PRIORITIZED ROLLOUTS FOR EFFICIENT WORLD MODEL-BASED VISION-LANGUAGE-ACTION POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models have emerged as a powerful paradigm for embodied intelligence, but fine-tuning them with reinforcement learning (RL) remains constrained by the cost of real-world robot interaction. Model-based reinforcement learning (MBRL) reduces this cost by using a learned world model to generate rollouts for policy optimization. However, it becomes computationally expensive as VLA policies and world models scale. Existing methods typically treat states equally, overlooking substantial differences in their utility for policy improvement. In this paper, we show that policy uncertainty helps identify states with greater potential for policy improvement. The policy exhibits high uncertainty at only a small subset of states, often during decision-sensitive stages where small action differences can alter task outcomes, suggesting that policy improvements at these states could be particularly valuable. Building on these findings, we introduce **U-GROW**, a lightweight, plug-and-play sampling layer that directs more model rollouts to these informative states. By modifying only the branched-start distribution, U-GROW can be integrated into existing MBRL pipelines without changing the policy optimization objective. Experiments in both simulated and real-world manipulation tasks demonstrate the efficiency and effectiveness of U-GROW, supporting the use of policy uncertainty to guide experience generation.

1 INTRODUCTION

Recent advances in Vision-Language-Action (VLA) models have marked substantial progress toward general-purpose robotic control (Kim et al., 2025; Amin et al., 2026). Building on the broad capabilities learned through offline imitation, reinforcement learning (RL) can further improve the task performance and generalization of VLA policies (Chen et al., 2025; Li et al., 2026). However, online RL on physical robots requires repeated real-world robot interaction, which is slow, costly, and difficult to parallelize. World models can alleviate this bottleneck by generating rollouts for policy optimization, thereby reducing the need for additional real-robot interaction (Yu et al., 2026; Hou et al., 2026). Recent methods following this paradigm have demonstrated effective RL fine-tuning of VLA policies using imagined rollouts (Zhu et al., 2026; Zhang et al., 2026b).

As VLA policies and world models scale (Zhang et al., 2024; NVIDIA, 2026), the computational costs of rollout generation and policy optimization increase (Zhu et al., 2026; Zhang et al., 2026b; Guo et al., 2026), making efficient use of model-generated experience essential. Intuitively, states may differ in their value for policy improvement. For example, correcting a small alignment error may turn failure into success, whereas further optimizing already reliable free-space motion may yield little additional benefit. However, existing methods typically do not explicitly account for these state-level differences. This raises a central question: *Which states offer the greatest potential for policy improvement, and how can they be identified?*

Policy uncertainty may help identify such states. High uncertainty can indicate less reliable action predictions (Ren et al., 2026), suggesting potential room for policy improvement. Our empirical analysis shows that the policy exhibits high uncertainty at only a small subset of visited states, often during decision-sensitive manipulation stages such as grasping, placement, and release. At these stages, small differences in spatial alignment or timing can alter subsequent interactions and task

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

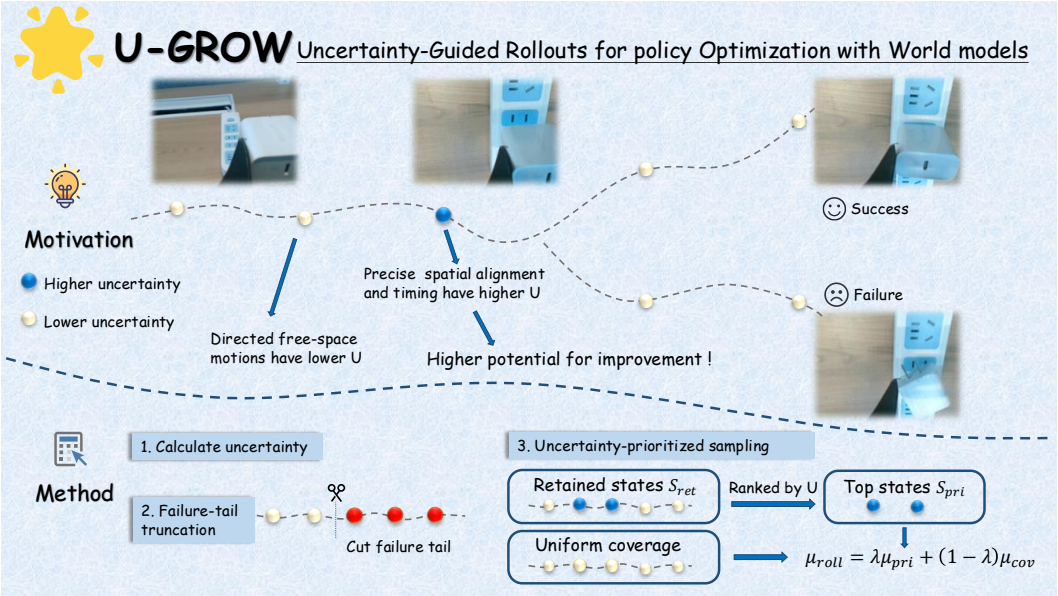

Figure 1: Overview of U-GROW

outcomes. More directly, a policy-switching intervention shows that, at comparable replacement rates, replacing the base policy with a stronger policy at high-uncertainty states yields substantially higher success rates than doing so at low-uncertainty states. Together, these findings support using policy uncertainty to identify states with greater potential for policy improvement.

Motivated by these observations, we propose U-GROW (**U**ncertainty-**G**uided **R**ollouts for policy **O**ptimization with **W**orld models), a lightweight method that directs model rollouts toward states with greater potential for policy improvement. Specifically, we adopt short-horizon branched rollouts (Janner et al., 2019) and use an uncertainty-guided sampling layer to select their starting states. After filtering out difficult-to-recover failure tails, this layer prioritizes high-uncertainty states as rollout starts. By modifying only the rollout-start distribution, U-GROW can be plugged into existing MBRL pipelines without changing their policy optimization objectives. We evaluate U-GROW on the RoboTwin 2.0 and LIBERO benchmarks, as well as four real-world manipulation tasks. Across these settings, U-GROW improves both sample efficiency and final success rates over uniform rollout-start sampling, with the same world model and rollout budget. These results highlight the value of uncertainty-guided rollout allocation for policy optimization with world models.

2 Related Works

2.1 Reinforcement Learning for VLA

Reinforcement learning (RL) can improve the performance and generalization of Vision-Language-Action (VLA) policies beyond offline imitation (Guo et al., 2025). For autoregressive VLA policies such as OpenVLA (Kim et al., 2025), recent work develops on-policy RL frameworks for post-training (Lu et al., 2025; Li et al., 2026) and examines their effects on semantic generalization and execution robustness (Liu et al., 2025). For flow-matching policies (Black et al., 2025a;b), π_{RL} and FPO enable RL fine-tuning through stochastic denoising formulations and flow-matching-based surrogate objectives, respectively (Chen et al., 2025; McAllister et al., 2026). These online approaches rely on repeated environment interaction, which is costly and difficult to parallelize when training on physical robots (Guo et al., 2026). Offline model-free RL reduces this interaction burden by learning directly from existing datasets (Levine et al., 2020). Recent methods leverage value estimates to learn from data of varying quality; for example, HVD uses hierarchical value decomposition for whole-body control (Zhang et al., 2026a). Beyond purely offline training, Amin et al. (2026) use advantage conditioning to combine offline pretraining with iterative learning from demonstrations, autonomous experience, and expert corrections. Without an explicit dynamics model to generate ad-

ditional experience, however, policy improvement in the offline setting remains strongly dependent on the coverage of the available data (Levine et al., 2020).

