	$94.3_{\pm 1.8}$

Table 2: **ManiSkill results.** Success rates (%) before and after adding RoboAct-WorldVLA.

Method	Pick Cube	Stack Cube	Pick Single YCB	Pick Single EGAD	Pick Clutter YCB	Avg.	Δ Avg.
OpenVLA (Kim et al., 2024)	65	55	0	15	0	27	–
OpenVLA + RoboAct-WorldVLA	68	59	8	23	6	33	+6
Dita (Hou et al., 2025)	79	80	62	72	36	66	–
Dita + RoboAct-WorldVLA	82	83	66	76	42	70	+4
CogACT (Li et al., 2024)	95	90	65	75	25	70	–
CogACT + RoboAct-WorldVLA	96	92	69	79	32	74	+4

4 Experiments and Mechanism Analysis

We evaluate **RoboAct-WorldVLA** on RLBench using the full offline-to-online pipeline with a flow-based action generator, on ManiSkill3 to assess whether the framework improves the performance of open-source VLA policies, and on JAKA and NexArm for real-world manipulation with multi-view observations. Unless otherwise specified, RoboAct-WorldVLA denotes the full training and fixed-budget candidate-selection pipeline described in Section 3.

RLBench (James et al., 2020). We evaluate RoboAct-WorldVLA on ten RLBench tasks using a flow-based action generator. The task suite covers precision insertion, multi-object arrangement, articulated-object interaction, sorting, stacking, and long-horizon placement. We evaluate our method using four random seeds, with 100 evaluation episodes per task for each seed, and report the mean and standard deviation of success rates across seeds (Table 1).

ManiSkill3 (Tao et al., 2025). We evaluate whether RoboAct-WorldVLA can be used with different open-source VLA generators. Each VLA backbone serves as the action generator, while the latent world model and action-value evaluator are used as the shared consequence-aware decision layer. Table 2 reports success rates before and after adding WorldVLA. **See Appendix E.2 for qualitative rollout visualizations.**

Real-world experiments. We evaluate RoboAct-WorldVLA on three EFM-10 (He et al., 2026) tasks using the dual-arm JAKA platform: *Toy-Match* for semantic grounding, *Cup-Hang* for bimanual manipulation under occlusion, and *Box-Push* for contact-rich action-effect prediction (Table 3a). On NexArm, we evaluate placing specified fruit or vegetables into a basket, stacking blocks, pushing a block into a target region, and hanging a cup (Table 3b). Training data for NexArm were collected through teleoperation. **Appendix D details both platforms; Appendices E.3 and E.4 show qualitative task sequences on JAKA and NexArm, respectively.**

Real-world future-feature prediction. The supplementary video also examines world-model predictions during real robot manipulation. In the illustrated JAKA Cup-Hang sample, the cosine similarity between predicted and observed future features at $h = 10$ is **0.846**, compared with **0.742** for copy-current, indicating that the world model better captures the subsequent feature change in this example. **See the supplementary video for multi-view observations, comparisons across prediction horizons, and additional real-world examples.**

Table 3: **Real-world results.** Success rates (%) on JAKA and NexArm. Δ is the mean task-wise gain.

(a) JAKA

Method	Toy-Match	Cup-Hang	Box-Push	Avg. SR	Δ Avg.
OpenVLA (Kim et al., 2024)	48.0	57.0	20.0	41.7	–
OpenVLA + RoboAct-WorldVLA	57.0	67.0	28.0	50.7	+9.0
Dita (Hou et al., 2025)	62.0	70.0	32.0	54.7	–
Dita + RoboAct-WorldVLA	68.0	78.0	38.0	61.3	+6.7
CogACT (Li et al., 2024)	67.0	77.0	35.0	59.7	–
CogACT + RoboAct-WorldVLA	73.0	85.0	43.0	67.0	+7.3

(b) NexArm

Method	Pick-and-place	Stack Blocks	Push Block	Cup Hang	Avg. SR	Δ Avg.
OpenVLA	64.0	42.0	75.0	42.0	55.8	–
OpenVLA + RoboAct-WorldVLA	66.0	56.0	78.0	47.0	61.8	+6.0
Dita	71.0	57.0	85.0	50.0	65.8	–
Dita + RoboAct-WorldVLA	73.0	64.0	87.0	60.0	71.0	+5.3
CogACT	75.0	63.0	90.0	57.0	71.3	–
CogACT + RoboAct-WorldVLA	78.0	67.0	92.0	65.0	75.5	+4.3

Table 4: **Ablation study.** Four-task RLBench success rates (%); $\pm$ gives the average sample SD of two four-seed groups (100 episodes/task/seed). Latency is relative to RoboAct-WorldVLA ($K = 6$; 100%).

Variant	Place Cups	Stack Blocks	Sort Shape	Put in Cup.	**Avg.**	K	Rel. latency
A. Candidate selection							
Direct execution (Encoder + policy)	$56.3_{\pm 3.6}$	$64.8_{\pm 3.6}$	$74.3_{\pm 4.5}$	$77.8_{\pm 2.3}$	$68.3_{\pm 1.9}$	1	50.7
Random selection (6 candidates)	$57.3_{\pm 3.8}$	$65.8_{\pm 5.7}$	$75.3_{\pm 4.6}$	$78.8_{\pm 5.3}$	$69.3_{\pm 2.2}$	6	98.0
Evaluator-only (AC-style RL)	$56.5_{\pm 4.7}$	$65.0_{\pm 3.7}$	$74.5_{\pm 3.6}$	$78.0_{\pm 2.8}$	$68.5_{\pm 1.8}$	6	75.0
B. System training, data, and components							
Without online fine-tune	$60.3_{\pm 6.7}$	$69.7_{\pm 2.8}$	$78.3_{\pm 2.4}$	$81.7_{\pm 1.6}$	$72.5_{\pm 2.4}$	6	98.9
Success-only data	$58.7_{\pm 5.0}$	$67.3_{\pm 2.3}$	$76.7_{\pm 3.6}$	$80.0_{\pm 4.4}$	$70.7_{\pm 2.1}$	6	100.6
Fused-feature WM (w/o RoboAct-CLIP)	$57.3_{\pm 4.2}$	$66.0_{\pm 4.1}$	$75.7_{\pm 5.5}$	$79.0_{\pm 7.0}$	$69.5_{\pm 2.3}$	6	96.6
Non-aligned evaluator	$58.0_{\pm 4.7}$	$66.7_{\pm 2.5}$	$76.3_{\pm 4.1}$	$79.7_{\pm 2.3}$	$70.2_{\pm 2.1}$	6	95.4
Without uncertainty scoring	$60.7_{\pm 2.5}$	$69.0_{\pm 6.5}$	$78.7_{\pm 7.5}$	$82.0_{\pm 3.0}$	$72.6_{\pm 2.0}$	6	93.1
C. Candidate budget							
RoboAct-WorldVLA ($K = 1$)	$56.7_{\pm 3.9}$	$65.3_{\pm 5.9}$	$74.7_{\pm 4.0}$	$78.3_{\pm 5.3}$	$68.8_{\pm 2.9}$	1	60.0
RoboAct-WorldVLA ($K = 3$)	$61.0_{\pm 5.3}$	$70.0_{\pm 1.6}$	$79.0_{\pm 3.1}$	$82.0_{\pm 2.4}$	$73.0_{\pm 2.3}$	3	70.1
RoboAct-WorldVLA ($K = 12$)	$64.3_{\pm 2.8}$	$73.7_{\pm 5.7}$	$83.3_{\pm 6.5}$	$85.7_{\pm 2.4}$	$76.8_{\pm 3.0}$	12	151.7
RoboAct-WorldVLA	$\mathbf{64.0_{\pm 2.9}}$	$\mathbf{73.3_{\pm 3.1}}$	$\mathbf{83.0_{\pm 1.4}}$	$\mathbf{85.3_{\pm 5.3}}$	$\mathbf{76.4_{\pm 1.2}}$	**6**	**100.0**

Ablation settings (Table 4). *Direct execution (Encoder + policy)* removes the world model and evaluator and executes the action generated by the policy. *Random selection (6 candidates)* runs the full candidate generation, world-model prediction, and evaluation pipeline, but selects uniformly among six candidates without using their scores. *Evaluator-only (AC-style RL)* directly scores candidates with a separately trained network, analogous to the critic in actor–critic (AC) RL. Its inputs are the current representations, robot state, language instruction, and candidate action. It uses no world model, predicted future features, or prediction-derived semantic and uncertainty signals, and is trained using the same candidate data and outcome labels. *Without online fine-tune* removes online reward-guided fine-tuning from the full system. *Success-only data* removes failed trajectories and counterfactual candidates from world-model and evaluator training. *Fused-feature world model (w/o RoboAct-CLIP)* replaces the RoboAct-CLIP representations with a single fused latent feature. *Non-aligned evaluator* replaces frozen action–text aligned projections with an MLP over unaligned visual/action features and language embeddings. *Without uncertainty scoring* excludes uncertainty from candidate evaluation. The K-sweep changes only the number of evaluated candidates; the $K = 1$ configuration still runs prediction and scoring. RoboAct-WorldVLA with $K = 6$ is the common reference for all groups.

