WoVR: World Models as Reliable Simulators for Post-Training VLA Policies with RL

Zhennan Jiang[2,3,4,*] **Shangqing Zhou**[2,3,*] **Yutong Jiang**[3] **Zefang Huang**[4] **Mingjie Wei**[4]
Yuhui Chen[2,3] **Tianxing Zhou**[4] **Zhen Guo**[5] **Hao Lin**[5] **Quanlu Zhang**[5]
Yu Wang[1] **Haoran Li**[2,3,†] **Chao Yu**[1,†] **Dongbin Zhao**[2,3,4,†]

[*]Equal contribution [†]Corresponding authors

[1]Tsinghua university [2]University of Chinese Academy of Sciences
[3]Institute of Automation, Chinese Academy of Sciences [4]Zhongguancun Academy [5]Infinigence AI

https://huggingface.co/collections/RLinf/rlinf-wovr https://github.com/RLinf/RLinf

Abstract: Reinforcement learning (RL) promises to unlock capabilities beyond imitation learning for Vision–Language–Action (VLA) models, but its requirement for massive real-world interaction prevents direct deployment on physical robots. Recent work attempts to use learned world models as simulators for policy optimization, yet closed-loop imagined rollouts inevitably suffer from hallucination and long-horizon error accumulation. Such errors not only degrade visual fidelity, but also mislead policy optimization by providing unreliable learning signals. We propose **WoVR**, a reliable world-model-based RL framework for post-training VLA policies. Instead of assuming a faithful world model, WoVR explicitly regulates how RL interacts with imperfect imagined dynamics. It improves rollout stability through a controllable action-conditioned video world model, reshapes imagined interaction to reduce effective error depth via Keyframe-Initialized Rollouts, and maintains policy–simulator alignment through World Model-Policy co-evolution. Extensive experiments demonstrate that WoVR enables stable long-horizon imagined rollouts and effective policy optimization, achieving superior LIBERO performance and consistent real-world gains across multiple robotic platforms. These results show that world models can serve as practical simulators for RL when hallucination is explicitly controlled. Additional visualization results are available at https://wovr-rlinf.github.io/.

Keywords: World Model, Reinforcement Learning, Vision-Language-Action

1 Introduction

Vision–Language–Action (VLA) models [1, 2, 3, 4] have been increasingly adopted for robotic manipulation, where actions are generated end-to-end by conditioning on language instructions and visual observations. Most existing VLA systems are trained via imitation learning. While effective in many downstream tasks, this paradigm fundamentally limits the performance ceiling of VLA policies, as it is tightly constrained by the quality and coverage of demonstration data.

Recent VLA-RL methods [5, 6, 7, 8, 9, 10, 11] improve beyond imitation learning [12]. However, applying RL to real-world VLA policies presents a fundamental dilemma: off-policy methods are generally more sample-efficient but often suffer from distribution shift and training instability, whereas on-policy updates [13, 14, 15] require massive parallel environment interaction for stable and efficient training, making them impractical for real robots due to costly data collection [16, 17]. Although simulation-based alternatives have been explored [7, 18, 19], aligning simulators with real-world dynamics remains highly challenging. These constraints motivate replacing real-environment interaction with a learned world model that serves as a simulator for policy optimization.

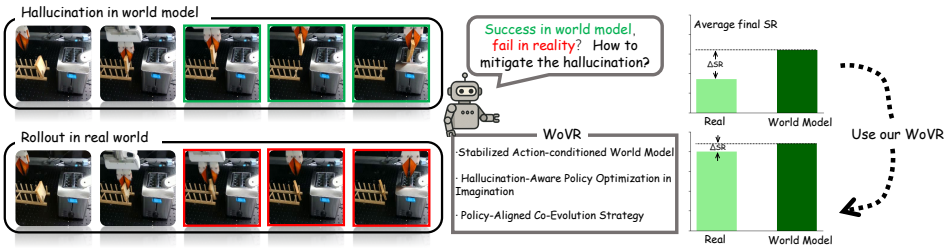


Figure 1: **Hallucination in Closed-Loop World Model Rollouts.** The world model imagines a successful grasp (green frames), but real-world execution fails (red frames).To address this critical mismatch, we propose three hallucination-aware mechanisms.

This direction has become increasingly viable with the emergence of large-scale generative video models [20, 21]. Several works directly treat pretrained video generators as simulators and perform reinforcement learning entirely in imagination [22, 23]. **However, learned world models are not faithful simulators.** In this work, we define *hallucination* as a systematic mismatch between imagined and real outcomes in closed-loop interaction. The world model may generate visually plausible yet physically incorrect rollouts, resulting in spurious success signals under the policy's actions. In closed-loop autoregressive rollouts, prediction errors compound with horizon length due to:

- Autoregressive feedback: the model conditions on its own generated frames, amplifying small early errors;
- Distribution shift: as the policy evolves, its action distribution drifts away from the data used to train the world model, increasing out-of-distribution prediction failures.

If hallucinated trajectories are directly used for policy optimization, RL is incentivized to exploit systematic model errors rather than true task progress. This leads to a critical question:

If world models inevitably hallucinate, how can RL remain reliable under imperfect imagined dynamics?

We argue that using world models for RL is not primarily a modeling problem, but a reliability problem. To make world-model-based RL reliable, one must control hallucination at three interconnected levels: controllable simulator design, reliable interaction protocol, and policy–model alignment. To this end, we propose **WoVR**, a **Wo**rld-model-based framework for post-training **V**ision–Language–Action policies with **R**einforcement Learning. Rather than assuming the learned world model to be a faithful simulator, WoVR explicitly regulates how reinforcement learning interacts with imperfect imagined dynamics. We first strengthen the simulator itself by constructing a rollout-stable, action-controllable video world model with stabilized autoregressive context modeling, reducing long-horizon drift and structural collapse. To mitigate long-horizon prediction-error accumulation, we further introduce *Keyframe-Initialized Rollouts (KIR)*, which start imagined rollouts from task-critical states, shortening the effective prediction depth and reducing hallucination compounding. Finally, as policy optimization shifts the action distribution and induces distribution mismatch between the policy and the world model, we introduce *PACE*, a policy-aligned co-evolution strategy that restores alignment by iteratively refining the world model under the evolving policy distribution, without requiring continuous online supervision. Together, these components form a unified hallucination-aware reinforcement learning framework that enables reliable policy optimization in imagination. In summary, our contributions are as follows.

- We identify hallucination under closed-loop imagined interaction as a fundamental reliability challenge in world-model-based RL for VLA, showing that autoregressive error accumulation and policy-induced distribution shift can systematically corrupt optimization signals.
- We propose WoVR, a hallucination-aware RL framework that jointly regulates controllable simulator design, reliable imagined interaction, and a policy-aligned co-evolution strategy, enabling stable on-policy optimization entirely in imagination.

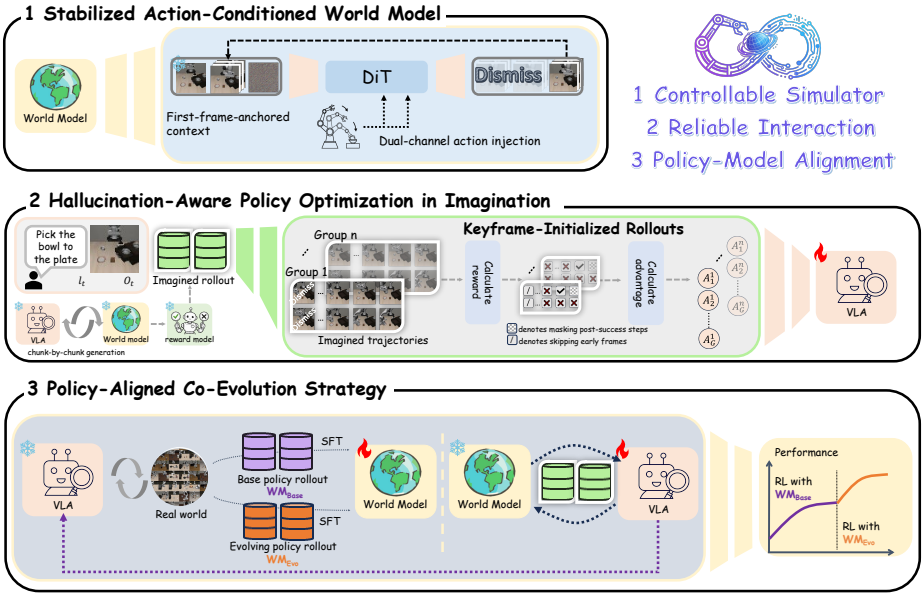


Figure 2: **Overview of WoVR**. WoVR builds a reliability-driven reinforcement learning framework entirely around the learned world model. It first strengthens the world model as a controllable simulator, ensuring rollout-stable and action-responsive generation. On top of this simulator, it designs a reliable interaction protocol via Keyframe-Initialized Rollouts (KIR) to reduce effective error depth and prevent optimization on hallucinated success. Finally, it maintains policy–model alignment through PACE, which co-evolves the world model with the evolving policy to mitigate distribution shift and preserve simulator reliability.

- WoVR achieves state-of-the-art world-model quality with strong perceptual and temporal consistency while maintaining high rollout efficiency at 23 FPS. More importantly, it consistently improves policy performance in both simulation and real-world deployment, boosting LIBERO success under different SFT initializations and improving success rates across multiple robotic platforms.

2 Related Work

2.1 RL Fine-tuning for VLA Models

On-policy reinforcement learning [5, 13] has been increasingly adopted to fine-tune VLA models beyond imitation learning [5, 6]. However, directly transferring on-policy fine-tuning to real robots remains impractical, as such methods require large-scale parallel rollouts, repeated environment resets, and tightly coupled policy–environment interaction, which are difficult to support under real-world hardware. To mitigate this, some off-policy approaches [8, 24] introduce offline data reuse or human intervention, but often suffer from limited scalability and performance degradation during online updates. An alternative direction builds large-scale real-robot infrastructures, yet existing systems [25, 26] still cannot practically support fully on-policy algorithms at scale. These limitations suggest that the challenge of online RL for VLA is systemic rather than algorithmic, motivating world-model-based approaches that decouple policy optimization from real-world interaction.