2.2 World Models for Reinforcement Learning

Model-based reinforcement learning (MBRL) improves data efficiency by using learned dynamics to generate experience for policy optimization (Janner et al., 2019; Yu et al., 2020; Sun et al., 2023; Lin et al., 2025; 2026). Recent work extends this paradigm to VLA post-training through action-conditioned visual rollouts and task feedback (Hou et al., 2026; Xiao et al., 2025; Li et al., 2025; Zhu et al., 2026). WMPO optimizes VLA policies through closed-loop interaction with a learned world model, using the resulting full-horizon trajectories for policy updates (Zhu et al., 2026). A central challenge in this setting is that model prediction errors can compound over long rollouts, compromising the reliability of generated experience (Janner et al., 2019). To mitigate this issue, WoVR initializes rollouts from keyframes (Jiang et al., 2026), while VLA-MBPO uses short-horizon, chunk-level rollouts branched from offline states (Zhang et al., 2026b). However, model-based VLA post-training remains computationally demanding (Zhu et al., 2026; Zhang et al., 2026b), motivating more effective strategies for allocating computational resources.

3 Preliminaries

3.1 Markov Decision Process

We consider a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \rho_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively, $T(s_{t+1}, r_t \mid s_t, a_t)$ denotes the joint distribution of the next state and reward, ρ_0 is the initial-state distribution, and $\gamma \in [0, 1)$ is the discount factor. The objective is to learn a policy $\pi(a \mid s)$ that maximizes the expected discounted return:

$$J(\pi) = \mathbb{E}_{\rho_0, \pi, T}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right].$$

3.2 Offline Model-based Reinforcement Learning

In offline MBRL, the agent only has access to a fixed dataset $\mathcal{D}_{\text{off}} = \{(s_t, a_t, r_t, s_{t+1})\}$ collected by a behavior policy. A dynamics model $\hat{T}_\phi(s_{t+1} \mid s_t, a_t)$ and a reward model $\hat{r}_\phi(s_t, a_t)$ are learned from $\mathcal{D}_{\text{off}}$ via supervised learning. MBRL methods often perform k-step branched rollouts with $\hat{T}_\phi$ starting from states $s \in \mathcal{D}_{\text{off}}$, store the generated transitions in a buffer $\mathcal{D}_{\text{wm}}$, and update the policy $\pi(a \mid s)$ using samples from $\mathcal{D}_{\text{wm}}$ (Janner et al., 2019; Yu et al., 2020).

3.3 Self-Consistency Uncertainty Estimation

Following Ren et al. (2026), we estimate state-wise uncertainty from sensitivity to denoising discretization. For an ideal straight flow, the same state s and initial noise z should yield an invariant action chunk across denoising step counts; disagreement therefore indicates local policy instability.

Let $A_\theta^{(k)}(s, z) \in \mathbb{R}^{H \times D}$ denote the action chunk generated with k denoising steps, where H is the action horizon and D is the action dimension. For a set of $N \geq 2$ denoising step counts $\mathcal{K} = \{k_1, \ldots, k_N\}$, let $\bar{A}_{h,j}(s, z) = \frac{1}{N}\sum_{n=1}^{N} A_{\theta,h,j}^{(k_n)}(s, z)$ denote the element-wise mean. For $p \geq 1$, we define the normalized element-wise disagreement as

$$u_{h,j}^{(p)}(s, z) = \frac{\left[\frac{1}{N}\sum_{n=1}^{N}\left|A_{\theta,h,j}^{(k_n)}(s, z) - \bar{A}_{h,j}\right|^p\right]^{1/p}}{\left[\frac{1}{N}\sum_{n=1}^{N}\left|A_{\theta,h,j}^{(k_n)}(s, z)\right|^p\right]^{1/p}}. \tag{1}$$

Averaging over action timesteps and dimensions yields a scalar uncertainty score:

$$U^{(p)}(s, z) = \frac{1}{HD}\sum_{h=1}^{H}\sum_{j=1}^{D} u_{h,j}^{(p)}(s, z). \tag{2}$$

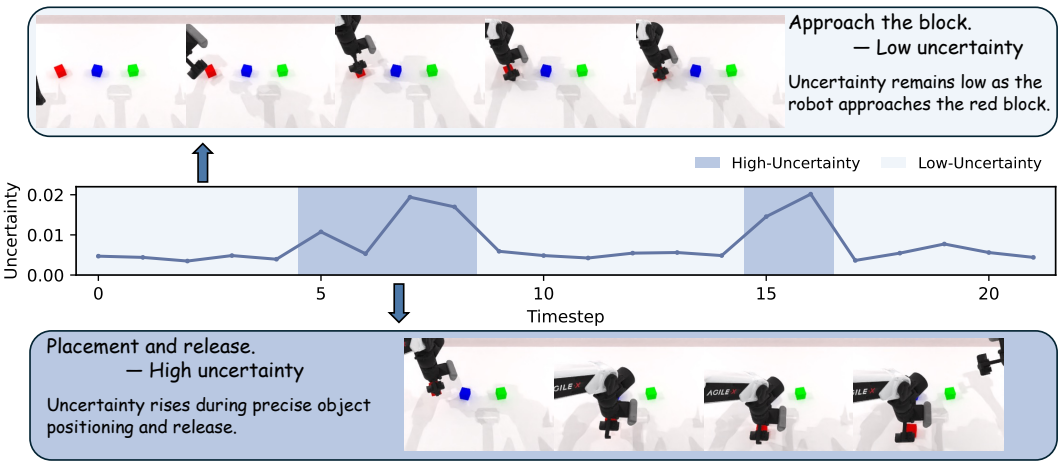

Figure 3: **Policy uncertainty across manipulation phases.** The curve shows policy uncertainty $U(s_t, z_t)$ along a representative trajectory. Shaded regions mark low- and high-uncertainty segments, with corresponding snapshots shown at the top and bottom, respectively. Uncertainty remains low as the robot approaches the red block but rises during placement and release, which require precise spatial alignment and timing.

4 Analyzing Policy Uncertainty in Embodied Manipulation

We first examine whether policy uncertainty can identify states with greater potential for policy improvement. To probe this, we conduct a policy-switching intervention on the Blocks Ranking RGB task in RoboTwin 2.0. We obtain the base policy π_{base} by supervised fine-tuning $\pi_{0.5}$ on expert demonstrations, and further fine-tune it with RL to obtain the stronger policy $\pi^\star$. During evaluation, π_{base} controls the robot by default. To decide whether to switch to $\pi^\star$ at each encountered state, we estimate policy uncertainty under π_{base} using the self-consistency estimator $U(s_t, z_t)$ defined in Equation 2. We then compare two intervention conditions. In the high-U condition, $\pi^\star$ replaces π_{base} at states satisfying $U(s_t, z_t) > \kappa_H$; in the low-U condition, it replaces π_{base} at states satisfying $U(s_t, z_t) < \kappa_L$. We refer to these conditions as high-U and low-U switching, respectively. We vary the thresholds to obtain a range of replacement rates, defined as the fraction of encountered states at which $\pi^\star$ is used, and evaluate each setting over 100 episodes.

Figure 2a shows that high-U switching achieves higher success rates than low-U switching across a broad range of replacement rates. At comparable replacement rates of roughly one-third, high-U switching achieves an 80% success rate, compared with 50% for low-U switching. The larger gains from stronger-policy interventions at high-uncertainty states suggest that policy uncertainty can help identify states with greater potential for policy improvement. This motivates using uncertainty to guide rollout-start selection.

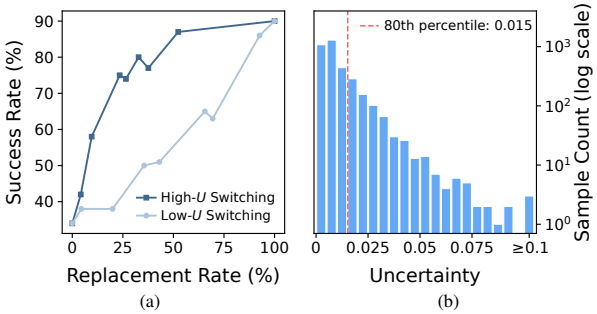

Figure 2: **(a)** Success rate when a stronger policy replaces the behavior policy at high- or low-uncertainty states. **(b)** Uncertainty across 3,532 visited states; the dashed line marks the 80th percentile.