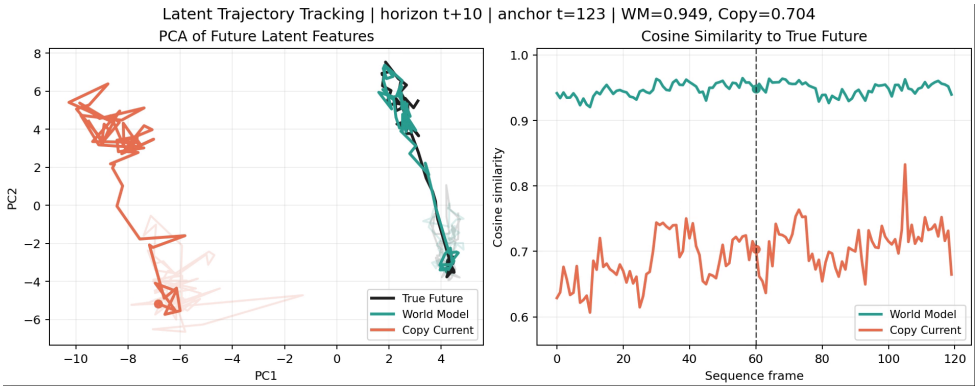


Figure 2: **Future-feature prediction at $h = 10$.** Left: PCA trajectories of true future features, world-model predictions, and copy-current. Right: cosine similarity to the true future features along the visualization sequence.

4.1 Result Analysis

Overall performance. On RLBench, RoboAct-WorldVLA reaches a ten-task average success rate of **86.0%**, outperforming TGM-VLA by **2.2** percentage points and leading on seven of ten tasks. On ManiSkill3, adding WorldVLA improves OpenVLA, Dita, and CogACT by **5.8**, **4.2**, and **3.8** percentage points on average, respectively. On JAKA, the mean task-wise gains are **9.0**, **6.7**, and **7.3** percentage points for OpenVLA, Dita, and CogACT, respectively; the corresponding gains on NexArm are **6.0**, **5.3**, and **4.3** points.

Overall system benefit. Direct execution achieves an average success rate of **68.3%**, compared with **76.4%** for RoboAct-WorldVLA, corresponding to an overall improvement of **8.1** percentage points over direct policy execution.

Candidate evaluation and prediction. Evaluator-only and random selection achieve average success rates of **68.5%** and **69.3%**, respectively, compared with **76.4%** for RoboAct-WorldVLA (Table 4). Evaluator-only performs **0.8** percentage points below random selection, while RoboAct-WorldVLA exceeds evaluator-only by **7.9** percentage points. Comparing evaluator-only with RoboAct-WorldVLA tests whether predicted consequences and their associated semantic and uncertainty signals improve candidate selection. Random selection controls for running the candidate-processing pipeline without using its scores to choose actions.

Component ablations and candidate budget. The configuration without online fine-tuning achieves **72.5%** average success (Table 4). The fused-feature world model and non-aligned evaluator achieve **69.5%** and **70.2%**, respectively, supporting the usefulness of the chosen representation and evaluator designs within this framework. Training only with successful data or removing uncertainty scoring also reduces performance. The candidate-budget sweep further shows a success–latency trade-off: $K = 6$ reaches **76.4%**, whereas $K = 12$ gives only a marginal gain to **76.8%** but increases relative latency from **100.0%** to **151.7%**. We therefore use $K = 6$ as the default setting.

See Appendix F for limitations.

4.2 World-Model Prediction and Action Evaluation

Future-feature prediction. We first examine whether the world model predicts feature changes after action execution. On the RLBench Close Jar validation data, cosine similarity to the true future features reaches **0.9537** and **0.9455** at $h = 2$ and $h = 10$, respectively, compared with **0.7278** and **0.6921** for copying the current features. In the visualization sequence, the predicted latent trajectory follows the changes in the true future features more closely than copy-current (Figure 2). These results show that the world model captures feature changes during execution in the frozen representation space, providing a basis for evaluating candidate consequences. **See Appendix B and the supplementary videos for dual-view feature comparisons, prediction-horizon analysis, and dynamic visualizations.**

Candidate action evaluation. The evaluator achieves **97.67%** classification accuracy in distinguishing task-progressing trajectories from failed, wrong-direction, or counterfactual trajectories. We further examine the relationship between candidate scores and execution outcomes during placement in RLBench Place Cups (Figure 3). Under the same observation, the correct-placement candidate and a wrong-placement candidate with its target shifted along negative y receive logged diagnostic scores $J_{\log}$ of **0.8001** and **0.7946**, respectively. Their corresponding rollouts show successful placement and incorrect placement ending in failure, so the score ordering agrees with the task outcomes. This example illustrates why action consequences matter: an executable motion can continue moving the cup while deviating from the language-specified goal. **See Appendix C and the supplementary video for candidate construction, diagnostic-score definitions, full rollouts, and world-model prediction analysis; additional RLBench rollouts are provided in Appendix E.1.**

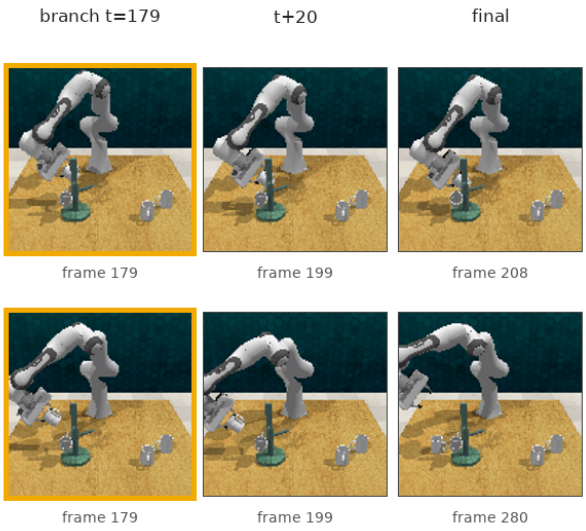


Figure 3: **Candidate evaluation in RLBench Place Cups.** Top: selected candidate ($J_{\log} = 0.8001$), followed by successful placement. Bottom: rejected candidate ($J_{\log} = 0.7946$), followed by incorrect placement. Candidates are scored under the same observation; panels show frames from the corresponding rollouts at $t = 179$, $t + 20$, and their final steps.

5 Conclusion

In this paper, we presented **RoboAct-WorldVLA**, a consequence-aware VLA framework that guides action generation with a latent action-understanding world model. By predicting the future latent effects of candidate actions and evaluating them with a language-aligned action-value scorer, RoboAct-WorldVLA selects actions based on anticipated task progress rather than imitation likelihood alone. Experiments on RLBench, ManiSkill3, and real-world manipulation show consistent improvements across flow-based and open-source VLA generators. These results demonstrate that explicit latent action understanding can effectively enhance the robustness and generalization of VLA-based robotic manipulation.

AI Use Statement

We used generative AI tools to assist with language polishing and translation of the manuscript, and AI-assisted coding ("vibe coding") to help develop the experimental code. The authors take full responsibility for the final content of the paper and the experimental code.

REFERENCES

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.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, Yuanbo Xu, and Chao Yu. LaWAM: Latent world action models for efficient dynamics-aware robot policies. *arXiv preprint arXiv:2606.15768*, 2026. URL https://arxiv.org/abs/2606.15768.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online RL fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025a. URL https://arxiv.org/abs/2510.25889.

Zixuan Chen, Jing Huo, Yangtao Chen, and Yang Gao. Robohorizon: An llm-assisted multi-view world model for long-horizon robotic manipulation. *arXiv preprint arXiv:2501.06605*, 2025b.

Zhenchen Dong, Jinna Fu, Jiaming Wu, Shengyuan Yu, Fulin Chen, and Yide Liu. Hybridflow: A two-step generative policy for robotic manipulation. *arXiv preprint arXiv:2602.13718*, 2026.

Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. RH20T: A comprehensive robotic dataset for learning diverse skills in one-shot. *arXiv preprint arXiv:2307.00595*, 2023. URL https://arxiv.org/abs/2307.00595.

Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation. *arXiv preprint arXiv:2501.18564*, 2025.

Yunhai Feng, Jiaming Han, Zhuoran Yang, Xiangyu Yue, Sergey Levine, and Jianlan Luo. Reflective planning: Vision-language models for multi-stage long-horizon robotic manipulation. *arXiv preprint arXiv:2502.16707*, 2025.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pp. 694–710. PMLR, 2023.

Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. Rvt-2: Learning precise manipulation from few demonstrations. *arXiv preprint arXiv:2406.08545*, 2024.

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-world: A controllable generative world model for robot manipulation. *arXiv preprint arXiv:2510.10125*, 2025a.

Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15665–15672. IEEE, 2025b.