2.2 World Models for Policy Optimization

A growing body of work has begun to incorporate learned world models into VLA policy optimization [27, 28, 29]. One line of work uses world models primarily as reward or evaluation signals for post-training, where predicted future outcomes are used to construct preferences or guide policy improvement [30, 31, 32, 33, 34]. Another line improves VLA policies by generating synthetic rollouts with learned world models, reducing reliance on costly real-world interaction [35, 36, 37]. However, these works stop short of using world models as closed-loop RL simulators for policy optimization.

Prophet [38], World-Env [39] and WMPO [22] move in this direction by replacing real-environment interaction with imagined rollouts during reinforcement learning. Yet both largely treat the world model as a drop-in simulator, without explicitly addressing the central challenge of hallucinated dynamics: in closed-loop rollouts, prediction errors accumulate and can be exploited by the policy, ultimately misleading optimization.

3 Methods

We propose WoVR, a reliability-driven world-model-based reinforcement learning framework for post-training VLA policies. As illustrated in Fig. 2, WoVR treats the world model as a generative simulator and controls hallucination across three levels: (1) Simulator-level control: we construct an action-controllable, rollout-stable video world model with dual-channel action injection and first-frame anchoring to suppress long-horizon drift. (2) Interaction-level reshaping: we redesign imagined interaction through Keyframe-Initialized Rollouts (KIR) to reduce effective error depth and prevent optimization on hallucinated success. (3) Alignment-level regulation: we introduce PACE, a policy–model co-evolution strategy that mitigates distribution shift by periodically aligning the world model with the evolving policy.

3.1 Stabilized Action-Conditioned World Model

WoVR relies on a learned video world model as a generative simulator for closed-loop imagined interaction. However, long-horizon autoregressive generation can accumulate hallucinations, causing scene drift and appearance degradation. We therefore design the world model to be both *action-controllable* and *rollout-stable*, so that the simulated dynamics remain consistent under iterative, policy-driven generation.

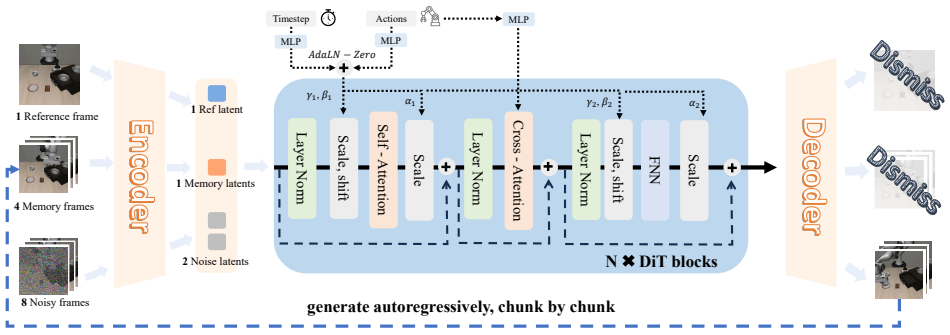


Figure 3: **Architecture of the proposed action-conditioned world model.** The world model is built upon a video diffusion backbone and conditioned on actions via a dual-channel action injection design, enabling frame-level controllability and stable chunk-by-chunk autoregressive generation for long-horizon imagined rollouts.

Our world model is built on the Wan2.2-TI2V-5B video diffusion backbone [21] and reformulated as an action-conditioned generator. As shown in Fig. 3, actions are injected through two complementary pathways: they modulate denoising features through timestep-conditioned normalization and replace text embeddings in cross-attention to provide global action context. This *dual-channel design* preserves the original DiT structure while enabling frame-level control.

To stabilize closed-loop rollout, we additionally use a *first-frame–anchored* context. At each autoregressive step, the model conditions on $[o_0,\ o_{t-c:t}]$, combining the initial frame with recent memory frames from the previous chunk. The fixed reference frame constrains global layout and appearance, while the memory frames preserve local dynamics, reducing drift and background collapse in long-horizon generation [40, 41, 42].

During training, we apply *noisy context augmentation*: non-reference context frames are mildly noised, while the first-frame anchor remains clean. This makes the model robust to self-generated context at inference time and reduces brittle copying from previous predicted frames.

Together, dual-channel action conditioning, first-frame anchoring, and noisy context augmentation turn the video generator into a rollout-stable simulator for imagined RL. Given anchored context and policy actions, the model autoregressively predicts and appends video chunks, producing long-horizon trajectories entirely in imagination. More details are provided in Appendix B.1.

When used as an RL simulator, the world model must also provide a reward signal. We support two modeling choices: a lightweight ResNet-based model for binary rewards, and a Qwen3-VL-based model for dense rewards. In practice, we find that the ResNet-based model is significantly more time-efficient while achieving comparable performance. Details are given in Appendix B.2.

3.2 Hallucination-Aware Policy Optimization in Imagination

WoVR optimizes the VLA policy by interacting with the learned world model, which serves as a generative simulator for closed-loop imagined rollouts. The key difficulty is that, in long-horizon rollouts starting from the initial state, world-model errors accumulate early and can eventually produce visually plausible but physically incorrect transitions and even spurious success signals. If reinforcement learning naively trusts such rollouts, the policy is encouraged to optimize toward hallucinated outcomes rather than real task progress.

To reduce the effective error depth of imagined interaction, we introduce Keyframe-Initialized Rollouts (KIR). As illustrated in Figure 4, instead of always initializing rollouts from the episode start o_0, we initialize a portion of rollouts from keyframes o_k that lie near task-critical intermediate states, especially failure states encountered by the current policy. The motivation is that many decisive contacts and corrections happen locally around these states, whereas starting from o_0 forces the world model to predict a long prefix before reaching them, during which compounding errors can already derail the rollout.

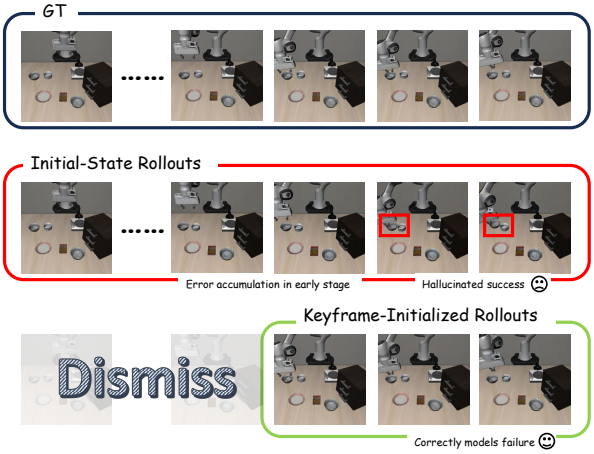


Figure 4: **Illustration of Keyframe-Initialized Rollouts (KIR).** Long-horizon rollouts accumulate prediction errors in early stages, leading to hallucinated success that contradicts the ground-truth failure. In contrast, KIR initialize rollouts near critical states, enabling physically consistent predictions that correctly model failure.

We adopt GRPO to update the policy using imagined rollouts. The GRPO objective is defined as:

$$J_{\mathrm{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i^{\mathrm{valid}}}\sum_{t=1}^{T_i^{\mathrm{valid}}}\min\Big(\rho_t^{(i)}(\theta)\hat{A}^{(i)}, \mathrm{clip}\big(\rho_t^{(i)}(\theta),\,1-\epsilon,\,1+\epsilon\big)\hat{A}^{(i)}\Big)\right], \quad (1)$$

where G is the number of imagined trajectories, $\hat{A}^{(i)}$ is the group-relative advantage. and T_i^{valid} is the number of valid timesteps up to the first success. This objective also complements KIR: keyframe-initialized rollouts tend to reach task resolution with fewer valid steps, and trajectory-length normalization increases their per-timestep contribution, so gradients are dominated by short, task-critical segments rather than long, drift-prone continuations.

3.3 PACE: Policy–Aligned Co-Evolution

While policy optimization proceeds entirely within the learned world model, the policy's action distribution continuously evolves and drifts away from the data used to train the initial world model. This inherent distribution shift leads to accumulating mismatch between the simulator and the improving policy, ultimately degrading the reliability of imagined rollouts.

To address this issue, we introduce **PACE**, a World Model–Policy co-evolution strategy. Instead of treating the world model as a fixed, static simulator throughout policy optimization, PACE allows the world model and VLA policy to evolve together throughout training.

Concretely, we realize this co-evolution through low-frequency, policy-driven refinement: we first train an initial world model, denoted as $\mathrm{WM}_{\mathrm{Base}}$, using trajectories collected from the base VLA policy. After the first stage of policy optimization within $\mathrm{WM}_{\mathrm{Base}}$, we collect a limited set of additional rollouts under the evolved policy and use them to further refine the world model. The refined model is referred to as $\mathrm{WM}_{\mathrm{Evo}}$. Importantly, this refinement is performed at very low frequency, distinguishing *PACE* from classical model-based reinforcement learning methods, which continuously update the dynamics model at high frequency during policy optimization. This low-frequency refinement provides two key advantages. First, unlike real-world online RL, it does not require continuous human supervision or environment resets during policy training, significantly reducing operational overhead. Second, by aligning the world model with the evolving policy distribution, *PACE* mitigates compounding model errors and maintains simulator reliability without sacrificing training stability.

4 Experiments

We conduct extensive experiments to evaluate the effectiveness of WoVR as a world-model-based reinforcement learning framework for post-training VLA policies. Our experimental design aims to systematically answer the following three questions:

- **Q1:** Is the proposed world model stable, controllable, and efficient enough to serve as a simulator for closed-loop reinforcement learning?
- **Q2:** Can WoVR effectively improve VLA task performance compared to existing world-model-based reinforcement learning methods?
- **Q3:** Do the policies optimized with WoVR reliably transfer to real-world manipulation tasks?

To answer these questions, we evaluate both the quality of the learned world model and the downstream policy performance. For world model evaluation, we focus on long-horizon, action-conditioned video generation under closed-loop, chunk-by-chunk autoregressive inference. We adopt standard perceptual and distributional metrics, including **LPIPS** [43], **FID** [44], **FVD** [45] and **FloLPIPS** [46] (Detailed calculations are provided in the Appendix C.). We also report inference throughput (**FPS**) to quantify generation efficiency. For policy evaluation, we use task success rate (**SR**) as the primary metric.