To better understand the gains from high-U switching, we characterize the distribution of uncertainty across states visited by π_{base}. We roll out π_{base} for 100 episodes, obtaining 3,532 visited states, and compute $U(s_t, z_t)$ at each state. As shown in Figure 2b, π_{base} exhibits low uncertainty at most visited states and high uncertainty at only a small fraction of them. The 80th percentile is 0.0152, while the maximum reaches 0.1668. Together with the policy-switching results, this pattern suggests that opportunities for effective policy correction are also concentrated, helping explain how high-U switching achieves substantial gains at a modest replacement rate.

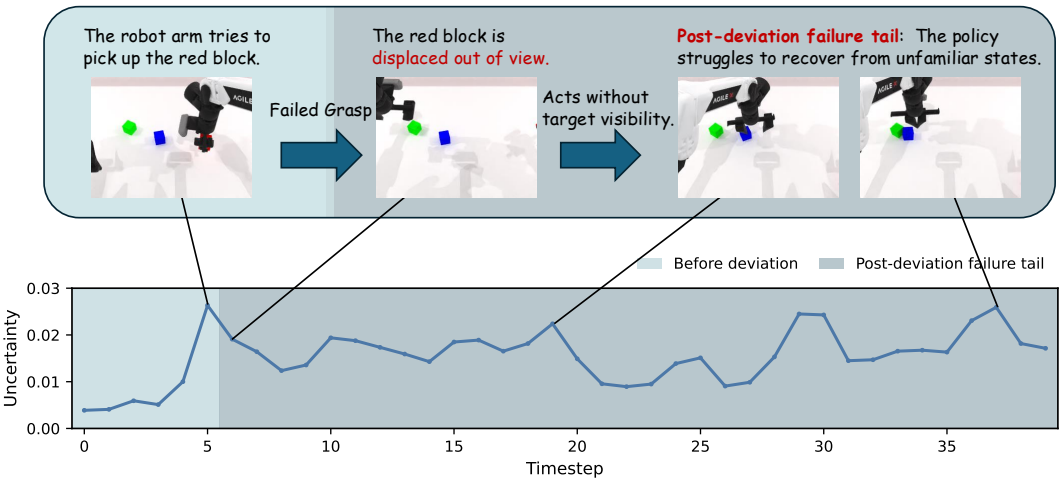

Figure 4: **Policy uncertainty in a post-deviation failure tail.** The curve shows policy uncertainty $U(s_t, z_t)$ along a representative failed trajectory. Snapshots illustrate a failed grasp that displaces the red block outside the camera's field of view, making recovery difficult. The shaded region marks the resulting post-deviation failure tail, in which the policy exhibits high uncertainty at several states.

We next examine how policy uncertainty varies across manipulation phases. We observe that uncertainty tends to be low during directed free-space motion but rises during grasping, placement, and release. Figure 3 illustrates this pattern in a representative trajectory: uncertainty remains low as the robot approaches the red block but increases during placement and release, which require mapping fine-grained geometric and temporal information to precise actions. Intuitively, higher uncertainty may indicate that the current policy has not yet learned this mapping reliably. Yet reliable action selection is particularly important here, as small action differences can alter subsequent object interactions and task outcomes. This offers a qualitative explanation for why high-U switching is effective: it directs stronger-policy interventions toward decision-sensitive states where the base policy may struggle and better actions can have a greater impact on task success.

However, high policy uncertainty does not always indicate greater potential for policy improvement. In failed trajectories, an early error can lead the policy into unfamiliar states from which recovery is difficult. We refer to the subsequent trajectory segment as a *post-deviation failure tail*. Figure 4 shows an example in which a grasping error displaces the red block outside the camera's field of view, making recovery difficult. Although the policy exhibits high uncertainty at several states in the resulting tail, these states may be less useful as rollout starts for policy improvement. These observations motivate excluding such failure tails from the prioritized rollout-start pool.

5 Method

Building on the observations above, we propose **U-GROW** (**U**ncertainty-**G**uided **R**ollouts for policy **O**ptimization with **W**orld models), a sampling layer that modifies only the rollout-start distribution without changing the policy optimization objective. Following VLA-MBPO (Zhang et al., 2026b), we generate short-horizon branched rollouts and use the resulting experience for policy updates with proximal policy optimization (PPO) (Schulman et al., 2017). To select rollout starts, U-GROW first filters out failure tails to construct a candidate pool for prioritized sampling, then prioritizes high-uncertainty states within this pool. We combine this prioritized sampling with uniform sampling over all offline states to broaden the state coverage for PPO critic training.

Let $\mathcal{D}_{\text{off}} = \{\tau_n\}_{n=1}^{N}$ denote a set of outcome-labeled offline trajectories. Each trajectory $\tau \in \mathcal{D}_{\text{off}}$ contains a state sequence $\{s_{\tau,t}\}_{t=0}^{T_\tau}$ and has a binary outcome label $y_\tau \in \{0, 1\}$, where $y_\tau = 1$ indicates success and $y_\tau = 0$ indicates failure. For each state, we draw $z_{\tau,t} \sim p_0$ and compute

$$u_{\tau,t} = U(s_{\tau,t}, z_{\tau,t}),$$

where U is the self-consistency uncertainty estimator defined in Equation 2. Using trajectory outcomes and uncertainty scores, U-GROW first truncates post-deviation failure tails, and then prioritizes high-uncertainty states among the remaining rollout-start candidates.

Failure-tail truncation. Failed trajectories may contain long tails after an early action error makes recovery unlikely. Based on the observed association between high uncertainty and reduced action reliability, we conservatively set the cutoff at the first high-uncertainty state and truncate the subsequent tail. We define high uncertainty using the following dataset-level quantile threshold:

$$\kappa_{\text{fail}} = Q_{1-\rho_{\text{fail}}}\left(\{u_{\tau,t} \mid \tau \in \mathcal{D}_{\text{off}},\ 0 \le t \le T_\tau\}\right), \tag{3}$$

where Q_q denotes the empirical q-quantile, and $\rho_{\text{fail}} \in (0,1)$ controls the fraction of states regarded as high-uncertainty. For each failed trajectory $\tau \in \mathcal{D}_{\text{off}}$ with $y_\tau = 0$, we define its cutoff as the first timestep whose uncertainty reaches the threshold:

$$t_{\text{cut}}(\tau) = \begin{cases} \min\left\{t \mid 0 \le t \le T_\tau,\ u_{\tau,t} \ge \kappa_{\text{fail}}\right\}, & \text{if the set is nonempty,} \\ T_\tau, & \text{otherwise.} \end{cases} \tag{4}$$

We then define the retained rollout-start pool as

$$\begin{aligned} \mathcal{S}_{\text{ret}} = &\ \{s_{\tau,t} \mid \tau \in \mathcal{D}_{\text{off}},\ y_\tau = 1,\ 0 \le t \le T_\tau\} \\ &\cup \{s_{\tau,t} \mid \tau \in \mathcal{D}_{\text{off}},\ y_\tau = 0,\ 0 \le t \le t_{\text{cut}}(\tau)\}. \end{aligned} \tag{5}$$

Thus, all states from successful trajectories remain eligible, whereas each failed trajectory contributes only the prefix preceding its post-deviation tail.

Uncertainty-prioritized sampling. After truncating failure tails, we rank the states in $\mathcal{S}_{\text{ret}}$ by uncertainty. Let $\mathcal{S}_{\text{pri}} \subseteq \mathcal{S}_{\text{ret}}$ denote the top $\rho_{\text{pri}} \in (0,1]$ fraction of retained states. We define the prioritized rollout-start distribution as

$$\mu_{\text{pri}} = \text{Uniform}\left(\mathcal{S}_{\text{pri}}\right). \tag{6}$$

To broaden the state coverage of model-generated experience for PPO critic training, we retain a uniform component over all offline states:

$$\mu_{\text{cov}} = \text{Uniform}\left(\{s_{\tau,t} \mid \tau \in \mathcal{D}_{\text{off}},\ 0 \le t \le T_\tau\}\right). \tag{7}$$

The final rollout-start distribution is

$$\mu_{\text{roll}} = \lambda \mu_{\text{pri}} + (1-\lambda)\mu_{\text{cov}}, \qquad 0 \le \lambda \le 1, \tag{8}$$

where λ is the mixture weight for prioritized sampling. We initialize short-horizon branched rollouts at states sampled from μ_{roll} and use the resulting experience to update the policy and critic with PPO.