Yuxin He, Ruihao Zhang, Tianao Shen, Cheng Liu, and Qiang Nie. Towards exploratory and focused manipulation with bimanual active perception: A new problem, benchmark and strategy. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026. URL https://arxiv.org/abs/2602.01939.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7686–7697, 2025.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. URL https://arxiv.org/abs/2106.09685.

Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. Pointworld: Scaling 3d world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782*, 2026.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020. URL https://arxiv.org/abs/1909.12271.

Zhennan Jiang, Kai Liu, Yuxin Qin, Shuai Tian, Yupeng Zheng, Mingcai Zhou, Chao Yu, Haoran Li, and Dongbin Zhao. World4rl: Diffusion world models for policy refinement with reinforcement learning for robotic manipulation. *arXiv preprint arXiv:2509.19080*, 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.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026. URL https://arxiv.org/abs/2601.16163.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. SimpleVLA-RL: Scaling VLA training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025a. URL https://arxiv.org/abs/2509.09674.

Hengtao Li, Pengxiang Ding, Runze Suo, Yihao Wang, Zirui Ge, Dongyuan Zang, Kexian Yu, Mingyang Sun, Hongyin Zhang, Donglin Wang, and Weihua Su. VLA-RFT: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators. *arXiv preprint arXiv:2510.00406*, 2025b. URL https://arxiv.org/abs/2510.00406.

Peiyan Li, Yixiang Chen, Hongtao Wu, Xiao Ma, Xiangnan Wu, Yan Huang, Liang Wang, Tao Kong, and Tieniu Tan. Bridgevla: Input-output alignment for efficient 3d manipulation learning with vision-language models. *Advances in Neural Information Processing Systems*, 38:63635–63673, 2025c.

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

Quanyi Li, Lan Feng, Haonan Zhang, Wuyang Li, Letian Wang, Alexandre Alahi, and Harold Soh. Grounded world model for semantically generalizable planning. *arXiv preprint arXiv:2604.11751*, 2026a. URL https://arxiv.org/abs/2604.11751.

Yajie Li, Bozhou Zhang, Chun Gu, Zipei Ma, Jiahui Zhang, Jiankang Deng, Xiatian Zhu, and Li Zhang. From imagined futures to executable actions: Mixture of latent actions for robot manipulation. *arXiv preprint arXiv:2605.12167*, 2026b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. URL https://arxiv.org/abs/2210.02747.

Yuanfang Peng, Jingjing Fu, Chuheng Zhang, Li Zhao, Jiang Bian, Mingyu Liu, Ling Zhang, Jun Zhang, and Rui Wang. What to ignore, what to react: Visually robust rl fine-tuning of vla models. *arXiv preprint arXiv:2605.13105*, 2026.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, et al. $\pi^*_{0.6}$: a VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a. URL https://arxiv.org/abs/2511.14759.

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*, 2025b.

Fanqi Pu, Lei Jiang, and Wenming Yang. TGM-VLA: Task-guided mixup for sampling-efficient and robust robotic manipulation. *arXiv preprint arXiv:2603.00615*, 2026. URL https://arxiv.org/abs/2603.00615.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring spatial representations for Visual-Language-Action model. *arXiv preprint arXiv:2501.15830*, 2025. doi: 10.48550/arXiv.2501.15830. URL https://arxiv.org/abs/2501.15830.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://www.jmlr.org/papers/v21/20-074.html.

Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. FLOWER: Democratizing generalist robot policies with efficient vision-language-flow models. In *Proceedings of The 9th Conference on Robot Learning*, 2025.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pp. 785–799. PMLR, 2023.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. ManiSkill3: GPU parallelized robotics simulation and rendering for generalizable embodied AI. *Robotics: Science and Systems*, 2025. URL https://arxiv.org/abs/2410.00425.

Homer Rich Walke, Kevin Black, Tony Z. Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A dataset for robot learning at scale. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1723–1736. PMLR, 2023. URL https://proceedings.mlr.press/v229/walke23a.html.

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

Feng Xu, Guangyao Zhai, Xin Kong, Tingzhong Fu, Daniel FN Gordon, Xueli An, and Benjamin Busam. STARE-VLA: Progressive stage-aware reinforcement for fine-tuning vision-language-action models. *arXiv preprint arXiv:2512.05107*, 2025.

Tenny Yin, Zhiting Mei, Zhonghe Zheng, Miyu Yamane, David Wang, Jade Sceats, Samuel M Bateman, Lihan Zha, Apurva Badithela, Ola Shorinwa, et al. Playworld: Learning robot world models from autonomous play. *arXiv preprint arXiv:2603.09030*, 2026.

Zhiyuan Zhang, Yuxin He, Yong Sun, Junyu Shi, Lijiang Liu, and Qiang Nie. RoboAct-CLIP: Video-driven pre-training of atomic action understanding for robotics. *arXiv preprint arXiv:2504.02069*, 2025. URL https://arxiv.org/abs/2504.02069.

Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, et al. FLARE: Robot learning with implicit world modeling. *arXiv preprint arXiv:2505.15659*, 2025.

Pengfei Zhou, Shengcong Chen, Di Chen, Jiaxu Wang, Rongjun Jin, Bingwen Zhu, Yike Pan, Songen Gu, Kuanning Wang, Shufeng Nan, Xingyu Qiu, Chenhao Qiu, Pu Yang, Yunuo Cai, Jianxiong Gao, Yifan Li, Yanwei Fu, Xiangyu Yue, Zhi Chen, and Jianlan Luo. τ_0-WM: A unified video-action world model for robotic manipulation. *arXiv preprint arXiv:2606.01027*, 2026. URL https://arxiv.org/abs/2606.01027.

A Architectures and Training Details

This appendix details the latent world model W_ψ, the language-aligned evaluator G_ω, and the generator optimization used in Fig. 1. It complements the inference and reinforcement-learning procedure in Section 3.

A.1 World Model Architecture and Training

Frozen representations. For observation history $o_{t-L+1:t}$ and instruction ℓ, we extract

$$z_t = E_\phi(o_{t-L+1:t}, \ell), \qquad b_t = B(I_t^{\text{ext}}, I_t^{\text{wrist}}), \tag{4}$$

where $o_t = \{I_t^{\text{ext}}, I_t^{\text{wrist}}, q_t\}$. The RoboAct-CLIP encoder E_ϕ and visual encoder B remain frozen. The world model operates on six feature types: subject, action, object, fused, external-view visual, and wrist-view visual features. The fused representation combines vision and language information; it is not a fourth independent semantic factor.

Candidate-conditioned endpoint prediction. Figure 4 shows the architecture. Fixed feature normalization and type-specific projections convert the current features into typed tokens. A causal action-prefix encoder represents the ordered actions $A_{t,h}$. Together with robot state q_t, frozen T5 (Raffel et al., 2020) text features, and the prefix length h, it conditions Transformer self-attention blocks through adaptive layer normalization (AdaLN). Type-specific residual Gaussian heads predict means and log variances in the normalized feature spaces, using the current normalized features as the residual reference. Each call to Eq. (1) predicts the endpoint of the supplied action prefix. Planning uses $A_{t,H} = A_t$ and $h = H$; online feature supervision uses the actually executed prefix and its observed endpoint.

Ensemble prediction and uncertainty. Independently trained predictors produce ensemble means and total variance. For a given normalized feature type, let μ_e and σ_e^2 denote the mean and variance from ensemble member e, and let E be the number of members. The ensemble combines within-member variance and disagreement between member means:

$$\bar{\mu} = \frac{1}{E}\sum_{e=1}^{E}\mu_e, \qquad v_{\text{total}} = \frac{1}{E}\sum_{e=1}^{E}\sigma_e^2 + \frac{1}{E}\sum_{e=1}^{E}(\mu_e - \bar{\mu})^2. \tag{5}$$

The variance expression is applied elementwise. Fixed normalization is inverted to return the predicted features to the frozen encoders' feature spaces. Held-out variance scaling produces the scalar uncertainty u_h used for candidate evaluation.

Offline world-model objective. Training uses both successful and failed transitions, $\mathcal{D}_{\text{wm}} = \mathcal{D}_{\text{succ}} \cup \mathcal{D}_{\text{fail}}$. Successful trajectories describe task progress, while failed trajectories provide no-progress, missed-grasp, collision, object-drop, and wrong-direction transitions. For a prefix of length h, the prediction targets are extracted from the corresponding observed future state using the frozen encoders. The loss is

$$\begin{aligned}\mathcal{L}_{\text{wm}} = \mathbb{E}_{\mathcal{D}_{\text{wm}}}\big[&\lambda_b \underbrace{\mathcal{D}(\hat{b}_{t+h}, \text{sg}(b_{t+h}))}_{\mathcal{L}_{\text{vis}}} + \lambda_z \underbrace{\mathcal{D}(\hat{z}_{t+h}, \text{sg}(z_{t+h}))}_{\mathcal{L}_{\text{actrep}}} \\ &+ \lambda_\Delta \underbrace{\mathcal{D}(\hat{z}_{t+h} - z_t, \text{sg}(z_{t+h} - z_t))}_{\mathcal{L}_\Delta} + \lambda_u \underbrace{\mathcal{L}_{\text{unc}}(\hat{b}_{t+h}, b_{t+h}, u_h)}_{\mathcal{L}_{\text{unc}}}\big].\end{aligned} \tag{6}$$

Here $\mathcal{D}$ combines cosine distance and regression loss, and sg denotes stop-gradient. The four terms supervise future visual prediction, future action-representation prediction, action-effect consistency, and uncertainty estimation.