We compare WoVR against several representative baselines spanning both *world model quality* and *policy optimization*. For world model quality, we include EVAC [28], Cosmos-Predict2 [47] and OpenSora [48](the world-model backbone adopted in WMPO [22]). All compared models are evaluated under the same chunk-wise autoregressive generation protocol to ensure a fair comparison. For policy optimization, we consider OpenVLA-OFT [1], a base VLA policy trained purely with imitation learning; GRPO (Online) [15], trained with real-environment interaction under the same rollout budget; and WMPO [22], which performs reinforcement learning using OpenSora.

4.1 Is the World Model Stable, Controllable, and Efficient?

Experimental Setup. We conduct all world model evaluations in the LIBERO environment [49]. A total of 3,000 VLA rollout trajectories, each with a length of 512 frames, are collected to train the world models. In addition, 200 held-out trajectories of the same length are used exclusively for evaluation. We compare WoVR against three representative action-conditioned world models: EVAC, Cosmos-Predict2, and OpenSora as adopted in WMPO.

Quantitative Results. Table 1 shows that WoVR outperforms all baselines across all metrics, indicating higher visual fidelity, stronger temporal consistency, and more accurate dynamics. These

Table 1: **Comparison of different world models.** Rollout denotes the rollout horizon length.

Method	Rollout	FPS ↑	LPIPS ↓ [43]	FID ↓ [44]	FVD ↓ [45]	FloLPIPS ↓ [46]
EVAC [28]	512	1.35	0.146	46.528	345.818	0.205
	256		0.130	49.153	354.983	0.192
	128		0.106	44.337	423.132	0.166
Cosmos-Predict2 [47]	512	3.50	0.315	165.862	275.737	0.265
	256		0.226	106.324	203.853	0.306
	128		0.164	77.555	304.456	0.281
OpenSora [22]	512	7.00	0.105	38.478	89.391	0.156
	256		0.082	33.577	94.998	0.122
	128		0.069	33.413	111.643	0.113
WoVR (Ours)	512	**23.0**	**0.091**	**34.252**	**68.011**	**0.154**
	256		**0.063**	**24.378**	**50.041**	**0.102**
	128		**0.047**	**18.553**	**39.047**	**0.079**

Table 2: **Task success rates (%) across LIBERO suites.**

Method	Spatial	Object	Goal	Long	Avg
One-Trajectory SFT					
OpenVLA-OFT [1]	63.6	36.4	48.2	13.8	40.5
w/ GRPO (online) [15]	66.6	45.2	52.2	14.6	44.6
w/ WMPO [22]	67.8	65.4	56.6	13.8	50.9
w/ **Ours**	**84.2**	**80.8**	**77.4**	**35.8**	**69.5**
Δ	+20.6	+44.4	+29.2	+22.0	+29.0
Full-Trajectory SFT					
OpenVLA-OFT [1]	93.6	83.0	90.0	85.6	88.1
w/ GRPO (online) [15]	94.6	86.2	92.2	85.8	89.7
w/ WMPO [22]	95.0	94.8	92.8	87.0	92.4
w/ **Ours**	**98.8**	**98.8**	**94.8**	**91.4**	**96.0**
Δ	+5.2	+15.8	+4.8	+5.8	+7.9

improvements become more pronounced as the rollout horizon increases, suggesting that WoVR is more robust in long-horizon autoregressive generation.

Despite adopting a larger backbone (Wan, ∼5B) than OpenSora (∼1.3B), WoVR achieves higher inference throughput by requiring only five diffusion steps and leveraging a 3D VAE for spatiotemporal latent encoding, whereas OpenSora typically relies on more sampling steps and a 2D VAE.

4.2 Can WoVR Effectively Improve VLA Task Performance?

Experimental Setup. We conduct policy optimization experiments on LIBERO Spatial, Object, Goal, and Long suites [49], each containing 10 tasks. Following SimpleVLA-RL [18], we initialize from OpenVLA-OFT and consider two supervised fine-tuning settings: *one-trajectory SFT* and *full-trajectory SFT*.

For a fair comparison, all methods use the same budget of 2,500 real-environment trajectories per suite. **GRPO** directly uses them for online policy optimization, while **WMPO** uses them to train a world model and then performs RL entirely in imagination. **WoVR** follows the two-stage policy–simulator co-evolution protocol in Sec. 3.3: it first trains an initial world model with 1,500 trajectories collected by the base policy, optimizes the policy in imagination, and then refines the world model with 1,000 additional trajectories collected by the evolved policy.

Quantitative Results. Table 2 reports **SR** across LIBERO suites under two SFT initializations. Under the *one-trajectory SFT* setting, GRPO yields only a marginal improvement (4.1 % ↑), indicating

Table 3: **Real-world success rates (%, 30 trials/task).**

Method	Franka Emika Panda				AgileX Piper			
	Pick Banana	Pick Bread	Open Drawer	**Avg**	Fold Towel	Pick Cube	Pick Tomato	**Avg**
OpenVLA-OFT [1]	36.7	70.0	46.7	51.1	10.0	23.3	13.3	15.5
w/ WMPO [22]	56.7	76.7	60.0	64.5	10.0	26.7	20.0	18.9
w/ **Ours**	**86.7**	**90.0**	**63.3**	**80.0**	**20.0**	**33.3**	**33.3**	**28.9**
Δ	+50.0	+20.0	+16.6	+28.9	+10.0	+10.0	+20.0	+13.4

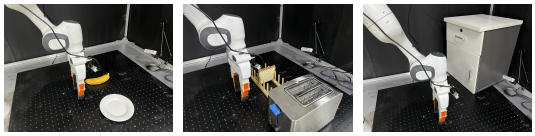
that limited real-environment interaction is insufficient for effective online RL. WMPO improves the average success rate to 50.9%, but shows no gain on LIBERO-Long (0 % $\uparrow$), suggesting that a less reliable world model cannot support long-horizon policy optimization. In contrast, WoVR achieves the best performance across all suites, increasing the average success rate to 69.5%, with especially large gains on Object and Long. Under the stronger *full-trajectory SFT* setting, where the base policy is already strong (88.1%), all methods have less room for improvement: GRPO reaches 89.7% and WMPO reaches 92.4%, while WoVR still achieves the highest average success rate of 95.9% and consistently outperforms WMPO.

These results show that world-model-based RL can outperform limited-budget online RL, especially when the simulator is sufficiently stable and policy-aligned. Additional ablations on *PACE* and *KIR* are provided in Appendix D.2.

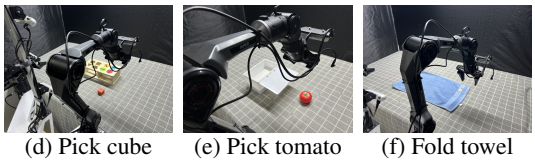
4.3 Do Policies Optimized with WoVR Reliably Transfer to the Real World?

Experimental Setup We conduct real-world transfer experiments on two robotic platforms with different control characteristics: the **Franka Emika Panda** and the **AgileX Piper**. As shown in Fig 5, we evaluate three manipulation tasks on each platform. Detailed task requirements are provided in Appendix F.2.

For each platform, we train a *multi-task* VLA policy and a corresponding world model over the three tasks. After training with WoVR, we directly deploy the optimized policies on the physical robots and evaluate their **SR** in real world, where each task is evaluated over 30 independent trials. More details can be seen in Appendix F.



(a) Pick banana (b) Pick bread (c) Open drawer



(d) Pick cube (e) Pick tomato (f) Fold towel

Figure 5: **Real-world experiments on two robotic platforms.** Top row: Franka Emika Panda; bottom row: AgileX Piper.

Quantitative Results. Table 3 reports real-world success rates on two robotic platforms. WMPO improves over the base policy, especially on Franka (+13.4 points), indicating that world-model-based policy optimization can provide useful learning signals. However, its gains are limited on the noisier AgileX Piper platform (+3.4 points), suggesting that naïve imagined RL remains vulnerable to world-model hallucination. In contrast, WoVR consistently achieves the best performance, improving the base policy by +28.9 points on Franka and +13.4 points on AgileX Piper. These results show that WoVR enables more stable and effective policy optimization in imagination, with strong transfer to both high-precision and noisier real-world robotic systems, without additional online interaction.

Generalization to Different VLA Backbones. To further evaluate the generality of WoVR, we conduct additional experiments using $\pi_{0.5}$ as the VLA backbone on the AgileX Piper platform. We follow π_{RL} [7] and optimize the policy with Flow-SDE under the unchanged WoVR framework.

As shown in Table 4, WoVR consistently improves the $\pi_{0.5}$ policy across all evaluated tasks, raising the average success rate by +22.3 points. The post-trained policy achieves high success rates on pick-and-place tasks. Although Fold Towel is a more challenging deformable-object task, WoVR still brings a +10.0 point gain. These results demonstrate that the benefit of WoVR is not tied to a specific VLA architecture.

Table 4: **Generalization to $\pi_{0.5}$ on AgileX Piper.** Success rates (%), 30 trials/task.

Method	Fold Towel	Pick Cube	Pick Tomato	Avg
$\pi_{0.5}$-base	20.0	60.0	23.3	34.4
w/ **Ours**	**30.0**	**86.7**	**53.3**	**56.7**
Δ	+10.0	+26.7	+30.0	+22.3

5 Conclusion

In this work, we first identify hallucination in closed-loop imagined interaction as a central reliability bottleneck in world-model-based RL for VLA policy optimization. To address this, we propose WoVR, a hallucination-aware framework that improves simulator stability with an action-controllable video world model, reduces effective prediction depth through KIR, and maintains policy–simulator alignment via PACE. Experiments on LIBERO and real-world manipulation tasks show that WoVR enables effective policy optimization. Together, these results suggest that hallucination-controlled world models can serve as practical RL simulators, enabling a closed-loop path where online deployment data continually refine the simulator and further improve VLA policies in imagination.

6 Limitations

WoVR identifies hallucination in closed-loop imagined rollouts as a key reliability bottleneck for world-model-based RL, but our analysis remains primarily empirical. In particular, we do not provide a formal characterization of how hallucination propagates into policy optimization, nor a regret bound for the policy learned in the imagined world model relative to the optimal policy in the real environment. In addition, while WoVR demonstrates promising results on real-world tabletop manipulation tasks, its capability in substantially longer-horizon tasks and mobile manipulation remains underexplored. These settings may amplify error accumulation through extended interaction horizons, changing viewpoints, larger state spaces, and more diverse contact dynamics. Scaling WoVR to such scenarios may require large-scale world-model pretraining on more diverse embodied data, as well as more sophisticated mechanisms for suppressing compounding errors.

