ROBOALIGN-R1: Distilled Multimodal Reward Alignment for Robot Video World Models

Hao Wu[1,‡] **Yuqi Li**[2,*] **Yuan Gao**[1,*] **Yizhou Zhao**[3] **Fan Zhang**[4] **Penghao Zhao**[5] **Qiufeng Wang**[3]
Fan Xu[6,†] **Weiyan Wang**[5,†] **Yingli Tian**[2] **Kun Wang**[7,†] **Xian Wu**[5,†] **Xiaomeng Huang**[1]

[1] THU [2] CUNY [3] CMU [4] CUHK [5] Tencent [6] SLAI [7] NTU

[‡]Project leader. [*]Equal contribution. [†]Corresponding authors.

wu-h25@mails.tsinghua.edu.cn

Abstract

Existing robot video world models are typically trained with low-level objectives such as reconstruction and perceptual similarity, which are poorly aligned with the capabilities that matter most for robot decision making, including instruction following, manipulation success, and physical plausibility. They also suffer from error accumulation in long-horizon autoregressive prediction. We present ROBOALIGN-R1, a framework that combines reward-aligned post-training with stabilized long-horizon inference for robot video world models. We construct ROBOTWORLDBENCH, a benchmark of 10,000 annotated video–instruction pairs collected from four robot data sources, and train a multimodal teacher judge, ROBOALIGN-JUDGE, to provide fine-grained six-dimensional evaluation of generated videos. We then distill the teacher into a lightweight student reward model for efficient reinforcement-learning-based post-training. To reduce long-horizon rollout drift, we further introduce *Sliding Window Re-encoding* (SWR), a training-free inference strategy that periodically refreshes the generation context. Under our in-domain evaluation protocol, ROBOALIGN-R1 improves the aggregate six-dimension score by 10.1% over the strongest baseline, including gains of 7.5% on Manipulation Accuracy and 4.6% on Instruction Following; these ranking improvements are further supported by an external VLM-based cross-check and a blinded human study. Meanwhile, SWR improves long-horizon prediction quality with only about 1% additional latency, yielding a 2.8% gain in SSIM and a 9.8% reduction in LPIPS. Together, these results show that reward-aligned post-training and stabilized long-horizon decoding improve task consistency, physical realism, and long-horizon prediction quality in robot video world models. Codes are available.

1 Introduction

Robot video world models are becoming an important component of embodied intelligence because they predict how scenes may evolve under candidate actions before executing them in the real world [1, 2, 3]. As learnable internal simulators, they support planning, policy evaluation, and long-horizon reasoning [4, 5, 6]. Unlike generic video generation, robot world models must capture action-conditioned dynamics, contacts, and scene evolution so that predicted futures remain consistent with manipulation outcomes and are useful for control [7, 8]. This requirement makes it challenging to achieve visual fidelity, task consistency, and physical plausibility simultaneously [9, 10].

However, most existing robot video world models [8, 11, 12, 13] are trained with maximum-likelihood, reconstruction, or perceptual objectives. These losses optimize statistical similarity rather than decision-making utility, leading to a fundamental mismatch: a prediction can be visually accurate