Optional online world-model update. Let $m \le M$ be the number of actions actually executed from the selected chunk. We query the model with the executed prefix $A_{t,m}^{\text{sel}}$ and $h = m$ to obtain $(\hat{b}_{t+m}, \hat{z}_{t+m}, u_m)$. The feature-prediction update compares the predicted and observed features at that same endpoint:

$$\mathcal{L}_{\text{wm}}^{\text{on}} = \mathcal{D}\Big((\hat{b}_{t+m}, \hat{z}_{t+m}), \text{sg}((b_{t+m}, z_{t+m}))\Big). \tag{7}$$

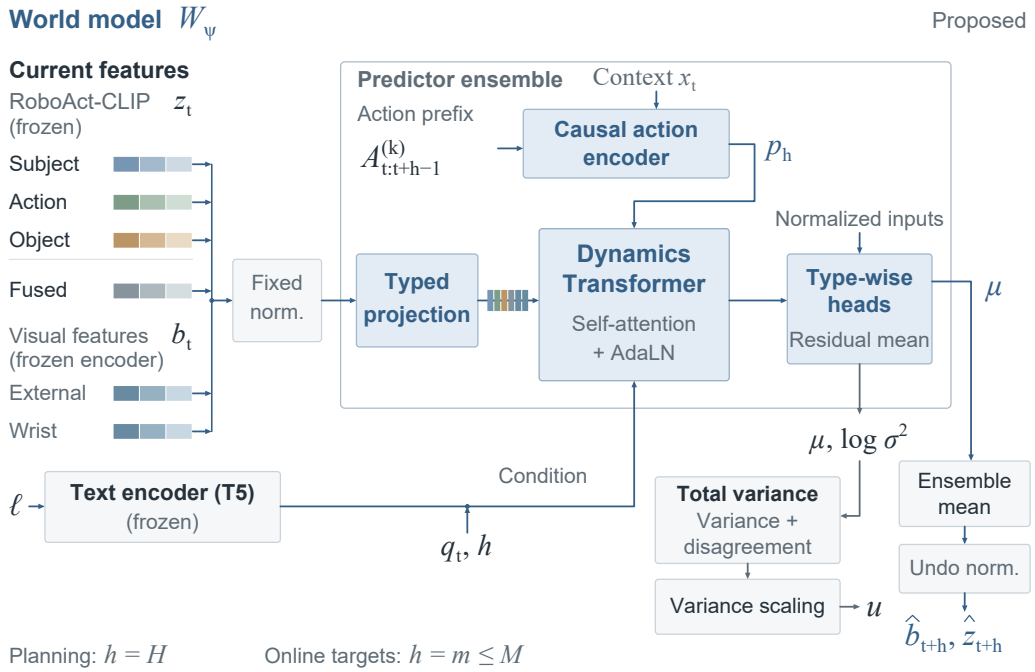


Figure 4: **Architecture of the latent world model.** Subject, action, object, fused, and two-view visual features are projected into typed tokens. An ordered action prefix, robot state, and frozen T5 text features condition their interaction through AdaLN. Type-specific residual Gaussian heads predict the same feature spaces as the frozen encoders. Independently trained predictors provide ensemble means and total variance; fixed normalization is inverted for the feature outputs, while held-out variance scaling yields scalar u. The Normalized inputs port reuses current standardized features for residual prediction. Planning queries $h = H$. Online targets use the actually executed prefix of length $h = m \le M$. Each query predicts one endpoint, not an autoregressive latent trajectory.

A.2 Evaluator Architecture and Training

Transition tokens and candidate query. Figure 5 shows the evaluator. Shared within-type projections combine each type's current feature, predicted feature, and feature difference into a transition token, giving six tokens in total. A query encodes the action prefix, robot state, instruction, and prefix length h. Cross-attention over the transition tokens retains a query residual. A scalar head also receives the prediction uncertainty and outputs the base score

$$s^{(k)}_{\omega,h} = G_\omega(z_t, b_t, \hat{z}^{(k)}_{t+h}, \hat{b}^{(k)}_{t+h}, A^{(k)}_{t,h}, q_t, \ell, h, u^{(k)}_h). \tag{8}$$

All candidates share the same evaluator. During planning, $s^{(k)}_\omega = s^{(k)}_{\omega,H}$ and $u^{(k)} = u^{(k)}_H$. The base score is combined with the semantic, feasibility, uncertainty, and action-regularization terms in Eq. (2) to obtain the final value $J^{(k)}$ used for selection and reinforcement learning.

Semantic progress signal. RoboAct-CLIP's pretrained action–text alignment provides a semantic compatibility signal. Using its frozen visual/action and language projections, Φ_v and Φ_l, we compute

$$\begin{aligned} c_t &= \cos(\Phi_v(b_t, z_t), \Phi_l(\ell)), \\ \hat{c}^{(k)}_{t+H} &= \cos\big(\Phi_v(\hat{b}^{(k)}_{t+H}, \hat{z}^{(k)}_{t+H}), \Phi_l(\ell)\big), \\ \Delta^{(k)}_{\text{sem}} &= \hat{c}^{(k)}_{t+H} - c_t. \end{aligned} \tag{9}$$

A positive value indicates increased similarity to the task-language embedding. This provides a proxy for semantic progress, rather than a guarantee of physical task completion.

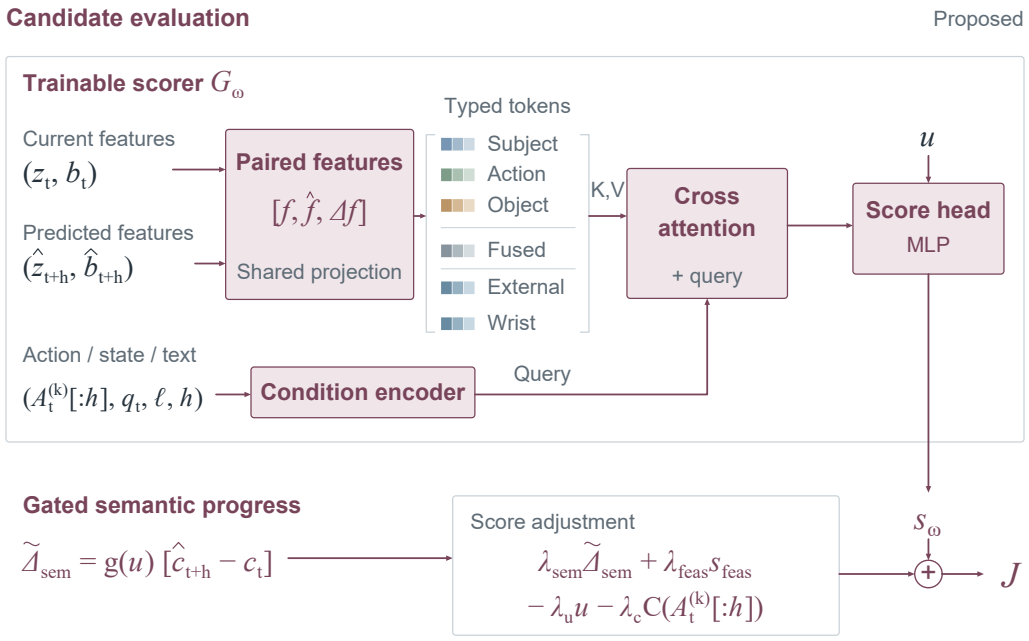


Figure 5: **Architecture of the language-aligned evaluator.** Shared within-type projections pair current and predicted features and their differences into six transition tokens. A query encodes the action prefix, robot state, instruction, and prefix length h. Cross-attention retains a query residual; a scalar head also receives uncertainty and produces s_ω. The external adjustment preserves $J = s_\omega + \lambda_{\text{sem}}\widetilde{\Delta}_{\text{sem}} + \lambda_{\text{feas}}s_{\text{feas}} - \lambda_u u - \lambda_c C(A)$, with uncertainty-gated future-minus-current semantic alignment. All candidates share one G_ω.

Uncertain predictions are downweighted by

$$\widetilde{\Delta}^{(k)}_{\text{sem}} = \sigma\left(\frac{\tau_u - u^{(k)}}{T_u}\right)\Delta^{(k)}_{\text{sem}}, \tag{10}$$

where σ is the sigmoid function, τ_u is the uncertainty threshold, and T_u controls the gate's temperature. In Eq. (2), $s^{(k)}_{\text{feas}}$ estimates action feasibility, while $C(A^{(k)}_t)$ penalizes large or unsmooth actions.