Acknowledgments

This work was supported by the New Generation Artificial Intelligence-National Science and Technology Major Project under Grant 2025ZD0122902; the National Natural Science Foundation of China (NSFC) under Grants 62406159 and 62293545; Zhongguancun Academy under Grant C20250301; the Beijing High Innovation Plan with Grant No. 202604841445; the Beijing Natural Science Foundation–Xiaomi Innovation Joint Fund under Grant L253007; the Beijing Major Science and Technology Project under Contract No. Z251100008125023; the Beijing Academy of Artificial Intelligence (BAAI); and the Shenzhen Science and Technology Program under Grant AI2026016.

References

[1] M. J. Kim, C. Finn, and P. Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, 2025. doi:10.15607/RSS.2025.XXI.017.

[2] K. Black, N. Brown, D. Driess, A. Esmail, M. R. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, 2025. doi:10.15607/RSS.2025.XXI.010.

[3] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. R. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, et al. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 2025.

[4] H. Li, Y. Chen, W. Cui, W. Liu, K. Liu, M. Zhou, Z. Zhang, and D. Zhao. Survey of vision-language-action models for embodied manipulation. *Acta Automatica Sinica*, 52(1):18–51, 2026. doi:10.16383/j.aas.c250394.

[5] H. Zang, M. Wei, S. Xu, Y. Wu, Z. Guo, Y. Wang, H. Lin, P. Wang, H. Yuan, Y. Zhang, L. Shi, Y. Xie, Z. Xu, Z. Liu, K. Chen, W. Tang, Q. Zhang, W. Zhang, C. Yu, and Y. Wang. RLux-VLA: A Unified and Efficient Framework for Reinforcement Learning of Vision-Language-Action Models. In *Proceedings of Robotics: Science and Systems*, 2026. doi:10.15607/RSS.2026.XXII.089.

[6] J. Liu, F. Gao, B. Wei, X. Chen, Q. Liao, Y. Wu, C. Yu, and Y. Wang. What Can RL Bring to VLA Generalization? An Empirical Study. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 97121–97151. Curran Associates, Inc., 2025. doi:10.52202/085713-3248.

[7] K. Chen, Z. Liu, T. Zhang, Z. Guo, S. Xu, H. Lin, H. Zang, X. Li, Q. Zhang, Z. Yu, G. Fan, T. Huang, Y. Wang, and C. Yu. π_{RL}: Online RL Fine-tuning for Flow-based Vision-Language-Action Models. *arXiv preprint arXiv:2510.25889*, 2025.

[8] Y. Chen, S. Tian, S. Liu, Y. Zhou, H. Li, and D. Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy. In *Proceedings of Robotics: Science and Systems, RSS 2025, Los Angeles, CA, USA, Jun 21-25, 2025*, 2025. doi:10.15607/RSS.2025.XXI.019.

[9] G. Lu, W. Guo, C. Zhang, Y. Zhou, H. Jiang, Z. Gao, Y. Tang, and Z. Wang. VLA-RL: Towards Masterful and General Robotic Manipulation with Scalable Reinforcement Learning. In *2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026.

[10] Y. Li, X. Ma, J. Xu, Y. Cui, Z. Cui, Z. Han, L. Huang, T. Kong, Y. Liu, H. Niu, W. Peng, J. Qiao, Z. Ren, H. Shi, Z. Su, J. Tian, Y. Xiao, S. Zhang, L. Zheng, H. Li, and Y. Wu. Gr-rl: Going dexterous and precise for long-horizon robotic manipulation. *arXiv preprint arXiv:2512.01801*, 2025.

[11] Y. Liu, H. Li, S. Tian, Y. Qin, Y. Chen, Y. Zheng, Y. Huang, and D. Zhao. Towards long-lived robots: Continual learning VLA models via reinforcement fine-tuning. In *Proceedings of Robotics: Science and Systems, RSS 2026, Sydney, Australia, Jul, 2026*, 2026. doi:10.15607/RSS.2026.XXII.086.

[12] K. Lei, H. Li, D. Yu, Z. Wei, L. Guo, Z. Jiang, Z. Wang, S. Liang, and H. Xu. Performant robotic manipulation with real-world reinforcement learning. *Science Robotics*, 11(116): eaed6267, 2026. doi:10.1126/scirobotics.aed6267.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[14] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[15] D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645 (8081):633–638, 2025. doi:10.1038/s41586-025-09422-z.

[16] W. Cui, C. Zhao, S. Wei, J. Zhang, H. Geng, Y. Chen, H. Li, and H. Wang. GAPartManip: A Large-Scale Part-Centric Dataset for Material-Agnostic Articulated Object Manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14791–14798, 2025. doi:10.1109/icra55743.2025.11128643.

[17] J. Luo, C. Xu, J. Wu, and S. Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025. doi:10.1126/scirobotics.ads5033.

[18] H. Li, Y. Zuo, J. Yu, Y. Zhang, Z. Yang, K. Zhang, X. Zhu, Y. Zhang, T. Chen, G. Cui, et al. SimpleVLA-RL: Scaling VLA Training via Reinforcement Learning. In *International Conference on Learning Representations*, pages 125846–125869, 2026.

[19] W. Liu, Y. Wan, J. Wang, Y. Kuang, W. Cui, X. Shi, H. Li, D. Zhao, Z. Zhang, and H. Wang. FetchBot: Learning generalizable object fetching in cluttered scenes via zero-shot Sim2Real. In J. Lim, S. Song, and H.-W. Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 859–884. PMLR, 2025.

[20] Z. Zheng, X. Peng, T. Yang, C. Shen, S. Li, H. Liu, Y. Zhou, T. Li, and Y. You. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024.

[21] T. Wan, A. Wang, B. Ai, B. Wen, C. Mao, C.-W. Xie, D. Chen, F. Yu, H. Zhao, J. Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[22] F. Zhu, Z. Yan, Z. Hong, Q. Shou, X. Ma, and S. Guo. WMPO: World Model-based Policy Optimization for Vision-Language-Action Models. In *International Conference on Learning Representations*, pages 62486–62502, 2026.

[23] H. Li, P. Ding, R. Suo, Y. Wang, Z. Ge, D. Zang, K. Yu, M. Sun, H. Zhang, D. Wang, and W. Su. Vla-rft: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators. *arXiv preprint arXiv:2510.00406*, 2025.

[24] X. Yuan, T. Mu, S. Tao, Y. Fang, Z. Zhang, and H. Su. Policy Decorator: Model-Agnostic Online Refinement for Large Policy Model. In *International Conference on Learning Representations*, pages 28129–28164, 2025.

[25] M. Pan, S. Feng, Q. Zhang, X. Li, J. Song, C. Qu, Y. Wang, C. Li, Z. Xiong, Z. Chen, Y. Liu, and J. Luo. Sop: A scalable online post-training system for vision-language-action models. *arXiv preprint arXiv:2601.03044*, 2026.

[26] H. Zang, S. Yu, H. Lin, T. Zhou, Z. Huang, Z. Guo, X. Xu, Y. Sheng, J. Zhou, S. Zhang, F. Gao, W. Tang, Y. Yue, Q. Zhang, X. Chen, C. Yu, and Y. Wang. USER: A Unified and Extensible System for Online Real-World Policy Learning in Embodied AI. In *Proceedings of Robotics: Science and Systems*, 2026. doi:10.15607/RSS.2026.XXII.037.

[27] J. Quevedo, A. K. Sharma, Y. Sun, V. Suryavanshi, P. Liang, and S. Yang. WorldGym: World Model as An Environment for Policy Evaluation. In *International Conference on Learning Representations*, pages 78932–78957, 2026.

[28] Y. Jiang, S. Chen, S. Huang, L. Chen, P. Zhou, Y. Liao, X. He, C. Liu, H. Li, M. Yao, and G. Ren. EnerVerse-AC: Envisioning Embodied Environments with Action Condition. In *NeurIPS 2025 Workshop on Embodied World Models for Decision Making*, 2025.

[29] Z. Jiang, K. Liu, Y. Qin, S. Tian, Y. Zheng, M. Zhou, Z. Zhang, C. Yu, H. Li, and D. Zhao. World4RL: Diffusion World Models for Policy Refinement With Reinforcement Learning for Robotic Manipulation. *IEEE Robotics and Automation Letters*, 11(10):12240–12247, 2026. doi:10.1109/lra.2026.3728345.

[30] S. Fei, S. Wang, L. Ji, A. Li, S. Zhang, L. Liu, J. Hou, J. Gong, X. Zhao, and X. Qiu. SRPO: Self-Referential Policy Optimization for Vision-Language-Action Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6718–6728, 2026.

[31] C.-Y. Hung, N. Majumder, H. Deng, L. Renhang, Y. Ang, A. Zadeh, C. Li, D. Herremans, Z. Wang, and S. Poria. NORA-1.5: A Vision-Language-Action Model Trained using World Model- and Action-based Preference Rewards. *arXiv preprint arXiv:2511.14659*, 2025.

[32] X. Sun, Z. Xu, C. Cao, Z. Liu, Y. Sun, J. Pang, R. Zhang, Z. Yang, K. Pang, D. He, M. Yuan, and J. Chen. AtomVLA: Scalable Post-Training for Robotic Manipulation via Predictive Latent World Models. In *2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026.

[33] Y. Chen, H. Li, Z. Jiang, H. Wen, and D. Zhao. TeViR: Text-to-video reward with diffusion models for efficient reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 56(2):893–905, Feb. 2026. doi:10.1109/TSMC.2025.3638818.

[34] X. Liu, Y. Li, Y. Chen, Y. Qin, H. Li, and D. Zhao. Dual-granularity contrastive reward via generated episodic guidance for efficient embodied RL. *arXiv preprint arXiv:2602.12636*, 2026. doi:10.48550/arXiv.2602.12636.

[35] Y. Guo, L. Shi, J. Chen, and C. Finn. Ctrl-World: A Controllable Generative World Model for Robot Manipulation. In *International Conference on Learning Representations*, pages 6121–6138, 2026.

[36] J. Yang, K. Lin, W. Zhang, J. Li, T. Lin, L. Wu, Y.-Q. Zhang, H. Zhao, P. Luo, Z. Su, H. Li, X. Yue, and L. Chen. Self-Improving Robot Policy with Compositional World Model. In *Proceedings of Robotics: Science and Systems*, 2026. doi:10.15607/RSS.2026.XXII.012.