6 Experiments

We conduct extensive experiments on both simulated and real-world robotic manipulation tasks to answer the following questions: (1) Does U-GROW improve both sample efficiency and final policy performance in simulation? (2) Do these gains also hold in real-world manipulation? (3) How well do the resulting policies generalize to distribution shifts? (4) How does each component of U-GROW contribute to its overall performance?

6.1 Simulation Task Experiments

Benchmarks. We evaluate U-GROW on RoboTwin 2.0 (Chen et al., 2026) and LIBERO (Liu et al., 2023). RoboTwin 2.0 comprises 50 bimanual manipulation tasks across multiple robot embodiments and supports structured domain randomization over clutter, lighting, backgrounds, tabletop height, and language instructions. Our experiments focus on four of these tasks: *Beat Block Hammer* (BH), *Blocks Ranking RGB* (BR), *Handover Block* (HB), and *Move Can Pot* (MC). The LIBERO evaluation spans all four suites, covering language-conditioned manipulation with variations in spatial layouts, object types, task goals, and their combinations. We report success rates over 100 evaluation rollouts per RoboTwin 2.0 task and 500 evaluation rollouts per LIBERO suite.

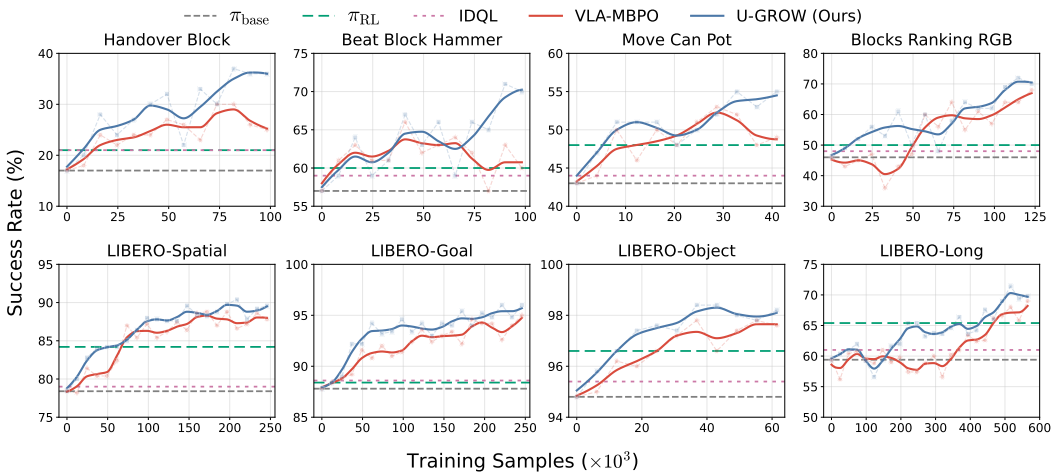

Figure 5: Success rate versus the cumulative number of training samples on RoboTwin 2.0 and LIBERO. Each training sample is a transition generated by the world model. Both methods use the same world model and total rollout budget; only the rollout-start distribution differs.

Table 1: Main results.

	RoboTwin 2.0					LIBERO				
Method	BH	BR	HB	MC	Avg.	Spatial	Object	Goal	Long	Avg.
π_{base}	57.0	46.0	17.0	43.0	40.8	78.4	94.8	87.8	59.4	80.1
π_{RL}	60.0	50.0	21.0	48.0	44.8	84.2	96.6	88.4	65.4	83.7
IDQL	59.0	48.0	21.0	44.0	43.0	79.0	95.4	88.6	61.0	81.0
VLA-MBPO	66.0	68.0	30.0	53.0	54.3	88.8	97.8	95.0	69.0	87.7
U-GROW (Ours)	**71.0**	**72.0**	**37.0**	**55.0**	**58.8**	**90.4**	**98.4**	**96.0**	**71.4**	**89.1**

Baselines. We compare U-GROW with four baselines: (1) π_{base}, the base policy obtained by supervised fine-tuning $\pi_{0.5}$ (Black et al., 2025a); (2) π_{RL}, an online RL method for flow-based VLA policies, trained under the same environment-interaction budget (Chen et al., 2025); (3) IDQL, an offline model-free RL baseline (Hansen-Estruch et al., 2023); and (4) VLA-MBPO, a model-based RL baseline that generates chunk-level branched rollouts from states sampled uniformly from the offline dataset (Zhang et al., 2026b). For a controlled comparison with VLA-MBPO, both methods use the same initial π_{base} checkpoint, offline dataset, world model, RL objective, rollout horizon, and total number of model-generated transitions; the only difference is the rollout-start distribution. Appendix C details the models, training settings, and hyperparameters.

U-GROW improves both sample efficiency and final policy performance. Under the same world model and rollout budget, U-GROW improves sample efficiency, generally achieving higher success rates with the same number of training samples across RoboTwin 2.0 and LIBERO (Figure 5). The advantage is particularly pronounced on long-horizon tasks such as LIBERO-Long and Handover Block. This pattern is consistent with the intuition that longer trajectories contain more intermediate states with uneven potential for policy improvement, making selective rollout allocation increasingly valuable. Table 1 further shows that U-GROW achieves the highest final success rates across all evaluated RoboTwin 2.0 tasks and LIBERO suites, with average gains of 4.5 and 1.4 percentage points over VLA-MBPO, respectively.

6.2 Real-world Task Experiments

We next examine whether the gains of U-GROW observed in simulation also hold in real-world manipulation, where noisy observations and complex contact dynamics pose additional challenges.

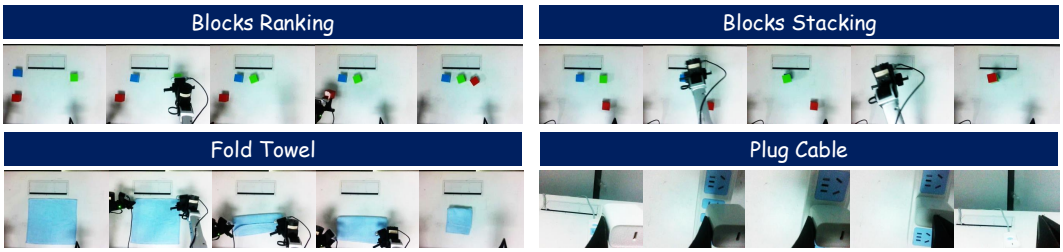

Figure 6: Real-world manipulation tasks on the Arx-X5 bimanual platform. *Blocks Ranking* requires arranging three colored blocks according to a language instruction, while *Blocks Stacking* requires stacking them in a specified order. *Fold Towel* involves bimanual folding of a deformable object, and *Plug Cable* requires sub-centimeter alignment to insert a cable into a socket.

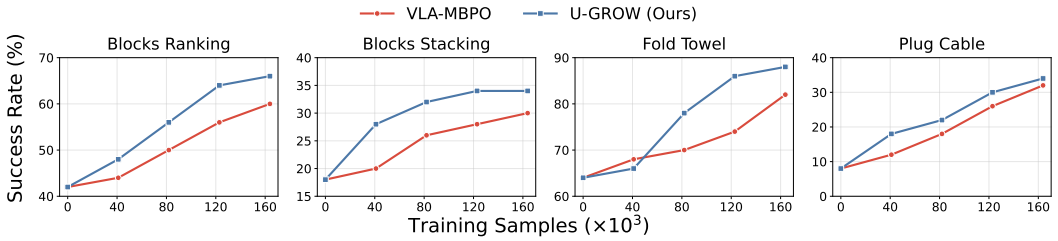

Figure 7: Success rate over the course of MBRL training on the four real-world tasks, evaluated on 50 trials per task.