Offline evaluator objective. The evaluator is trained with successful, failed, and counterfactual candidates:

$$\begin{aligned}\mathcal{L}^{\text{off}}_{\text{eval}} = &\underbrace{\text{BCE}(J, y_{\text{good}})}_{\mathcal{L}_{\text{bin}}} + \lambda_r \underbrace{\max(0, \delta_{\text{rank}} - J^+ + J^-)}_{\mathcal{L}_{\text{rank}}} \\ &+ \lambda_p \underbrace{\|\sigma(s_{\text{prog}}) - p_t\|_1}_{\mathcal{L}_{\text{prog}}} + \lambda_m \underbrace{\|\widetilde{\Delta}_{\text{sem}} - \text{sg}(\Delta^{\text{true}}_{\text{sem}})\|_1}_{\mathcal{L}_{\text{sem}}}.\end{aligned} \tag{11}$$

The four terms correspond to good/bad action classification, preference ranking with margin δ_{rank}, trajectory progress regression, and semantic calibration using true future features when available. Here s_{prog} is the progress-prediction score supervised by the trajectory-progress target p_t. During online training, environment feedback refines the final value using the temporal-difference objective in Eq. (3).

A.3 Training Procedure and Policy Optimization

Two-stage training procedure. RoboAct-WorldVLA is trained in two stages. During offline training, the generator learns from demonstrations, the world model learns candidate-conditioned future features from successful and failed transitions, and the evaluator learns from successful, failed,

and counterfactual candidates. During online fine-tuning, the system executes selected action prefixes and collects transitions and rewards to update the evaluator and improve the policy. The world model can optionally be refined using the observed transitions. The RoboAct-CLIP encoder remains frozen throughout both stages. Algorithm 1 summarizes the procedure.

Algorithm 1 Training Procedure of RoboAct-WorldVLA

Require: Frozen encoders; generator π_θ, world model W_ψ, and evaluator G_ω; demonstrations, successful and failed transitions, and counterfactual candidates; K, H, and $M \le H$.
Ensure: Trained generator, world model, and evaluator.
Keep the encoders frozen; for a VLA generator, update only its LoRA modules, adapters, and action head.
Stage 1: Offline training
Train the generator on demonstrations using Eq. (12) or Eq. (13).
Train the world model on successful and failed transitions using Eq. (6), with future-feature targets from frozen encoders.
Train the evaluator on successful, failed, and counterfactual candidates using Eq. (11).
Stage 2: Online fine-tuning
while the online training budget remains **do**
  Obtain current features z_t, b_t, robot state q_t, and instruction ℓ.
  Sample K candidate action chunks $\{A_t^{(k)}\}_{k=1}^{K}$ of length H from π_θ.
  Predict each candidate's future features and uncertainty using Eq. (1) with $h = H$.
  Compute $J^{(k)}$ using Eq. (2) and select $k^\star = \arg\max_k J^{(k)}$.
  Execute the selected prefix, recording its actual length $m \le M$, observed transition, and environment reward.
  Compute r_t using Eq. (14); form the evaluator and policy-improvement terms from the collected experience and value estimates.
  if online world-model updates are enabled **then**
    Query W_ψ with the executed prefix $A_{t,m}^{\mathrm{sel}}$ and $h = m$.
    Form $\mathcal{L}_{\mathrm{wm}}^{\mathrm{on}}$ using the observed endpoint features in Eq. (7).
  end if
  Update trainable parameters using Eq. (16), including the corresponding generator objective, optional world-model term, and offline stabilization.
  Proceed to the next interaction.
end while

Flow-based generator. We use flow matching (Lipman et al., 2022). Let A_t^{demo} be the demonstrated action chunk, $x_1 = A_t^{\mathrm{demo}}$, $x_0 \sim \mathcal{N}(0, I)$, and $\tau \sim \mathcal{U}(0, 1)$. We define $x_\tau = (1-\tau)x_0 + \tau x_1$, $d^\star = x_1 - x_0$, and predict $d_\theta = f_\theta(x_\tau, \tau, z_t, b_t, q_t, \ell)$. The offline generator loss is

$$\mathcal{L}_{\mathrm{gen}}^{\mathrm{off}} = \underbrace{\|d_\theta - d^\star\|_2^2}_{\mathcal{L}_{\mathrm{FM}}} + \lambda_a \underbrace{\|\hat{A}_t - A_t^{\mathrm{demo}}\|_1}_{\mathcal{L}_{\mathrm{act}}} + \lambda_s \underbrace{\sum_{i=1}^{H-1} \|\hat{a}_{t+i} - \hat{a}_{t+i-1}\|_2^2}_{\mathcal{L}_{\mathrm{smooth}}}. \tag{12}$$

The three terms correspond to flow matching, action reconstruction, and temporal smoothness.

Open-source VLA adaptation. Each VLA retains its native observation and language interface. We fine-tune only LoRA modules, lightweight adapters, and the action head, with $\theta_{\mathrm{train}} = \{\theta_{\mathrm{LoRA}}, \theta_{\mathrm{adapter}}, \theta_{\mathrm{head}}\}$. Offline adaptation uses

$$\mathcal{L}_{\mathrm{gen}}^{\mathrm{VLA}} = \|\pi_\theta^{\mathrm{VLA}}(o_t, q_t, \ell) - A_t^{\mathrm{demo}}\|_1 + \lambda_s \mathcal{L}_{\mathrm{smooth}}. \tag{13}$$

During online fine-tuning, the same trainable modules are updated by reward-guided objectives while the main VLA backbone remains fixed. The world model and evaluator provide consequence-aware filtering and reinforcement for the proposed actions.

Interaction reward. At each interaction step, the generator proposes K candidates, the world model predicts their consequences, and the evaluator selects A_t^{sel}. After executing $m \le M$ actions,

the reward is

$$r_t = r_t^{\text{env}} + \eta_{\text{sem}}\big[\cos(\Phi_v(b_{t+m}, z_{t+m}), \Phi_l(\ell)) - \cos(\Phi_v(b_t, z_t), \Phi_l(\ell))\big] - \eta_c C(A_t^{\text{sel}}). \tag{14}$$

When only benchmark rewards are used, $\eta_{\text{sem}} = 0$. The candidate budget K is fixed during training and inference; the experiments analyze its success–latency trade-off.

Reward-guided generator update. The evaluator is updated using Eq. (3), and the generator is reinforced toward high-return actions. For the flow generator,

$$\mathcal{L}_{\text{gen}}^{\text{RL}} = w_t \mathcal{L}_{\text{FM}}(A_t^{\text{sel}}), \qquad w_t = \text{sg}\Big[\exp\Big(\widehat{\text{Adv}}_t/\alpha\Big)\Big], \tag{15}$$

where $\widehat{\text{Adv}}_t$ is an advantage estimate and α controls the weighting temperature. For VLA generators, we use a policy-gradient objective when action log-probabilities are available, or advantage-weighted action regression otherwise.

Complete online objective. Combining policy improvement, evaluator refinement, the optional world-model update, and offline stabilization gives

$$\begin{aligned} \mathcal{L}_{\text{online}} &= \lambda_g \mathcal{L}_{\text{gen}}^{\text{RL}} + \lambda_e \mathcal{L}_{\text{eval}}^{\text{RL}} + \lambda_w \mathcal{L}_{\text{wm}}^{\text{on}} + \lambda_s \mathcal{L}_{\text{stab}}, \\ \mathcal{L}_{\text{stab}} &= \mathcal{L}_{\text{gen}}^{\text{off}} + \mathcal{L}_{\text{wm}} + \mathcal{L}_{\text{eval}}^{\text{off}} + \lambda_c C(A_t^{\text{sel}}). \end{aligned} \tag{16}$$

For a VLA generator, the generator stabilization term uses the corresponding offline adaptation objective in Eq. (13). The world model and evaluator remain part of candidate selection at deployment.

Table 5: Validation cosine similarity of latent world-model future-feature prediction on Close Jar. Copy-current denotes a baseline that directly uses the current latent feature as the future feature.

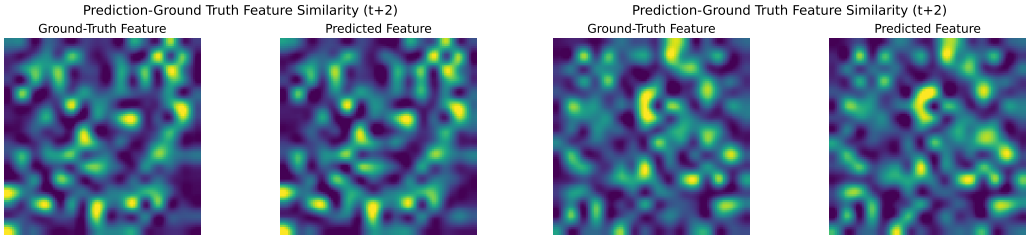
Task	WM $t+2$	WM $t+10$	Copy-current $t+2$	Copy-current $t+10$
Close Jar	**0.9537**	**0.9455**	0.7278	0.6921

B Latent World-Model Feature Similarity on Close Jar

We provide additional diagnostics for the latent world model used by RoboAct-WorldVLA on the RLBench Close Jar task. The diagnostic evaluates whether the world model predicts future representations that remain close to the representations extracted from true future observations. Since RoboAct-WorldVLA performs consequence prediction in a frozen latent space rather than in pixel space, the visualizations below should be interpreted as feature-space agreement rather than image reconstruction quality.