[37] Y. Guo, T. Lee, L. X. Shi, J. Chen, P. Liang, and C. Finn. Vlaw: Iterative co-improvement of vision-language-action policy and world model. *arXiv preprint arXiv:2602.12063*, 2026.

[38] J. Zhang, Z. Huang, C. Gu, Z. Ma, and L. Zhang. Reinforcing Action Policies by Prophesying. *arXiv preprint arXiv:2511.20633*, 2025.

[39] J. Xiao, Y. Yang, X. Chang, R. Chen, F. Xiong, M. Xu, W.-S. Zheng, and Q. Zhang. World-env: Leveraging world model as a virtual environment for vla post-training. *arXiv preprint arXiv:2509.24948*, 2025.

[40] J. Shin, Z. Li, R. Zhang, J.-Y. Zhu, J. Park, E. Shechtman, and X. Huang. MotionStream: Real-Time Video Generation with Interactive Motion Controls. In *International Conference on Learning Representations*, pages 7381–7406, 2026.

[41] S. Yang, W. Huang, R. Chu, Y. Xiao, Y. Zhao, X. Wang, M. Li, E. Xie, Y. Chen, Y. Lu, S. Han, and Y. Chen. LongLive: Real-time Interactive Long Video Generation. In *International Conference on Learning Representations*, pages 90037–90059, 2026.

[42] J. Tang, J. Liu, J. Li, L. Wu, H. Yang, P. Zhao, S. Gong, X. Yuan, S. Shao, and Q. Lu. Hunyuan-gamecraft-2: Instruction-following interactive game world model. *arXiv preprint arXiv:2511.23429*, 2025.

[43] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[44] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[45] T. Unterthiner, S. Van Steenkiste, K. Kurach, R. Marinier, M. Michalski, and S. Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

[46] D. Danier, F. Zhang, and D. Bull. FloLPIPS: A Bespoke Video Quality Metric for Frame Interpolation. In *2022 Picture Coding Symposium (PCS)*, pages 283–287, 2022. doi:10.1109/pcs56426.2022.10018062.

[47] J. Pennington, P. Joshi, and A. Bhide. Develop custom physical ai foundation models with nvidia cosmos predict-2, June 11 2025. NVIDIA Developer Blog.

[48] Z. Zheng, X. Peng, Y. Lou, C. Shen, T. Young, X. Guo, B. Wang, H. Xu, H. Liu, M. Jiang, et al. Open-Sora 2.0: Training a Commercial-Level Video Generation Model in $200k. *arXiv preprint arXiv:2503.09642*, 2025.

[49] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 44776–44791. Curran Associates, Inc., 2023. doi:10.52202/075280-1939.

[50] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, D. Podell, T. Dockhorn, Z. English, and R. Rombach. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 12606–12633. PMLR, 2024.

[51] B. Chen, D. Martí Monsó, Y. Du, M. Simchowitz, R. Tedrake, and V. Sitzmann. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion. In *Advances in Neural Information Processing Systems*, volume 37, pages 24081–24125. Curran Associates, Inc., 2024. doi:10.52202/079017-0759.

[52] S. Bai, Y. Cai, R. Chen, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[53] J. Zheng, J. Li, Z. Wang, D. Liu, X. Kang, Y. Feng, Y. Zheng, J. Zou, Y. Chen, J. Zeng, T. Wang, Y.-Q. Zhang, J. Liu, and X. Zhan. X-VLA: Soft-Prompted Transformer as Scalable Cross-Embodiment Vision-Language-Action Model. In *International Conference on Learning Representations*, pages 60580–60606, 2026.

[54] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the Continuity of Rotation Representations in Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5745–5753, 2019.

Appendix Contents

A GPU Allocation Strategy 15

B Implemental Details 15

B.1 Implemental Details of world model 15

B.2 Implemental Details of Reward Model 16

B.2.1 Sparse Reward Model 16

B.2.2 Dense Reward Modeling 16

B.3 Implemental Details of KIR 17

C Evaluation Metrics 17

D Ablation Study 18

D.1 Ablation on World Model Mechanisms 18

D.2 Ablation on PACE and KIR 19

D.3 Ablation on Reward Modeling 21

E Qualitative Failure Mode Analysis 22

F Real-World Experiments 23

F.1 Hardware Setup 23

F.2 Detailed Task Requirements 23

F.3 Practical Considerations for Real-World Deployment 23

F.4 Task Setting 25

A GPU Allocation Strategy

The reinforcement-learning pipeline can be decomposed into three components: *Generation*, *Simulator*, and *Training*. In WoVR, the *Simulator* is instantiated by the learned *world model*, which generates the next observation given the current observation and action. In the rollout phase, *Generation* performs policy inference to produce an (optionally chunked) action from the current observation, while the *Simulator* executes the action and returns the next observation; this closed-loop interaction repeats until a batch of trajectories is collected. In the optimization phase, *Training* updates the VLA policy using the collected trajectories, after which the system alternates back to rollout for the next iteration.

Following the system abstraction in RLinf-VLA, WoVR adopts a collocated (shared) GPU allocation strategy, where the three RL pipeline components co-exist on the same set of GPUs, with the *Simulator* implemented as the world-model rollout module. Unlike physical simulators that require dedicated device-side state, WoVR's simulator is a neural network; thus, offload/onload can be naturally realized by swapping only the *model parameters* between GPU and host memory, without migrating any external simulator state. In its original form, collocated execution relied on frequent GPU↔CPU offload/onload to keep only one component resident on GPUs at a time; however, in embodied settings the simulator and generator must interact iteratively, making per-interaction offload/onload prohibitively expensive. Therefore, we use the modified collocated strategy: offload/onload for Generation and Simulator happens only at the beginning and end of the rollout phase, avoiding repeated transfers during closed-loop imagined interaction, as illustrated in Fig. 6.

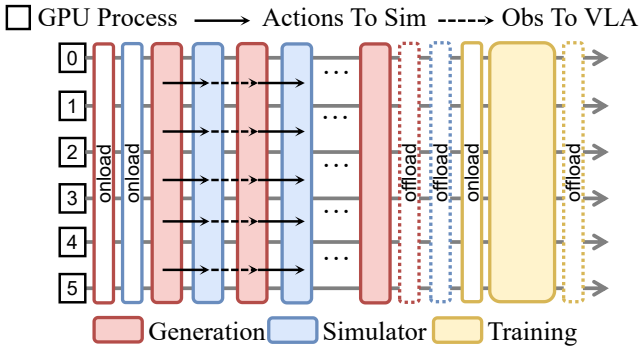


Figure 6: Collocated GPU Allocation Strategy

B Implemental Details

B.1 Implemental Details of world model

The world model generates imagined trajectories chunk by chunk in the latent space of the video diffusion backbone. At rollout step t, the anchored visual context is formed by concatenating the episode's initial frame and the latest generated memory frames, i.e., $[o_0,\ o_{t-c:t}]$. After VAE encoding, the corresponding latent context $[z_0,\ z_{t'-c':t'}]$ is concatenated with Gaussian noise latents for the next chunk. The action-conditioned DiT blocks then denoise these latents under the dual-channel action conditioning shown in Fig. 3, and the decoded future frames are appended to the rollout context. Repeating this procedure yields long-horizon closed-loop imagined trajectories under policy actions.

We train the model with the Rectified Flow objective [50]. Let $x_1 = z_{t'+1:t'+H'}$ denote the target future latents and $x_0 \sim \mathcal{N}(0, \mathbf{I})$ denote Gaussian noise. Given an interpolation time $\tau \in [0, 1]$, the linearly interpolated state is constructed as:

$$x_\tau = (1 - \tau)x_0 + \tau x_1. \tag{2}$$

The corresponding target velocity field along this interpolation trajectory is:

$$v_\tau = \frac{dx_\tau}{d\tau} = x_1 - x_0. \tag{3}$$

the model $u(\cdot;\phi)$ predicts the velocity v_τ from the intermediate latent x_τ:

$$\mathcal{L} = \mathbb{E}_{x_0,x_1,c,\tau}\left[\left\|u(x_\tau, c, \tau;\phi) - v_\tau\right\|^2\right], \tag{4}$$

where c includes the anchored visual context and the action sequence.

To improve robustness under closed-loop inference, we inject diffusion noise into the non-reference context latents during training.

$$z_{t'-c':t'} = \tau_{ctx} * z_{t'-c':t'} + (1 - \tau_{ctx}) * \epsilon, \tag{5}$$

where τ_{ctx} is a value close to 1 and ϵ denotes noise. The initial reference frame remains clean, while recent memory latents are mildly corrupted before being used as context. This augmentation reduces the train–inference gap caused by conditioning on self-generated frames and mitigates brittle context copying during long-horizon autoregressive rollout [51, 35].

B.2 Implemental Details of Reward Model

B.2.1 Sparse Reward Model

For sparse supervision, we use a learned reward classifier that predicts task success from the predicted next observation. Concretely, given the generated observation $\tilde{o}_{t+1}$ from the world model, the sparse reward model R^{sp}_ψ estimates the probability of task success, and the binary reward is defined as

$$r^{\text{sp}}_{t+1} = \mathbb{I}\left(R^{\text{sp}}_\psi(\tilde{o}_{t+1}) \geq 0.5\right), \tag{6}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. Following HiL-SERL [17], the sparse reward classifier is implemented as a lightweight network and trained with binary cross-entropy loss on labeled success states, defined as

$$\mathcal{L}_{binary}(\psi) = -\frac{1}{N}\sum_{i=1}^{N}\Big[r_i \log C_\psi(x_i) + (1 - r_i)\log\big(1 - C_\psi(x_i)\big)\Big]. \tag{7}$$

B.2.2 Dense Reward Modeling

To provide finer-grained learning signals beyond binary success, we further introduce a dense reward model built on top of Qwen3-VL [52]. Instead of directly regressing a scalar reward, we formulate dense reward prediction as an ordinal visual progress estimation problem. Given four consecutive observations $\tilde{o}_{t-2:t+1} = [\tilde{o}_{t-2}, \tilde{o}_{t-1}, \tilde{o}_t, \tilde{o}_{t+1}]$ and the task description l, the dense reward model R^{de}_ψ predicts a discrete progress level

$$\ell_t \in \{0, 1, \ldots, 10\}, \tag{8}$$

where 0 denotes no progress and 10 denotes task completion. The dense reward is then obtained by normalization:

$$r^{\text{de}}_t = \frac{\ell_t}{10}. \tag{9}$$

The model takes as input four consecutive RGB frames together with the task text, and outputs logits over 11 reward levels. Architecturally, we adopt a Qwen3-VL backbone with LoRA adaptation and attach a lightweight MLP reward head for classification. During training, labels are constructed automatically from trajectory rewards: failed trajectories are assigned level 0, successful states are assigned level 10, and pre-success states are linearly mapped to intermediate levels 1–9 according to their temporal proximity to the first successful state. To mitigate class imbalance, we further rebalance samples across reward levels during dataset construction.