Hardware and Tasks. We evaluate U-GROW on four real-world tasks using the Arx-X5 bimanual mobile manipulator, as illustrated in Figure 6. These tasks assess instruction following, contact interaction, manipulation of deformable objects, and precise insertion. (1) *Blocks Ranking* requires arranging three colored blocks according to a language instruction; (2) *Blocks Stacking* requires stacking the blocks in a specified order; (3) *Fold Towel* evaluates bimanual manipulation of a deformable object; and (4) *Plug Cable* requires sub-centimeter alignment to insert a cable into a 3-mm socket. For each task, we first collect 50 expert demonstrations for supervised fine-tuning of $\pi_{0.5}$, and then deploy the resulting SFT policy to collect 50 additional trajectories for world-model learning and subsequent MBRL training. Each policy is evaluated over 50 trials per task, comprising 30 under seen configurations and 20 under unseen configurations. Appendix B.2 details the hardware and control setup, data-collection procedures, and seen/unseen evaluation configurations.

Results. As shown in Figure 7, U-GROW learns more efficiently than VLA-MBPO and achieves higher final success rates on all four real-world tasks under the same world model and rollout budget. The gains are particularly evident on *Blocks Ranking* and *Fold Towel*. The improvements span rigid-object manipulation, deformable-object handling, and precise insertion, supporting the effectiveness of uncertainty-guided rollout allocation across different types of real-world interaction.

6.3 Generalization Evaluation

To assess the out-of-distribution generalization of U-GROW, we adopt a clean-to-randomized evaluation protocol on RoboTwin 2.0. All methods are trained in the clean setting and evaluated in unseen randomized environments, where variations in textures, distractor objects, lighting, and tabletop geometry introduce combined visual and geometric shifts. Additional evaluation details and representative scenes are provided in Appendix B.1.

Results. As shown in Figure 8, both MBRL methods, VLA-MBPO and U-GROW, outperform the SFT policy, π_{RL}, and IDQL across all four tasks in the randomized setting. Compared with VLA-MBPO, U-GROW further improves the average success rate from 24.0% to 26.0%, achieving higher success rates on three tasks and matching its performance on *Move Can Pot*. These results suggest

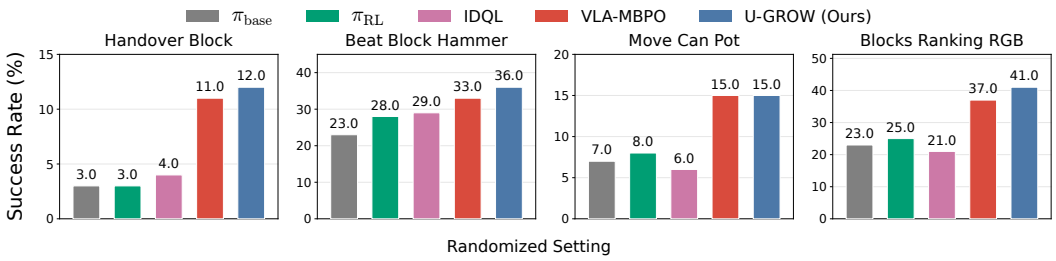

Figure 8: Success rates on the four RoboTwin 2.0 tasks in unseen randomized environments. Policies are trained in the clean setting and evaluated over 100 rollouts per task under variations in textures, distractor objects, lighting, and tabletop geometry.

that the benefits of model-based policy optimization extend beyond the clean training distribution, with additional gains from uncertainty-guided rollout allocation.

6.4 Ablation Study

To isolate the contributions of each component in U-GROW, we conduct an ablation study on the *Handover Block* task in RoboTwin 2.0 with three variants: (1) *w/o truncation*, which retains all states in failed trajectories; (2) *w/o prioritization*, which replaces the prioritized component with uniform sampling over the retained states; and (3) *w/o coverage*, which removes the uniform component over the full offline dataset.

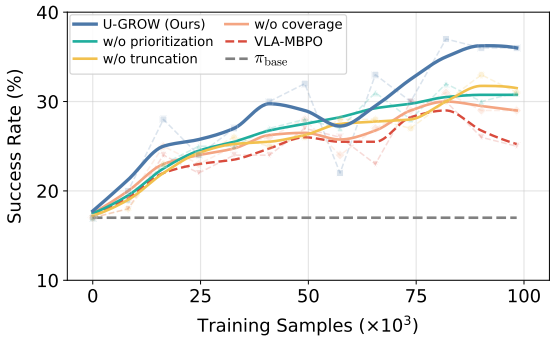

Figure 9: U-GROW ablation on *Handover Block*.

Results. As shown in Figure 9, the full method generally improves faster and achieves the highest final success rate. Without truncation, failure-tail states can absorb the prioritized rollout budget; without prioritization, the rollout budget is spread uniformly across retained states rather than concentrated on decision points with high uncertainty. Removing coverage produces the largest performance gap, possibly because restricting rollouts to a narrow high-utility subset reduces the state coverage available to the PPO critic, potentially weakening value estimation over the broader policy distribution and constraining sustained policy improvement. Together, the three components balance selective rollout allocation with the state coverage required for effective policy optimization.

7 Conclusion

In this work, we investigate which states offer greater potential for policy improvement and how to identify them. Our empirical analysis suggests that policy uncertainty can help identify such states: high uncertainty is concentrated in a small subset of visited states, often during decision-sensitive manipulation stages, where stronger-policy interventions yield greater gains in task success. Building on these findings, we introduce U-GROW, a lightweight rollout allocation method for model-based reinforcement learning. After failure-tail truncation, it prioritizes high-uncertainty states as starting points for short-horizon branched rollouts, while retaining uniform sampling over all offline states to preserve broad state coverage. Compared with uniform rollout-start sampling, U-GROW improves sample efficiency and final policy performance under the same world model and rollout budget. These results highlight the value of using policy uncertainty to direct model-generated experience toward states with greater potential for policy improvement.

AI USE STATEMENT

In this work, we used generative AI assistants for code generation, editing, and debugging to help implement our methods. We also used these assistants to translate text, assist with dataset reformatting, summarize and analyze existing literature, and retrieve information. Additionally, we used these assistants to polish the language and improve the clarity and readability of manuscript text written by the authors, rather than to draft manuscript content.

We did not use these assistants to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, design or provide feedback on the research methodology or experiments, conduct qualitative or thematic data analysis, or interpret experimental results. Tasks involving mathematical proofs, including providing key ingredients for proofs or assisting with proof writing, are not applicable to this work.

We reviewed all AI-assisted work. AI-assisted text was manually reviewed and revised by the authors, and AI-assisted code was inspected and tested against the intended implementation and experimental setup. We take responsibility for the final content of this work, including all text, claims, experimental results, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of our method and experimental protocols in Sections 5 and 6, with additional experimental settings and implementation details in Appendices A–C. These include computational resources, data collection and evaluation procedures, model training, policy optimization, and uncertainty-guided rollout-start selection. To further facilitate reproducibility, we will release our complete training and evaluation code upon acceptance.

REFERENCES

Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace B. Connors, James Darpinian, Karan Dhabalia, Jared Di Carlo, Danny Driess, Michael Robert Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Benjamin Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin Leblanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alexander Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: A VLA that learns from experience. In *Proceedings of the 22nd Robotics: Science and Systems (RSS'26), Sydney, Australia*, 2026.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning (CoRL'25), Seoul, South Korea*, 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of the 21st Robotics: Science and Systems (RSS'25), Los Angeles, CA*, 2025b.

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*, 2025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Proceedings of the 43rd International Conference on Machine Learning (ICML'26), Seoul, South Korea*, 2026.