Table 5 reports the validation cosine similarity of the predicted future features. We also compare against a copy-current baseline, which reuses the current feature as the future prediction. The learned world model is substantially closer to the true future feature than this baseline at both short and longer prediction horizons.

For each held-out transition, we compare the ground-truth future feature with the world-model prediction. We visualize the frozen 256-dimensional visual feature by reshaping it into a 16×16 latent feature map. Each ground-truth/prediction pair is plotted with the same robust percentile normalization so that the color scale is comparable within the pair. The numerical cosine similarities in Table 5 are computed on the original unnormalized feature vectors.



(a) External view (b) Wrist view

Figure 6: **Close Jar, dual-view future-feature prediction at $t+2$.** Each view compares the ground-truth future feature (left) with the world-model prediction (right) for a held-out transition. Each pair uses shared robust percentile normalization to qualitatively visualize feature-space agreement.

C Qualitative Example of World-Model-Guided Action Selection

This appendix provides a concrete example of how RoboAct-WorldVLA uses predicted action consequences to guide action generation. We follow the notation in Sec. 3. At time t, the frozen RoboAct-CLIP encoder provides the latent action-understanding representation z_t, the frozen visual encoder provides raw visual features b_t, and the policy observes the robot state q_t and language instruction ℓ. The action generator proposes candidate action segments $A_t^{(k)}$, and the latent world model predicts their future representations:

$$(\hat{b}_{t+H}^{(k)}, \hat{z}_{t+H}^{(k)}, u^{(k)}) = W_\psi(z_t, b_t, q_t, A_{t,H}^{(k)}, \ell, H). \tag{17}$$

The language-aligned action-value evaluator then scores each predicted consequence:

$$s_\omega^{(k)} = G_\omega(z_t, b_t, \hat{z}_{t+H}^{(k)}, \hat{b}_{t+H}^{(k)}, A_{t,H}^{(k)}, q_t, \ell, H, u^{(k)}), \tag{18}$$

Table 6: Action-value scores and world-model diagnostics at the branch state ($t = 179$). Higher evaluator scores are better; lower cup-placement error is better. The selected action has a higher logged action value $J_{\log}$, and the world-model rollout predicts a lower 20-step cup error for this candidate. The realized selected branch succeeds, whereas the rejected branch places the cup at a shifted wrong location and keeps a large residual placement error.

Candidate	s_{succ}	s_{rew}	s_{prog}	$J_{\log}$	WM h=1	WM h=20	Actual +20	Final
Selected action	0.9433	0.6152	0.4126	0.8001	0.0479	0.0849	0.0064	0.0129
Rejected action	0.9423	0.5897	0.4088	0.7946	0.0478	0.0872	0.1606	0.1876

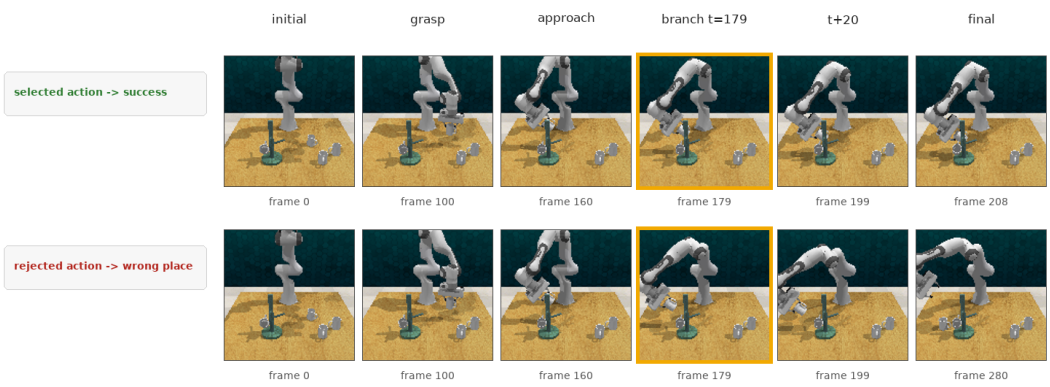


Figure 7: External-view keyframes for the selected and rejected branches. The yellow outline marks the decision state at $t = 179$. Selecting the candidate with higher $J_{\log}$ and lower predicted rollout error moves the trajectory toward the cup stand and completes the placement. The rejected candidate actively moves toward a shifted target and leaves the cup at an incorrect location.

which is combined with the other score components in Eq. (2) to obtain $J^{(k)}$. The executed candidate is selected by

$$k^\star = \arg\max_{k=1,\dots,K} J^{(k)}. \tag{19}$$

Thus, action generation is reinforced by predicted action effects rather than by imitation likelihood alone.

We visualize this mechanism on the RLBench PLACE CUPS task. The two rollouts are generated from the same seed, variation, and episode index. The selected branch uses the unshifted dense waypoint action and succeeds after 208 steps. The rejected branch is a controlled wrong-place counterfactual: it uses the same dense waypoint controller, but shifts the placement target by $0.16\,\text{m}$ along the negative y direction. This branch actively moves the cup to an incorrect location and times out after 280 steps. At the branch state ($t = 179$), we score the selected action and the same-phase wrong-place action at the successful branch state; the downstream outcome of the rejected candidate is shown using its corresponding real wrong-place rollout.

For this RLBench diagnostic, the reported score combines the logged success, dense-reward, and progress components:

$$J_{\log}^{(k)} = \frac{s_{\text{succ}}^{(k)} + 0.25 s_{\text{rew}}^{(k)} + 0.25 s_{\text{prog}}^{(k)}}{1.5}. \tag{20}$$

This is the reported diagnostic score used to compare the two branch candidates in Table 6. The final RoboAct-WorldVLA action value is defined by Eq. (2):

$$J^{(k)} = s_\omega^{(k)} + \lambda_{\text{sem}} \tilde{\Delta}_{\text{sem}}^{(k)} + \lambda_{\text{feas}} s_{\text{feas}}^{(k)} - \lambda_u u^{(k)} - \lambda_c C(A_t^{(k)}). \tag{21}$$

We also report a geometric cup-placement error predicted by the rollout world model as an interpretable consequence diagnostic. Lower predicted and realized cup error indicates that the candidate is moving the system toward the language-specified goal.

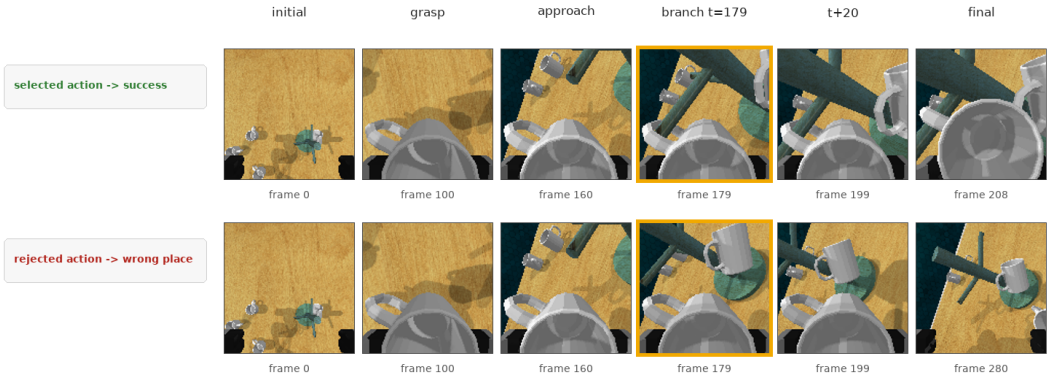
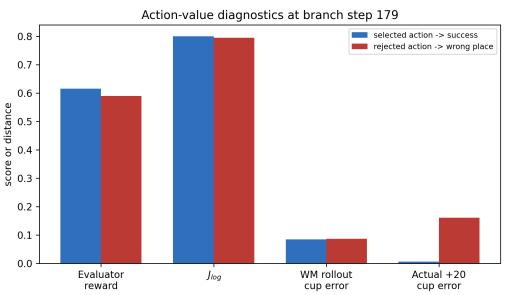
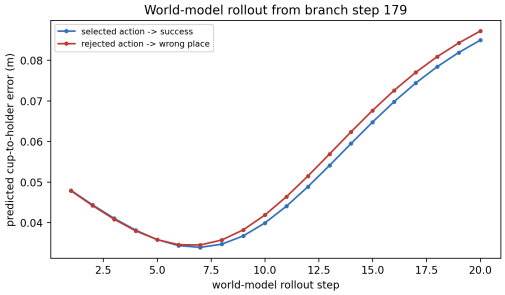


Figure 8: Wrist-view keyframes for the same selected and rejected branches. The wrist view makes the wrong-place outcome visually explicit: the rejected branch carries the cup away from the holder instead of seating it on the stand.