The dense reward model is trained with a standard cross-entropy objective:

$$\mathcal{L}_{\text{dense}}(\psi) = \mathbb{E}_{(\tilde{o}_{t-2:t+1}, d, \ell_t)}\left[-\log p^{\text{de}}_\psi(\ell_t \mid \tilde{o}_{t-2:t+1}, d)\right]. \tag{10}$$

B.3 Implemental Details of KIR

Keyframe-Initialized Rollouts (KIR) are implemented as a change to the initial context fed into the world model, rather than as a change to the model architecture. The Wan encoder used by our world model is a causal 3D VAE whose temporal input length follows the form $1 + 4m$. In our rollout implementation, we use five frames (m=1) in total: one persistent reference frame plus four temporal context frames.

For standard long-horizon rollout from the episode start, the first frame is repeated to construct a valid VAE input:

$$\mathcal{C}^{\text{init}} = [o_0, o_0, o_0, o_0, o_0] = \text{Repeat}(o_0, 5), \tag{11}$$

and the first generated chunk is

$$\tilde{o}_{1:9} = W_\phi\big(\mathcal{C}^{\text{init}}, a_{0:8}\big)\,, \tag{12}$$

where W_ϕ denotes the action-conditioned world model. Equivalently, when comparing against a KIR rollout at a task-specific horizon, the non-KIR context can be written as

$$\tilde{o}_{1:9} = W_\phi(\text{Repeat}(o_0, 5), a_{0:8})\,, \tag{13}$$

which removes the local pre-failure context and relies only on the initial observation. This initialization forces the world model to imagine the entire prefix before reaching task-critical states, which increases the effective depth of error accumulation.

KIR instead initializes imagined rollouts near a task-critical state. Ideally, for a trajectory that is about to fail around time $T_{\text{KIR}} + 1$, we would use the first frame together with the last four frames before the failure transition:

$$\mathcal{C}_T^{\text{KIR}} = [o_0, o_{T_{\text{KIR}}-3}, o_{T_{\text{KIR}}-2}, o_{T_{\text{KIR}}-1}, o_{T_{\text{KIR}}}], \tag{14}$$

and generate the first imagined chunk from this keyframe context:

$$\tilde{o}_{T_{\text{KIR}}+1:T_{\text{KIR}}+9} = W_\phi\big(\mathcal{C}_T^{\text{KIR}}, a_{T_{\text{KIR}}:T_{\text{KIR}}+8}\big)\,. \tag{15}$$

The first frame o_0 remains the global anchor, while $o_{T_{\text{KIR}}-3:T_{\text{KIR}}}$ provides the local pre-failure context.

In practice, we avoid manually cherry-picking keyframes for every trajectory. For each task, we set a task-level keyframe index T_{KIR} based on when the base policy typically approaches a critical failure region. This fixed task-level choice keeps KIR inexpensive and reproducible while still placing the first imagined chunk near the decisive part of the task. After the first KIR chunk is generated, subsequent chunks follow the same autoregressive procedure as the standard world-model rollout, using the first frame anchor and the latest generated memory frames as context.

C Evaluation Metrics

We adopt LPIPS [43], FID [44], FVD [45], FloLPIPS [46], and FPS as evaluation metrics for generated videos. Specifically:

- **LPIPS** (Learned Perceptual Image Patch Similarity) measures frame-level perceptual similarity using deep visual features;

$$d\,(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \Big\| w_l \odot \Big(\hat{F}^l_{h,w}(x) - \hat{F}^l_{h,w}(x_0) \Big) \Big\|_2^2 \tag{16}$$

  where x and x_0 denote the generated and real frames, $\hat{F}^l_{h,w}(\cdot)$ denotes the normalized deep feature at spatial location (h, w) of layer l, and w_l is the learned channel-wise weighting.

- **FID** (Fréchet Inception Distance) evaluates the distributional similarity between generated and real frames based on image-level feature statistics;

$$\text{FID} = \|\mu_r - \mu_g\|_2^2 + \text{Tr}\Big(\Sigma_r + \Sigma_g - 2\,(\Sigma_r \Sigma_g)^{1/2}\Big) \tag{17}$$

  where (μ_r, Σ_r) and (μ_g, Σ_g) denote the mean and covariance of image-level features extracted from real and generated frames, respectively.

- **FVD** (Fréchet Video Distance) extends the FID formulation from image-level features to video-level features. It evaluates the distributional similarity between real and generated videos by computing the mean and covariance statistics over spatiotemporal video representations;

$$\mathrm{FVD} = \left\|\mu_r^v - \mu_g^v\right\|_2^2 + \mathrm{Tr}\Big(\Sigma_r^v + \Sigma_g^v - 2\left(\Sigma_r^v \Sigma_g^v\right)^{1/2}\Big) \tag{18}$$

where (μ_r^v, Σ_r^v) and (μ_g^v, Σ_g^v) denote the mean and covariance of video-level features extracted from real and generated videos, respectively.

- **FloLPIPS** measures perceptual similarity after motion alignment along estimated optical-flow trajectories;

$$\mathrm{FloLPIPS} = \frac{1}{T-1}\sum_{t=1}^{T-1}\sum_{l}\frac{\sum_{h,w} M_t(h,w)\left\|w_l \odot \left(\hat{F}_{h,w}^l(V_t) - \hat{F}_{h,w}^l(\hat{V}_t)\right)\right\|_2^2}{\sum_{h,w} M_t(h,w)} \tag{19}$$

The motion weight is defined as:

$$M_t(h,w) = \left\|\mathrm{Flow}(V_t, V_{t+1})_{h,w} - \mathrm{Flow}(\hat{V}_t, \hat{V}_{t+1})_{h,w}\right\|_2 \tag{20}$$

where $\hat{V}_t$ and V_t denote the generated and real frames at time t. $\hat{F}_{h,w}^l(\cdot)$ denotes the normalized feature extracted by the perceptual network at the l-th layer and spatial location (h, w), and w_l is the learned perceptual weight for that layer. $\mathrm{Flow}(V_t, V_{t+1})$ and $\mathrm{Flow}(\hat{V}_t, \hat{V}_{t+1})$ denote the optical flow between two adjacent frames in the real and generated videos, respectively. $M_t(h, w)$ measures the discrepancy between the motion pattern in the real video and that in the generated video at location (h, w) based on optical flow.

- **FPS** measures the number of frames generated per second, quantifying the generation efficiency of the world model.

D Ablation Study

D.1 Ablation on World Model Mechanisms

We first conduct ablation studies on the core design choices of the proposed world model, aiming to understand how different context modeling mechanisms affect long-horizon video generation stability. Specifically, we investigate the following factors: (i) the number of memory frames used as visual context, (ii) the use of a fixed reference frame, and (iii) the effect of adding noise to context frames during training.

Experimental Variants. We compare the full WoVR model against three ablated variants:

- WoVR w/o ref, which removes the fixed reference frame from the context window;
- WoVR w. mem=1, which uses only a single-frame context;
- WoVR w/o noisy context, which disables noise injection on context frames during training.

All variants are trained and evaluated on the **LIBERO-Spatial** suite only. We train the world model using 1,500 VLA rollout trajectories and evaluate on a held-out set of 24 trajectories.

Quantitative Results. Table 5 reports the quantitative results measured by LPIPS, FID, FVD and FloLPIPS under different rollout horizons. Compared to using a single-frame context, employing a multi-frame context with a fixed reference anchor significantly improves performance across all metrics.

To better understand the failure modes behind these quantitative trends, we provide qualitative comparisons in Fig. 7. As shown in the figure, models without a fixed reference frame or noisy context exhibit noticeable spatial drift and object disappearance over long-horizon rollouts, whereas the full WoVR model remains visually stable and consistent with the ground truth.

Table 5: **Ablation study on world model mechanisms (LIBERO-Spatial).** Rollout denotes the rollout horizon length.

Method	Rollout	Metrics			
		LPIPS↓	FID↓	FVD↓	FloLPIPS↓
WoVR (Ours)	512	0.091	36.687	73.493	0.154
	256	0.069	27.238	63.948	0.110
	128	0.051	20.780	49.017	0.081
WoVR w/o ref	512	0.133	73.942	123.502	0.168
	256	0.089	49.406	86.000	0.116
	128	0.064	35.559	86.146	0.090
WoVR w. mem=1	512	0.120	64.501	86.042	0.165
	256	0.086	46.790	81.742	0.117
	128	0.065	36.047	79.605	0.095
WoVR w/o noisy context	512	0.099	44.712	77.284	0.160
	256	0.074	31.691	61.660	0.115
	128	0.054	23.444	58.836	0.085

Removing the reference frame leads to a clear degradation in performance, especially under longer rollout horizons. This result suggests that anchoring the context with a fixed reference frame effectively suppresses error accumulation in the autoregressive feedback loop, which is critical for maintaining stability in long-horizon video generation.

Furthermore, disabling noise injection on context frames also results in noticeable performance drops. While the degradation is moderate for short rollouts, the gap becomes more pronounced as the rollout length increases. This observation indicates that adding mild noise to context frames improves robustness in long-horizon generation by reducing over-reliance on precise conditioning inputs, thereby alleviating the train–inference gap.

Overall, these results demonstrate that the proposed context modeling strategy—combining a fixed reference frame, a multi-frame memory window, and noisy context augmentation—plays a crucial role in stabilizing long-horizon video generation. Together, these mechanisms enable WoVR to maintain high fidelity and temporal consistency under closed-loop autoregressive inference, providing a more reliable simulator for downstream reinforcement learning.

D.2 Ablation on PACE and KIR

We further ablate two interaction-level reliability mechanisms in WoVR: PACE, which aligns the world model with the updated policy distribution, and KIR, which initializes imagined rollouts near task-critical states to reduce effective prediction depth. This ablation is designed to separate the benefit of collecting policy-aligned data from the benefit of keyframe-initialized imagined interaction.