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 *Proceedings of the 42nd IEEE International Conference on Robotics and Automation (ICRA'25), Atlanta, GA*, 2025.

Yanjiang Guo, Tony Lee, Lucy Xiaoyang Shi, Jianyu Chen, Percy Liang, and Chelsea Finn. VLAW: Iterative co-improvement of vision-language-action policy and world model. In *Proceedings of the 43rd International Conference on Machine Learning (ICML'26), Seoul, South Korea*, 2026.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit Q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Bohan Hou, Gen Li, Jindou Jia, Tuo An, Xinying Guo, Sicong Leng, Haoran Geng, Yanjie Ze, Tatsuya Harada, Philip Torr, Oier Mees, Marc Pollefeys, Zhuang Liu, Jiajun Wu, Pieter Abbeel, Jitendra Malik, Yilun Du, and Jianfei Yang. World model for robot learning: A comprehensive survey. *arXiv preprint arXiv:2605.00080*, 2026.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS'19), Vancouver, Canada*, 2019.

Zhennan Jiang, Shangqing Zhou, Yutong Jiang, Zefang Huang, Mingjie Wei, Yuhui Chen, Tianxing Zhou, Zhen Guo, Hao Lin, Quanlu Zhang, Yu Wang, Haoran Li, Chao Yu, and Dongbin Zhao. WoVR: World models as reliable simulators for post-training VLA policies with RL. *arXiv preprint arXiv:2602.13977*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning (CoRL'24), Munich, Germany*, 2025.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

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. In *Proceedings of the 14th International Conference on Learning Representations (ICLR'26), Rio de Janeiro, Brazil*, 2026.

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*, 2025.

Haoxin Lin, Yu-Yan Xu, Yihao Sun, Zhilong Zhang, Yi-Chen Li, Chengxing Jia, Junyin Ye, Jiaji Zhang, and Yang Yu. Any-step dynamics model improves future predictions for online and offline reinforcement learning. In *Proceedings of the 13th International Conference on Learning Representations (ICLR'25), Singapore*, 2025.

Haoxin Lin, Siyuan Xiao, Yi-Chen Li, Zhilong Zhang, Yihao Sun, Chengxing Jia, and Yang Yu. ADM-v2: Pursuing full-horizon roll-out in dynamics models for offline policy learning and evaluation. In *Proceedings of the 14th International Conference on Learning Representations (ICLR'26), Rio de Janeiro, Brazil*, 2026.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS'23), New Orleans, LA*, 2023.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can RL bring to VLA generalization? An empirical study. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS'25), San Diego, CA*, 2025.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. VLA-RL: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *Proceedings of the 14th International Conference on Learning Representations (ICLR'26), Rio de Janeiro, Brazil*, 2026.

NVIDIA. Cosmos 3: Omnimodal world models for physical AI. *arXiv preprint arXiv:2606.02800*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Haoxiang Ren, Yifei Sheng, Zhilong Zhang, Runjie Xu, Bowen Lu, Haoxin Lin, and Yang Yu. When to trust your teacher: Uncertainty-aware on-policy distillation for embodied policies. In *Workshop on Diffusion for Robot Learning at the 22nd Robotics: Science and Systems Conference (RSS'26), Sydney, Australia*, 2026.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yihao Sun, Jiaji Zhang, Chengxing Jia, Haoxin Lin, Junyin Ye, and Yang Yu. Model-Bellman inconsistency for model-based offline reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23), Honolulu, HI*, 2023.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Junjin Xiao, Yandan Yang, Xinyuan Chang, Ronghan Chen, Feng Xiong, Mu Xu, Wei-Shi Zheng, and Qing Zhang. World-Env: Leveraging world model as a virtual environment for VLA post-training. *arXiv preprint arXiv:2509.24948*, 2025.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS'20), Virtual Event*, 2020.

Yang Yu, Shiyuan Zhang, Yifei Sheng, Haoxiang Ren, and Haoxin Lin. How should world models be evaluated for embodied decision-making? A decision-making-centric position. *arXiv preprint arXiv:2606.15032*, 2026.

Zhilong Zhang, Ruifeng Chen, Junyin Ye, Yihao Sun, Pengyuan Wang, Jingcheng Pang, Kaiyuan Li, Tianshuo Liu, Haoxin Lin, Yang Yu, and Zhi-Hua Zhou. WHALE: Towards generalizable and scalable world models for embodied decision-making. *arXiv preprint arXiv:2411.05619*, 2024.

Zhilong Zhang, Yunpeng Mei, Xinghao Du, Hongjie Cao, Haonan Wang, Pengyuan Min, Chenyu Wang, Pengfei Chen, Chenbo Xin, Yijie Wang, Wenyu Luo, Yihao Sun, Yidi Wang, Lei Yuan, Gang Wang, and Yang Yu. Hierarchical value-decomposed offline reinforcement learning for whole-body control. In *Proceedings of the 14th International Conference on Learning Representations (ICLR'26), Rio de Janeiro, Brazil*, 2026a.

Zhilong Zhang, Haoxiang Ren, Yihao Sun, Yifei Sheng, Haonan Wang, Zhichao Wu, Haoxin Lin, Pierre-Luc Bacon, and Yang Yu. Towards practical world model-based reinforcement learning for vision-language-action models. In *Proceedings of the 43rd International Conference on Machine Learning (ICML'26), Seoul, South Korea*, 2026b.

Fangqi Zhu, Zhengyang Yan, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. WMPO: World model-based policy optimization for vision-language-action models. In *Proceedings of the 14th International Conference on Learning Representations (ICLR'26), Rio de Janeiro, Brazil*, 2026.

A Computational Resources

All training runs are conducted on 8 NVIDIA H100 GPUs. World-model training on the collected offline dataset takes approximately 40 hours per run, while subsequent model-based policy optimization takes approximately 5-6 hours per run under the specified model-rollout budget.

B Experimental setup

Figure 10 provides an overview of the manipulation tasks used in our experiments, with representative sequences from LIBERO, RoboTwin 2.0, and the Real-World Tasks setup.



Figure 10: Representative manipulation sequences across the three experimental settings.

B.1 Simulation experiments

We use four RoboTwin 2.0 tasks (Chen et al., 2026): Beat Block Hammer (BH), Blocks Ranking RGB (BR), Handover Block (HB), and Move Can Pot (MC). For LIBERO (Liu et al., 2023), we use four suites: Spatial, Object, Goal, and Long. For supervised fine-tuning (SFT), we use 100 and 40 expert trajectories per task in RoboTwin 2.0 and LIBERO, respectively. We also collect an offline dataset comprising 100 trajectories per RoboTwin task and 500 trajectories per LIBERO suite. This dataset is used to train the world model and provide starting states for branched rollouts during policy optimization. Each policy is evaluated over 100 episodes per RoboTwin 2.0 task in each of the clean and randomized settings, and over 500 episodes per LIBERO suite.

Generalization Evaluation Setup. We use a clean-to-randomized protocol on RoboTwin 2.0: policies are trained in the clean setting, and the same checkpoints are evaluated in unseen random-

ized environments. The randomized setting varies textures, distractor objects, lighting, and tabletop geometry. Figure 11 compares clean training scenes with representative randomized test scenes for all four tasks. Success rates in the random setting are reported in Figure 8.