(a) Logged score components and $J_{\log}$.



(b) World-model predicted cup error over a 20-step rollout.

Figure 9: Branch diagnostics used to explain the action choice. The evaluator assigns a higher logged action value to the selected candidate, and the world model predicts a lower short-horizon placement error. The realized wrong-place rollout then confirms that the rejected branch quickly diverges from the target placement.

This example illustrates the intended inference behavior of RoboAct-WorldVLA. The generator produces locally plausible candidates, the world model predicts their latent and geometric consequences, and the evaluator assigns a higher action value to the candidate whose predicted future better matches the task direction. The resulting action choice moves the robot onto a successful trajectory, while the rejected candidate produces an active but incorrect placement.

D Real-World Experimental Setup

JAKA platform. We conduct real-world experiments on a dual-arm JAKA K-1 humanoid manipulation platform. The system consists of two 7-DoF robotic arms, each with a 3 kg payload and a 760 mm reach. Each arm is equipped with a VI-H six-axis force–torque sensor and an AG-105-145 electric parallel gripper. For visual perception, we use an Orbbec Gemini336L RGB-D camera as the external camera to observe the drawer and tabletop workspace. In addition, two wrist-mounted Logitech C922 Pro Stream Webcams are installed on the left and right arms to provide close-range views for manipulation.

NexArm platform and data collection. We use the ESP32 version of NexArm, with five actuated motion joints and one gripper opening/closing channel. The arm uses bus servos and an ESP32–AT32F421 dual-controller system. A wrist-mounted USB RGB camera and an external USB RGB camera each capture images at 640×480 resolution. The arm communicates with the host computer

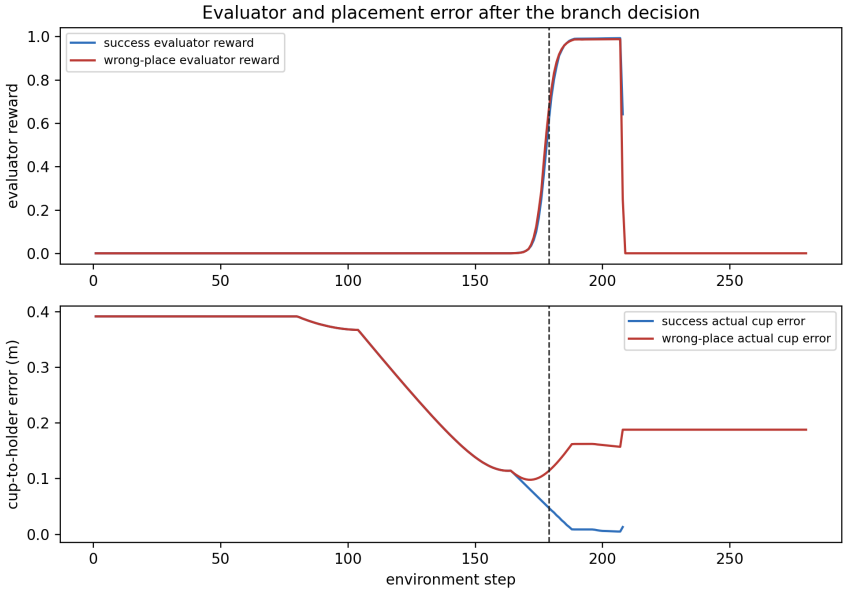


Figure 10: Evaluator reward and geometric cup-placement error along the realized trajectories. The selected branch reaches low final cup error and high late evaluator reward. The wrong-place branch keeps a large residual placement error and receives a near-zero late evaluator reward.

Table 7: Hardware configuration of the real-world dual-arm experimental platform.

Component	Specification
Robot platform	JAKA K-1 humanoid dual-arm platform
Manipulators	2× 7-DoF arms, 3 kg payload, 760 mm reach
Force–torque sensing	VI-H six-axis force–torque sensor
End effector	AG-105-145 electric parallel gripper
External camera	Orbbec Gemini336L RGB-D camera
Wrist cameras	2× Logitech C922 Pro Stream Webcam
Workspace	Drawer and tabletop manipulation setup

through a 1 Mbps USB serial connection. Joint positions and both image streams are recorded at a software sampling rate of 20 Hz. The recorded state and action each have six channels: five motion joints and the gripper. The 20 Hz rate refers to software data acquisition.

The task set includes placing specified fruit or vegetables into a basket, stacking blocks, pushing a block into a specified region, and hanging a cup. The selected recordings illustrate peanut placement into a target container, green-block stacking, pushing a yellow block into a yellow region, and cup hanging.

E Qualitative Visualization Results

This section presents qualitative manipulation sequences on RLBench, ManiSkill3, JAKA, and NexArm. In each composite figure, task sequences are arranged in rows and key frames progress from left to right.

E.1 RLBench Qualitative Results

Figure 11 shows representative rollout visualizations on RLBench. Each row corresponds to a different task, and the frames from left to right illustrate the execution process from the initial state to task completion. The examples cover precision insertion, object placement, stacking, sorting, articulated-object manipulation, and long-horizon manipulation. These qualitative results show that

RoboAct-WorldVLA can generate actions that progressively move the scene toward the language-specified goal.

E.2 ManiSkill3 Qualitative Results

Figure 12 presents qualitative rollout visualizations on ManiSkill3. Following the same format, each row shows one task trajectory and each column corresponds to a key execution frame. These examples demonstrate that RoboAct-WorldVLA can serve as a consequence-aware decision layer for different VLA action generators, improving action selection in object picking, stacking, and cluttered manipulation scenarios.

E.3 JAKA Real-World Qualitative Results

Figure 13 shows representative JAKA rollouts for Toy-Match, Cup-Hang, and Box-Push. The tasks involve semantic matching, bimanual manipulation, and contact-rich pushing under multi-view observation and continuous end-effector control.



Figure 11: Qualitative visualization results on RLBench. RoboAct-WorldVLA maintains stable closed-loop behavior across diverse tasks involving insertion, placement, stacking, sorting, sweeping, and articulated-object manipulation.

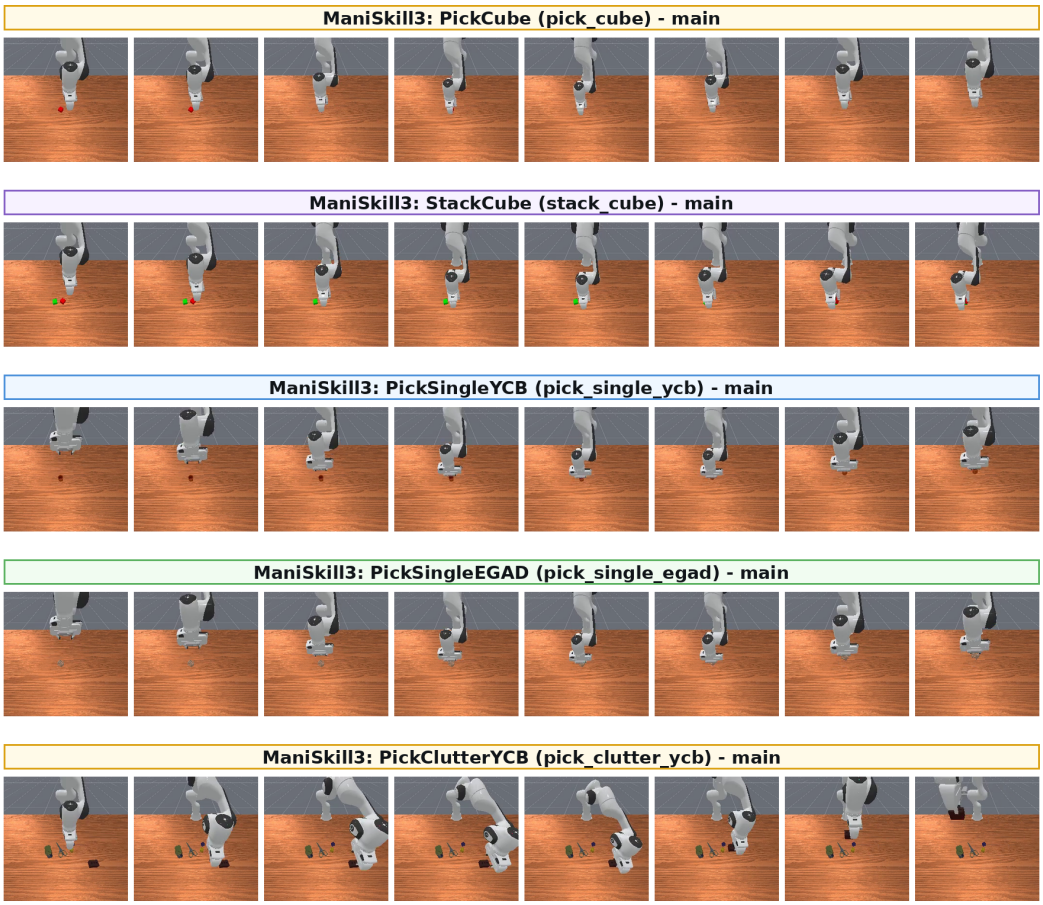


Figure 12: Qualitative visualization results on ManiSkill3. The results illustrate that the proposed latent world-model-guided evaluation helps select actions with better predicted task progress across object-level and scene-level variations.