Experimental Setup. We compare four WoVR variants under the one-trajectory SFT policy-optimization protocol from Sec. 4.2:

- **WoVR w/o PACE (1,500 base)**, which trains the world model only on 1,500 trajectories rolled out by the base policy;
- **WoVR w/o PACE (2,500 base)**, which trains the world model on 2,500 trajectories rolled out by the base policy, increasing the amount of data without changing the policy distribution;
- **WoVR w/o KIR**, which keeps the PACE data protocol but removes keyframe initialization, so imagined rollouts are initialized from the episode start;

Results. Table 6 reports the completed ablations on LIBERO-Spatial and LIBERO-Object. WoVR achieves the best performance on both suites, reaching an average success rate of 82.5% over the

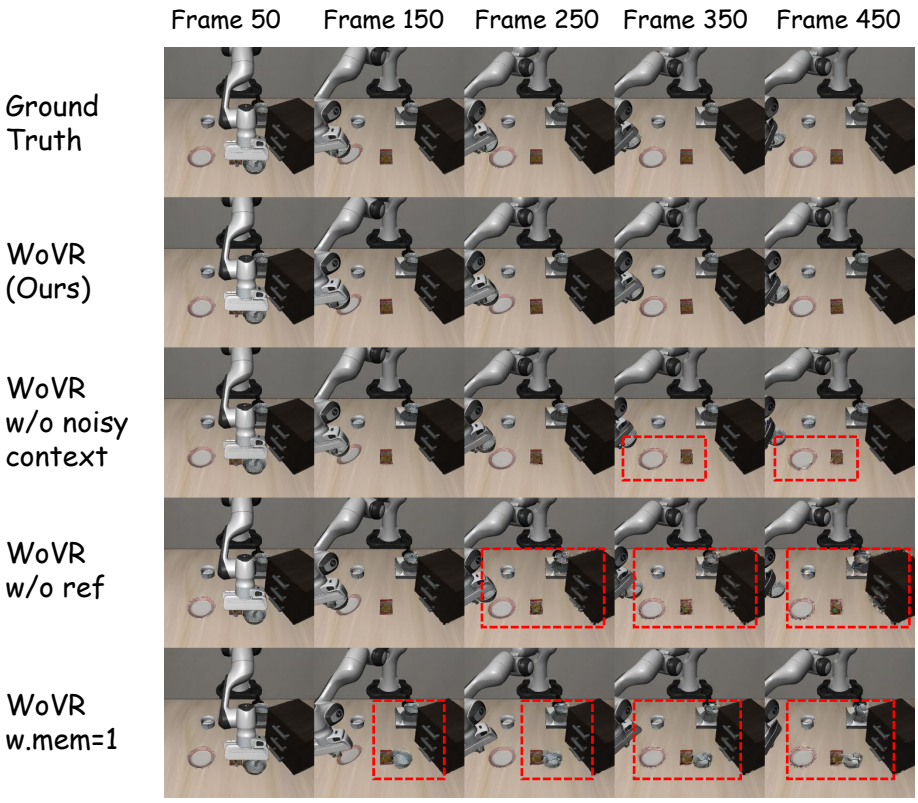


Figure 7: **Qualitative ablation results on LIBERO-Spatial.** Ablated variants exhibit error accumulation and visual drift under long-horizon rollouts, while the full WoVR model remains stable and consistent with the ground truth.

Table 6: **Ablation study on PACE and KIR.** All variants optimize the policy through imagined rollouts. Avg is computed over the two reported suites.

Method	Spatial	Object	Avg ↑
One-Trajectory SFT			
OpenVLA-OFT-base [1]	63.6	36.4	50.0
WoVR w/o PACE (1,500 base)	75.4	76.2	75.8
WoVR w/o PACE (2,500 base)	77.8	77.2	77.5
WoVR w/o KIR	81.6	77.8	79.7
WoVR (1,500 base + 1,000 aligned)	**84.2**	**80.8**	**82.5**
Δ	+20.6	+44.4	+32.5

two reported suites. The two variants without PACE isolate the effect of additional base-policy data: increasing the world-model training set from 1,500 to 2,500 base-policy trajectories improves the two-suite average only from 75.8% to 77.5%. This modest +1.7-point gain suggests that simply adding more data from the same base-policy distribution brings limited benefit once that distribution is sufficiently covered.

Fig. 8 shows how WoVR evolves during one-trajectory SFT training. Before the PACE transition, the imagined training success rate can keep improving while the real-evaluation curve lags behind, revealing a growing mismatch between the evolving policy and the base world model. After updating the simulator from $\mathrm{WM}_{\mathrm{Base}}$ to $\mathrm{WM}_{\mathrm{Evo}}$, the imagined training curve is re-calibrated to the policy-induced distribution and becomes more consistent with real-environment evaluation. This post-PACE alignment indicates that rollout data collected from the evolved policy helps correct accumulated simulator mismatch and mitigates hallucination-induced over-optimism in imagined RL. Together with the quantitative results in Table 6, these curves support the role of PACE in making world-model training signals more faithful to real-world policy performance.

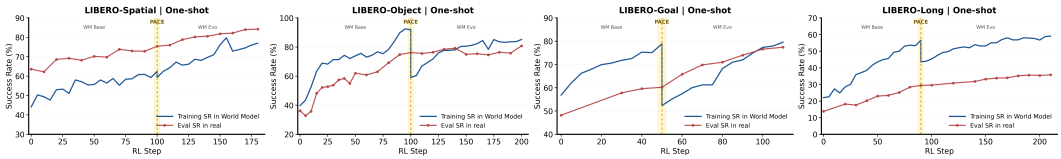


Figure 8: **One-trajectory SFT training and real-evaluation curves under WoVR.** Blue denotes the training success rate in the world model, red denotes the evaluation success rate in the real environment, and the shaded vertical band marks the PACE transition from $\mathrm{WM}_{\mathrm{Base}}$ to $\mathrm{WM}_{\mathrm{Evo}}$.

The KIR ablation further shows the importance of initializing imagined rollouts near task-critical states. Removing KIR reduces performance from 84.2% to 81.6% on Spatial and from 80.8% to 77.8% on Object, lowering the two-suite average from 82.5% to 79.7%. These results indicate that KIR improves imagined interaction even when PACE is retained, by shortening the effective prediction depth and reducing early error accumulation. Together, the PACE and KIR ablations suggest that WoVR benefits from both policy-aligned simulator refinement and keyframe-initialized rollout design.

D.3 Ablation on Reward Modeling

We further analyze the impact of different reward modeling choices on policy optimization, comparing a lightweight ResNet-based sparse reward model with a Qwen3-VL-based dense reward model.

Experimental Setup. We keep all other components fixed and vary only the reward model used during policy optimization. The sparse reward model predicts binary task success, while the dense reward model provides fine-grained progress signals. Both variants are evaluated on the same training setup and metrics as in Sec. 4.2.

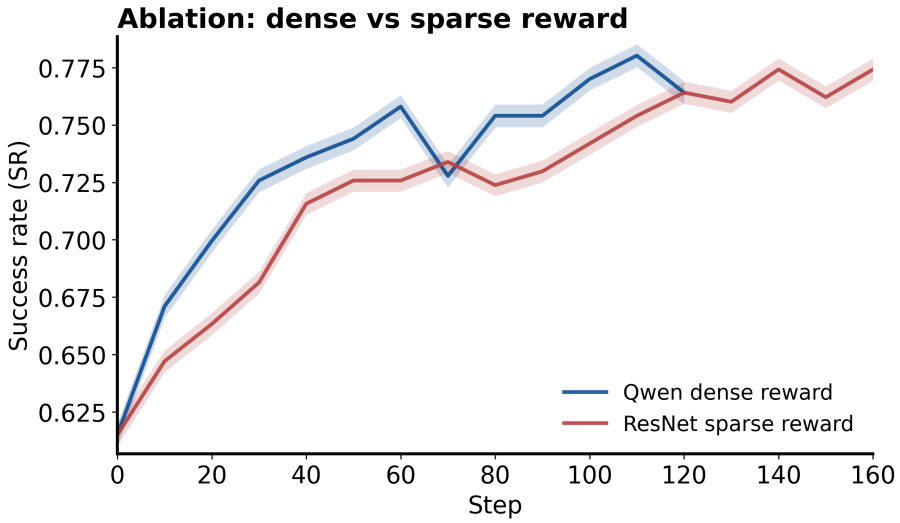


Figure 9: Ablation on reward modeling: dense vs. sparse reward.

Results and Analysis. Fig. 9 shows the learning curves under the two reward settings. We observe that the dense reward model improves sample efficiency in the early stage of training, leading to faster initial performance gains. However, both reward designs converge to similar final performance.

Despite its advantage in early learning, the dense reward model incurs significantly higher computational cost, as it relies on a large vision-language backbone (Qwen3-VL 2B). In practice, we find that rollout with the dense reward model is approximately $3\times$ slower than with the lightweight sparse reward model.

What's more, binary rewards are widely adopted in manipulation benchmarks such as LIBERO as well as real-world robotic setups. Taking the above accounts into considerations, we adopt the sparse reward model in all main experiments.

E Qualitative Failure Mode Analysis

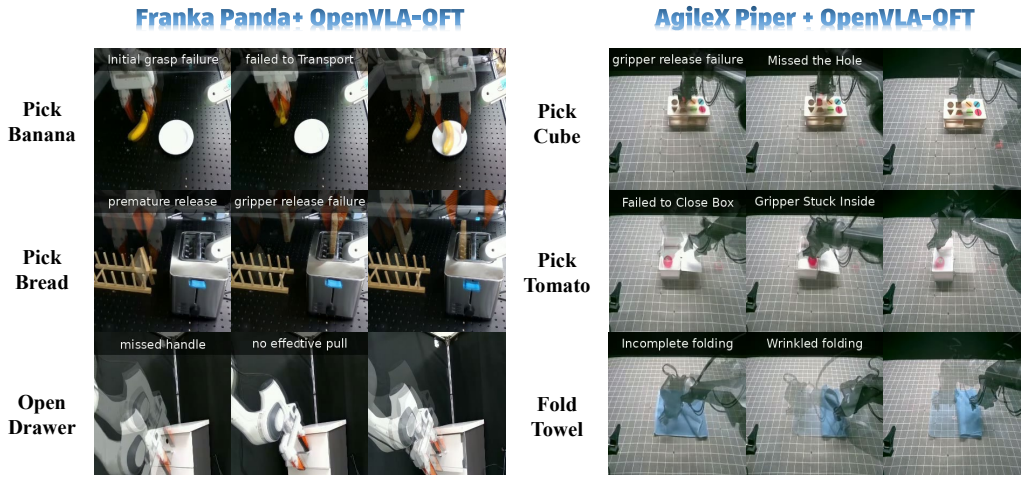


Figure 10: Visualization of real-world execution before and after WoVR policy optimization.