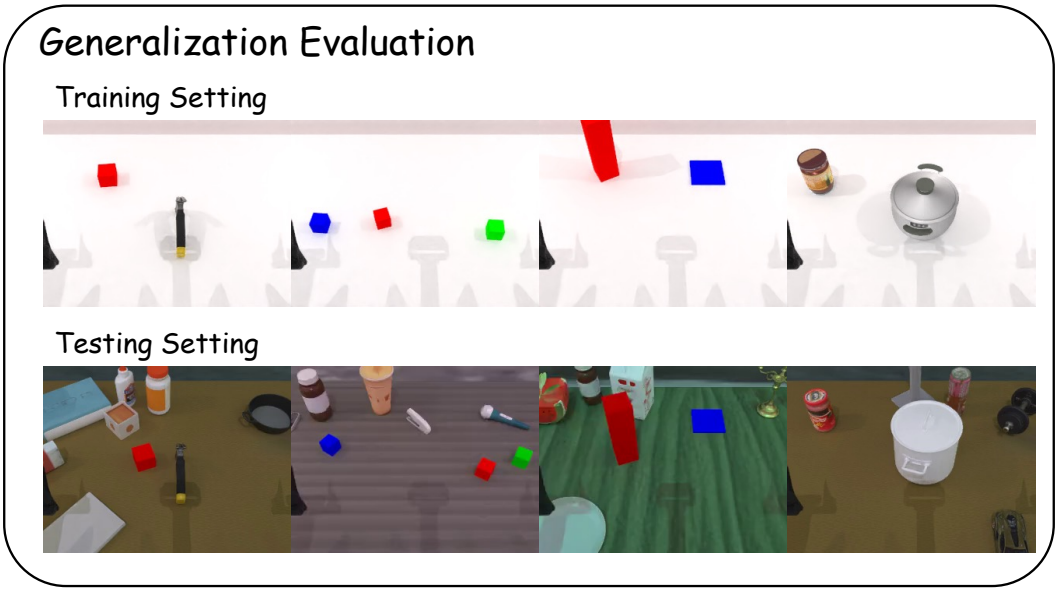

Figure 11: Clean-to-randomized evaluation on RoboTwin 2.0. The top row shows the clean training setting, and the bottom row shows representative randomized test scenes. From left to right, the columns correspond to Beat Block Hammer, Blocks Ranking RGB, Handover Block, and Move Can Pot. Policies trained in the clean setting are evaluated under visual and geometric shifts without additional training in the randomized environments.

B.2 Real-world experiments

Hardware and control. We use the Arx-X5 bimanual platform shown in Figure 12 for the real-world experiments. The platform has 14 degrees of freedom and is equipped with three Intel RealSense D435i RGB-D cameras for multi-view observations. Expert demonstrations are collected using a master–slave teleoperation setup, in which two handheld master arms control the corresponding robot arms. The system operates at a control frequency of 15 Hz.

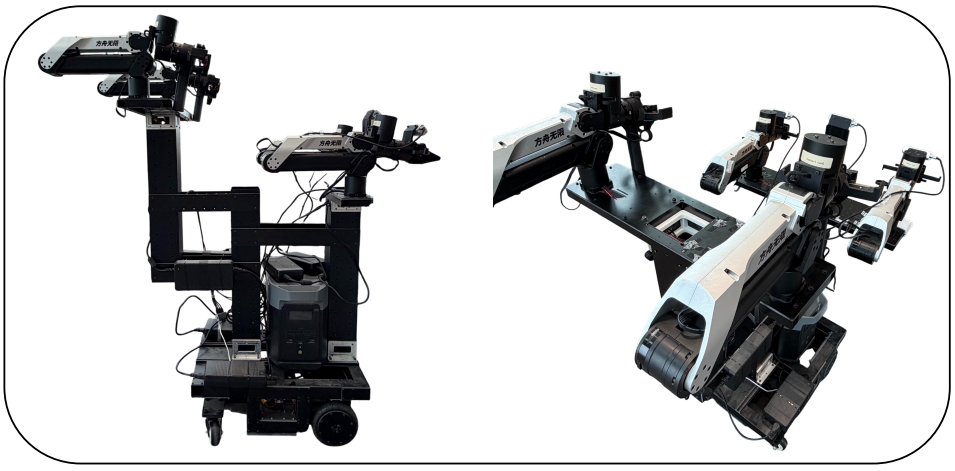

Figure 12: Arx-X5 platform for real-world data collection and policy evaluation.

Tasks. The lower panel of Figure 10 shows representative execution sequences for the four real-world tasks. *Blocks Ranking* requires arranging three colored blocks according to a language instruction, and *Blocks Stacking* requires stacking blocks in a specified order. *Fold Towel* involves folding a towel through coordinated bimanual manipulation, while *Plug Cable* requires aligning a cable plug with a socket and inserting it.

Data collection and evaluation. For each task, we collect 50 expert demonstrations for SFT and then deploy the resulting policy to collect 50 additional trajectories. We use these trajectories to train the world model and initialize branched rollouts during policy optimization. Each policy is evaluated over 50 trials per task, comprising 30 under seen configurations and 20 under unseen configurations. The unseen configurations introduce task-specific variations: language instructions specifying unseen block orders in *Blocks Ranking* and *Blocks Stacking*, unseen towel colors in *Fold Towel*, and changes in object positions in *Plug Cable*. Figure 13 shows the objects used in the seen configurations and the two held-out towels used for unseen evaluation in *Fold Towel*. Figure 7 reports success rates over all 50 trials per task.

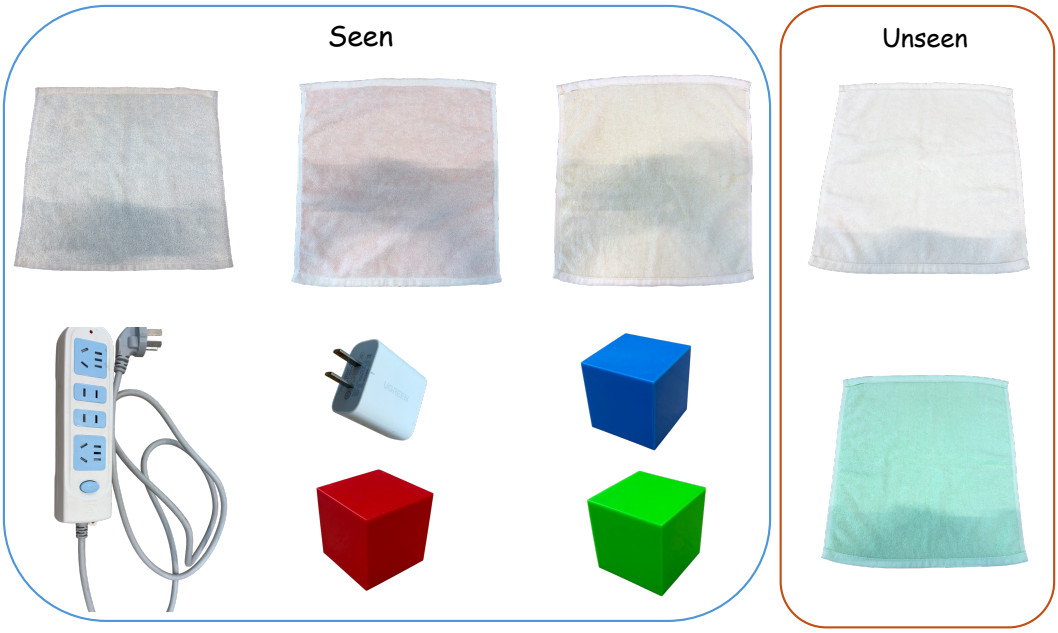

Figure 13: Objects used in the real-world experiments. **Left (seen):** three towels for *Fold Towel*, colored blocks for *Blocks Ranking* and *Blocks Stacking*, and a power strip and plug adapter for *Plug Cable*. **Right (unseen):** two held-out towels for evaluating *Fold Towel* on unseen object instances.

C Implementation Details

C.1 World Model

We adapt the pretrained Wan2.2-TI2V-5B video diffusion model (Team Wan et al., 2025) to predict K future observations conditioned on five context observations, five historical actions, and a policy action chunk of length K. At each time step, camera views are stacked vertically into a single image. The VAE encodes the five context images into two temporal latent slices, which are assigned diffusion timestep zero and remain fixed throughout denoising.

Action conditioning. Actions condition every DiT block through two pathways. For *action cross-attention*, a two-layer MLP embeds each action, and video tokens attend to the resulting embeddings. For *action-dependent modulation*, we prepend three zero actions to the action sequence and concatenate each consecutive group of four actions to match the VAE's temporal compression. A separate

MLP maps each group to an embedding, which is combined with the diffusion-timestep embedding to control normalization and residual gating in the DiT blocks, as well as output-head normalization.