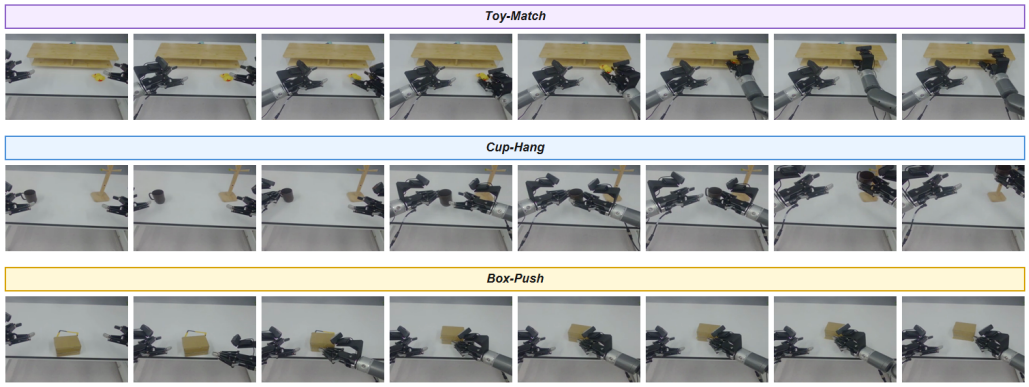


Figure 13: JAKA real-world rollouts for Toy-Match, Cup-Hang, and Box-Push. Frames progress from left to right within each task.

E.4 NexArm Qualitative Results

Figures 14 and 15 show four NexArm task execution sequences. The illustrated tasks are peanut placement into a box, green-block stacking, pushing a yellow block into a yellow region, and cup hanging. Each task pairs external and wrist views at the same recorded frame indices, with time progressing from left to right.

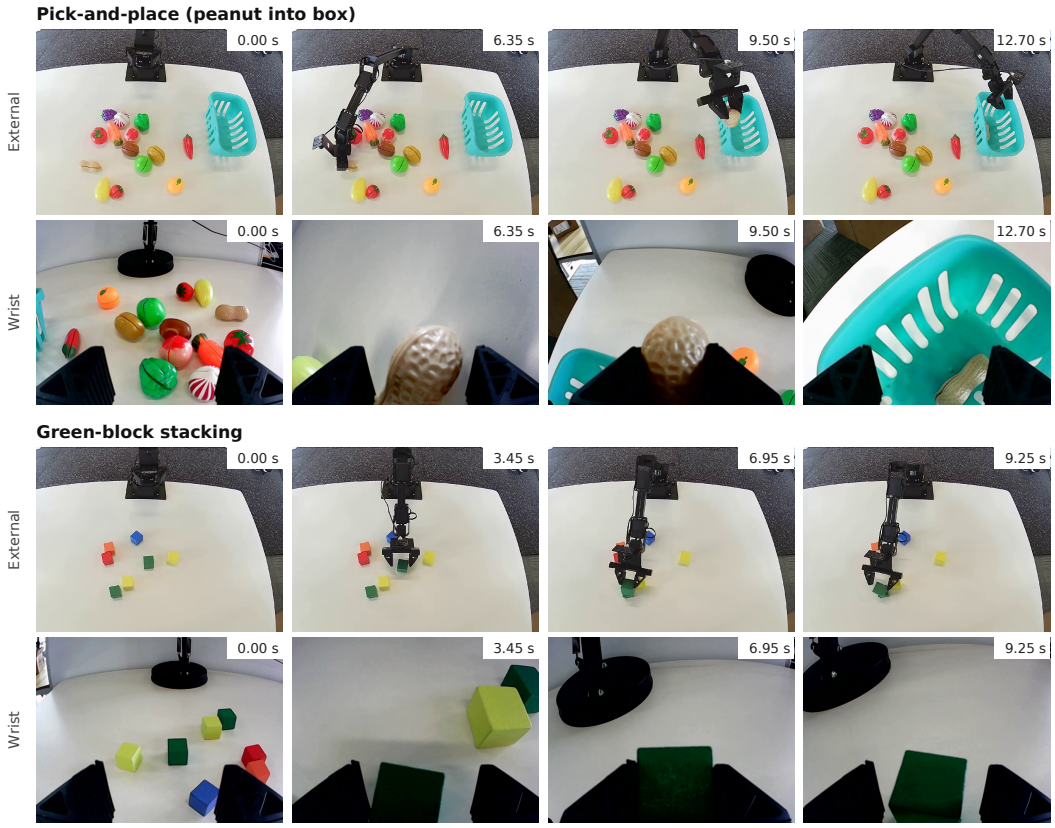


Figure 14: **NexArm placement and stacking.** Four recorded times per task, with external views above and wrist views below.

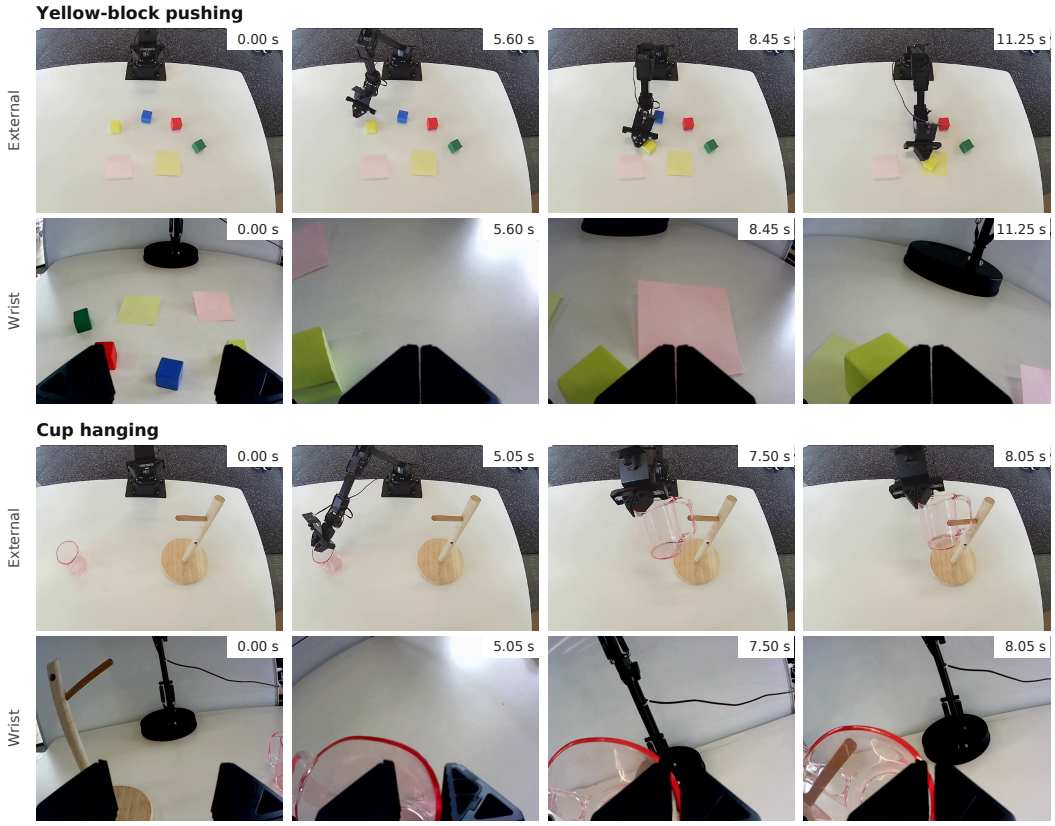


Figure 15: **NexArm pushing and cup hanging.** Four recorded times per task pair external and wrist views. The sequences show the yellow block being pushed into the yellow target region and the cup being grasped, lifted, and hung on the peg.

F LIMITATIONS

RoboAct-WorldVLA still has several limitations. First, it relies on the frozen RoboAct-CLIP latent space; failure modes that are poorly represented in this space, including subtle contact states, force-dependent constraints, and transparent or deformable objects, may be difficult to evaluate. Second, the world model predicts fixed-horizon latent consequences and may miss very long-horizon dependencies or delayed constraints. Third, fixed-budget candidate evaluation improves robustness but adds planning latency, so faster control loops may require early rejection, evaluator distillation, or adaptive compute allocation. Fourth, the evaluator benefits from failed trajectories and counterfactual candidates. Insufficient training data of these types may lead to inaccurate candidate scores, reducing action-selection quality and task success rates.