We further analyze representative failure modes of base VLA model and compare them with the behaviors after WoVR. Fig. 10 visualizes execution traces across two robotic platforms and six manipulation tasks. The comparison shows that WoVR not only improves task success rates, but also changes the dominant failure patterns of the base VLA policy.

On the Franka Panda platform, the base policy exhibits different failure modes across the three tasks. In **Pick Banana**, the robot often repeatedly attempts to grasp the banana without establishing a stable grasp, or successfully grasps the banana but fails to move it above the plate. After applying WoVR, the policy more reliably grasps the banana and transfers it into the plate with faster and more direct motions. In **Pick Bread**, the base policy mainly fails by prematurely opening the gripper, causing the bread to drop, or by keeping the gripper closed after reaching the target region. WoVR largely suppresses these two failure modes and produces more consistent release behavior. In **Open Drawer**, the base policy frequently fails to reach the handle accurately or fails to pull the drawer after making contact. With WoVR, the robot more often completes the full interaction sequence, including handle approach, contact, and drawer pulling.

On the AgileX Piper platform, failures are more strongly affected by noisier state estimation and less stable low-level control. In **Pick Cube**, the base policy often moves the cube to the target region but fails to release the gripper, or places the cube with an incorrect pose such that it does not fall into the target hole. WoVR substantially reduces these errors by producing more stable placement and release behaviors. In **Pick Tomato**, the base policy commonly fails to close the box after placing the tomato, or leaves the gripper trapped inside the box. After WoVR optimization, these failure cases are mitigated, although the task remains sensitive to contact and box geometry. In **Fold Towel**, the base policy often fails to complete the fold or keeps holding the towel after folding. WoVR improves the temporal coordination between folding and release, leading to a higher success rate on this deformable-object task.

Overall, the qualitative analysis suggests that WoVR improves real-world execution by reducing recurrent action-level failure modes, including unstable grasping, premature or delayed gripper release, inaccurate target placement, and incomplete contact-rich interactions. These results are consistent with the quantitative real-world improvements reported in Sec. 4.3, and indicate that



(a) AgileX Piper



(b) Franka Emika Panda

Figure 11: Two types of robotic arm platforms

hallucination-aware policy optimization in imagination can translate into more reliable closed-loop behavior on physical robots.

F Real-World Experiments

F.1 Hardware Setup

We evaluate our method on two distinct real-world robotic platforms: the AgileX Piper robotic arm (left) and the Franka Emika Panda robotic arm (right) as shown in Fig.11, each equipped with two-finger grippers. This was done to test the robustness of our method under different control precision hardware conditions.

AgileX Piper AgileX Piper is a 6-degree-of-freedom robotic arm. We conduct remote operation based on the **cobot Magic** platform. We use the D435 camera with a head view to capture RGB images and synchronize and record the data at a frequency of 10Hz.

Franka Emika Panda Franka Emika Panda is a 7-degree-of-freedom robotic arm. We use the fixed third-person perspective D435 to record RGB images and use a spatial mouse for remote operation. Data is recorded at a frequency of 30Hz.

F.2 Detailed task requirements

Here, we provide the detailed requirements for each task on both robotic platforms.

On Franka platform, we evaluate three manipulationtasks:

- Pick Banana: Pick a banana and place it onto a plate;
- Pick Bread: Pick a bread and place it into the toaster;
- Open Drawer: Pull the drawer open;

On AgileX Piper platform,we also evaluate three other tasks:

- Pick Cube: Pick up a cube and place it into a box;
- Pick Tomato: Pick up a tomato, place it into a box, and close the lid;
- Fold Towel: Fold the towel into a compact configuration.

F.3 Practical Considerations for Real-World Deployment

Here, we introduce the selection of the action space and control method for the robotic arm. This is of crucial importance for the actual operation of the less precise robotic arm.

Platforms and Action Space. Experiments are conducted on two robotic platforms: the **AgileX Piper** manipulator and the **Franka Emika Panda**. For both platforms, the action space is defined in the delta end-effector (delta EEF) space, where the VLA policy outputs relative end-effector displacements that are also used as inputs to the world model.

Since the lower-level drives of both robotic arms are controlled in the absolute joint space, each predicted delta action is converted into an absolute end-effector pose, followed by inverse kinematics (IK) to obtain executable joint commands.

Rotation Parameterization. A critical design choice is the parameterization of rotational actions. Directly regressing ΔRPY is often unstable, since Euler angles suffer from discontinuities and singularities, and small prediction errors may lead to large deviations in the reconstructed absolute orientation, thereby affecting IK consistency.

To improve robustness, we adopt the Rotation 6D representation [53, 54]. Given the current end-effector orientation $R_t \in SO(3)$ and the next orientation $R_{t+1} \in SO(3)$, we first compute the local relative rotation:

$$\Delta R_t = R_t^\top R_{t+1}. \tag{21}$$

Following the standard 6D rotation representation, $\mathrm{RotMatTo6D}(\cdot)$ denotes flattening the first two columns of a rotation matrix. The corresponding delta rotation in 6D form is computed as:

$$\Delta \mathbf{r}_{6D,t} = \mathrm{RotMatTo6D}(\Delta R_t) = \left[\Delta R_t[:,1]^\top, \Delta R_t[:,2]^\top\right]^\top \in \mathbb{R}^6. \tag{22}$$

Here, $\Delta \mathbf{r}_{6D,t}$ represents the relative rotation from the current pose to the next pose in the local end-effector frame.

During execution, the predicted 6D vector is converted back to a valid rotation matrix through Gram–Schmidt orthogonalization. Specifically, given two predicted vectors $\mathbf{a}_1, \mathbf{a}_2 \in \mathbb{R}^3$, we compute:

$$\mathbf{b}_1 = \frac{\mathbf{a}_1}{\|\mathbf{a}_1\|}, \tag{23}$$

$$\mathbf{b}_2 = \frac{\mathbf{a}_2 - (\mathbf{b}_1^\top \mathbf{a}_2)\mathbf{b}_1}{\left\|\mathbf{a}_2 - (\mathbf{b}_1^\top \mathbf{a}_2)\mathbf{b}_1\right\|}, \tag{24}$$

$$\mathbf{b}_3 = \mathbf{b}_1 \times \mathbf{b}_2, \qquad \widehat{\Delta R}_t = [\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]. \tag{25}$$

The absolute orientation is then updated as:

$$\widehat{R}_{t+1} = R_t \widehat{\Delta R}_t. \tag{26}$$

Compared with RPY, Rotation 6D avoids angle wrapping and gimbal-lock singularities. Compared with quaternions, it avoids the double-cover ambiguity, where $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation. In addition, the network can directly regress an unconstrained 6D vector, which is later projected to a valid rotation matrix by orthogonalization. This makes the action representation more continuous and better suited for learning stable end-effector rotation commands.

The final action space is defined as:

$$(\Delta x, \Delta y, \Delta z, \Delta \mathbf{r}_{6D}, \text{gripper}).$$

Platform-Specific Considerations. While the overall control pipeline is shared, the two platforms exhibit different levels of state estimation accuracy.

The Franka Panda provides highly accurate and stable end-effector state feedback, making the standard update

$$\mathbf{a}_i = \mathbf{s}_i + \Delta \mathbf{a}_i$$

sufficient in practice. Here, the addition only denotes delta-to-absolute action conversion, not direct physical motion composition.

Table 7: Task and embodiment settings in WoVR real-world experiments. Task-specific data scales are reported in Panel (a), while embodiment-specific training configurations are summarized in Panel (b).

(a) Task-specific data scale.

Embodiment	Task	VLA SFT Demos	WM SFT Rollouts
Franka	Pick banana	25	120
Franka	Pick bread	25	120
Franka	Open drawer	25	120
Agilex	Pick cube	75	180
Agilex	Pick tomato	50	180
Agilex	Fold towel	50	210

(b) Embodiment-specific training configuration.

Embodiment	VLA SFT Steps	WM SFT Steps	Max Episode Steps	RL Epochs
Franka	50k	750k	160	50
Agilex	50k (OpenVLA-OFT) 30k ($\pi_{0.5}$)	1.25M	224	60

In contrast, the AgileX Piper exhibits noticeable noise and bias in the observed end-effector state, which leads to drift when applying the same update rule. To mitigate this issue, we instead compute actions using an accumulated formulation:

$$\mathbf{a}_i = \mathbf{s}_0 + \sum_{k=1}^{i} \Delta \mathbf{a}_k.$$

Here, s_0 represents the given initial position. This reduces sensitivity to noisy feedback and improves long-horizon execution stability.

Although this formulation may introduce discrepancies under external disturbances (e.g., contacts), such errors are compensated by the visual feedback loop of the VLA policy.

Summary. These results highlight that while the proposed delta-space formulation generalize across platforms, handling state estimation noise is critical for reliable real-world deployment. The proposed design enables stable closed-loop execution under both high-precision and low-cost hardware settings.

F.4 Task Setting

The amount of VLA SFT data differs across tasks because the tasks have different levels of difficulty. We allocate more demonstrations to harder tasks to ensure that the supervised policy reaches a sufficient initial performance before reinforcement learning. The amount of data used for world model training follows the same principle: tasks with more complex dynamics, longer horizons, or more diverse interactions require more rollout data for reliable modeling.

In our real-world experiments, we do not use PACE for data collection due to efficiency considerations. Instead, we construct the world-model training set by rolling out multiple VLA SFT checkpoints saved at different training steps. This strategy increases trajectory diversity and broadens action-space coverage, as checkpoints from different training stages induce distinct behavior distributions. For example, on the Franka platform, we collect rollout trajectories from checkpoints saved at 25k, 30k, 35k, 40k, 45k, and 50k SFT steps. We then select the best-performing checkpoint based on real-world evaluation, report its performance, and use it as the initialization for subsequent RL.