Training. We initialize three separate world models from the base Wan weights: one trained jointly across the four LIBERO suites, one across the four RoboTwin tasks, and one across the four real-world tasks. For each model, we fine-tune the full DiT and both action MLPs while keeping the VAE frozen, using a timestep-weighted flow-matching MSE loss over future video latents. Table 2 lists the training and rollout settings.

Table 2: World model training and rollout settings.

Setting	RoboTwin 2.0	LIBERO	Real-World Tasks
Image resolution	256×256	256×256	256×256
Context frames	5	5	5
Executed action chunk length (K)	20	8	20
Video sequence length	25	13	25
Denoising steps	5	5	5
Guidance scale	1.0	1.0	1.0
Noise-schedule shift	5.0	5.0	5.0
Learning rate	10^{-5}	10^{-5}	10^{-5}
Batch size	256	256	256

C.2 Reward model

We fine-tune Qwen3.5-0.8B (Qwen Team, 2026) to predict task completion with `Yes`/`No` responses. We train a separate reward model for each LIBERO suite and each RoboTwin or real-world task, and keep all reward models fixed during policy optimization.

Inputs and supervision. Each example pairs a task instruction with visual observations from a single time step. LIBERO training uses the main-camera image. RoboTwin and real-world training use three separate images, ordered as the head, left-wrist, and right-wrist views, in a single multimodal prompt. Labels are derived from trajectory outcomes: for successful trajectories, the final five frames are labeled `Yes`, and earlier frames are labeled `No`. All frames from failed trajectories are negative. We balance the two classes by oversampling positive examples to match the negatives.

Training. We fine-tune the language model and visual alignment module while freezing the vision encoder. The objective is autoregressive cross-entropy over the assistant's answer tokens, with prompt tokens masked from the loss. Table 3 reports the optimization settings.

Table 3: Reward model training settings.

Setting	RoboTwin	LIBERO	Real-World Tasks
Learning rate	2×10^{-5}	2×10^{-5}	2×10^{-5}
Weight decay	0.1	0.1	0.1
Adam (β_1, β_2)	(0.9, 0.95)	(0.9, 0.95)	(0.9, 0.95)
Adam ϵ	10^{-8}	10^{-8}	10^{-8}
Learning-rate schedule	Cosine	Cosine	Cosine
Warmup fraction	0.05	0.05	0.05
Gradient norm clipping	1.0	1.0	1.0
Global batch size	256	256	256
Maximum sequence length	4096	4096	4096
Precision	BF16	BF16	BF16

Rollout rewards. During model rollouts, the reward model evaluates each of the K generated observations independently, conditioned on the task instruction and the camera views at that time

step. Let $z_{c,j}^{\mathrm{Yes}}$ and $z_{c,j}^{\mathrm{No}}$ denote the logits of the `Yes` and `No` tokens at the first answer position for time step j of action chunk c. We normalize these two logits to obtain the completion score

$$p_{c,j} = \frac{\exp(z_{c,j}^{\mathrm{Yes}})}{\exp(z_{c,j}^{\mathrm{Yes}}) + \exp(z_{c,j}^{\mathrm{No}})}.$$

A chunk is predicted successful if any generated observation has $p_{c,j} \geq 0.5$. We assign its binary reward to the final step, with zero rewards at all preceding steps:

$$r_{c,j} = \mathbf{1}[j = K]\, \mathbf{1}\left[\max_{1 \leq m \leq K} p_{c,m} \geq 0.5\right].$$

An episode-level success flag remains set once any chunk is predicted successful and is cleared on reset. Rewards are computed independently for each chunk.

C.3 Policy optimization

We update the policy and critic with PPO using chunk-level rewards and log probabilities. The critic takes detached VLM features as input. We compute advantages using generalized advantage estimation (GAE) and normalize them within each task. Table 4 lists the optimization settings.

Table 4: Policy optimization settings.

Setting	RoboTwin	LIBERO	Real-World Tasks
Action chunk	20	8	20
Action dimension	14	7	14
Denoising steps	5	3	5
Rollout chunk	2	3	2
Actor learning rate	2.5×10^{-6}	5×10^{-6}	2.5×10^{-6}
Critic learning rate	5×10^{-5}	10^{-4}	5×10^{-5}
Batch size	1024	1024	1024
PPO update epochs	5	10	5
Discount Rate	0.99	0.99	0.99
GAE	0.95	0.95	0.95
Noise level	0.5	0.3	0.5
Adam (β_1, β_2)	(0.9, 0.95)	(0.9, 0.95)	(0.9, 0.95)

C.4 Uncertainty-Guided Rollout Starts

Before generating rollouts in each training iteration, we recompute uncertainty scores for all candidate offline states under the current policy using the estimator in Section 3.3. We then update the rollout-start distribution using these scores. Table 5 lists the default hyperparameters for uncertainty estimation and rollout-start selection, with sensitivity to selected hyperparameters analyzed in Appendix D.1.

Table 5: Uncertainty-guided rollout-start settings.

Setting	RoboTwin	LIBERO	Real-World Tasks
Uncertainty norm order (p)	2	2	2
Denoising step counts ($\mathcal{K}$)	$\{3, 5\}$	$\{3, 5\}$	$\{3, 5\}$
Truncation parameter (ρ_{fail})	0.1	0.1	0.1
Prioritized fraction (ρ_{pri})	0.2	0.2	0.2
Prioritized sampling weight (λ)	0.5	0.5	0.5

D Additional Results

D.1 Hyperparameter Sensitivity

We examine the sensitivity of U-GROW to three hyperparameters on *Handover Block*: the failure-tail truncation parameter ρ_{fail}, the prioritized fraction ρ_{pri}, and the mixing weight λ. We vary one hyperparameter at a time while keeping the others at their default values in Table 5. All configurations use the same initial policy checkpoint, offline dataset, world model, and total rollout budget. Figure 14 shows the success rates across the tested configurations.

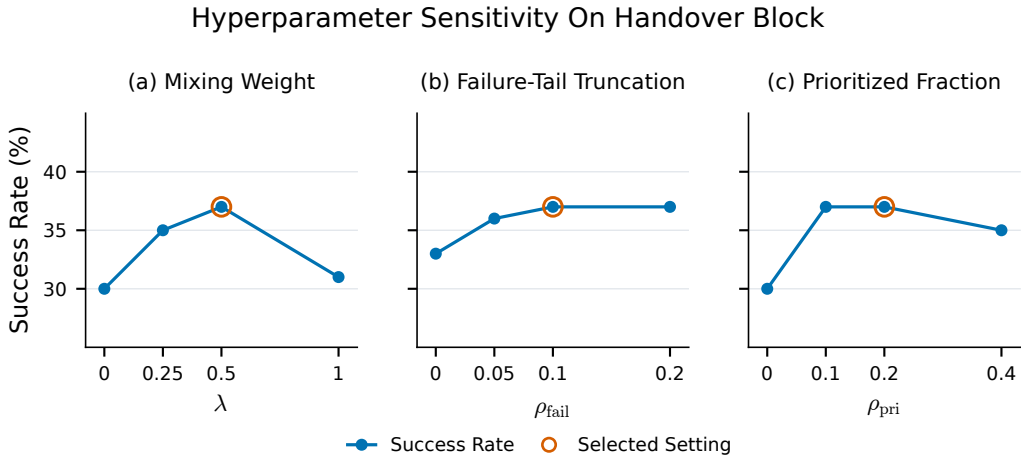

Figure 14: Hyperparameter sensitivity on *Handover Block*. Each point reports the success rate from a single training run. Orange circles mark the default settings used in the main experiments.

As shown in Figure 14, performance changes only modestly across the tested settings near the defaults. In contrast, setting λ to either endpoint causes a larger decline, suggesting the importance of balancing prioritization with broad state coverage.