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

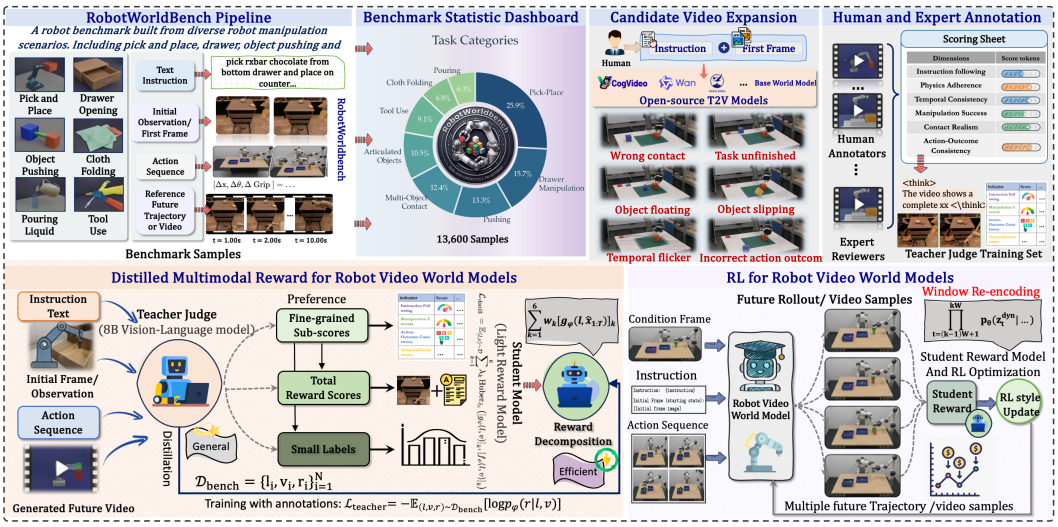


Figure 1: **Overview of RoboAlign-R1.** RobotWorldBench provides robot-centric benchmark statistics and fine-grained annotations for training a multimodal teacher judge. The teacher is distilled into a lightweight student reward model for efficient reinforcement-learning-based post-training of robot video world models. In parallel, a sliding-window re-encoding strategy stabilizes long-horizon autoregressive rollouts by periodically refreshing the visual context during inference.

yet task-incorrect or physically implausible. This gap becomes more pronounced in long-horizon generation, where small errors accumulate and task-level correctness dominates frame-level similarity.

Recent efforts to address this issue via reinforcement learning still rely largely on low-level rewards such as MSE, LPIPS, or SSIM. While efficient, these metrics fail to capture high-level properties critical for robot reasoning, including instruction following, manipulation success, and physically consistent interactions [8, 14]. As a result, current training objectives remain poorly aligned with the capabilities that matter most for downstream control.

These limitations highlight two key challenges. 1) ***reward misalignment***: low-level metrics are scalable but weak, while stronger multimodal evaluators are computationally prohibitive for online optimization. 2) ***long-horizon drift***: autoregressive generation accumulates errors over time, degrading physical realism and task consistency [15]. To address these challenges, we propose RoboAlign-R1, a unified framework for improving both training-time alignment and inference-time stability in robot video world models (Figure 1). Our approach introduces a distilled multimodal reward that aligns model optimization with task-level objectives, and a sliding window re-encoding (SWR) strategy that mitigates long-horizon error accumulation during inference.

Specifically, we construct RobotWorldBench, a large-scale benchmark of $10K$ annotated video–instruction pairs curated from four robot datasets. Using this data, we train a multimodal teacher model (RoboAlign-Judge) that provides fine-grained, multi-dimensional evaluation of generated videos. We then distill this teacher into a lightweight student reward model, enabling efficient reinforcement-learning-based post-training. Complementing this, SWR periodically refreshes the autoregressive context without retraining, improving long-horizon prediction quality with minimal overhead. Our contributions are threefold:

❶ ***Reward-aligned training framework.*** We introduce RoboAlign-R1, which replaces weak low-level online rewards with a distilled multimodal reward aligned with task success. This improves aggregate in-domain evaluation performance by $+10.1\%$ over the strongest baseline while reducing online reward cost by over $10\times$, with gains validated by both VLM-based evaluation and blinded human studies.

❷ ***Benchmark and scalable reward distillation.*** We build RobotWorldBench, a dataset of $10K$ annotated video–instruction pairs from four robot datasets, and a teacher–student distillation pipeline that compresses an $8B$ multimodal judge into a $98M$ reward model running at $\sim$50 videos/sec.

❸ ***Stable long-horizon inference.*** We propose sliding window re-encoding, a training-free decoding strategy that mitigates autoregressive drift by refreshing context during rollout. SWR improves

long-horizon quality with minimal overhead, yielding $+2.8\%$ SSIM, $+0.62$ dB PSNR, and -9.8% LPIPS at about 1% additional latency, with a 12.2% reduction in ROI-LPIPS.

2 Related Work

The relevant technical background is as follows; the full stack context can be found in Appendix A.

Robot Video World Models and RL Post-Training. World models act as internal simulators for planning and long-horizon decision making by predicting future observations under candidate actions [1, 5]. Recent advances in video generation and embodied learning have led to growing interest in robot video world models for modeling manipulation dynamics, interaction outcomes, and future observations [3, 16, 17, 18, 11, 12, 19, 13]. Most existing methods, however, still focus on generation or representation quality and are trained with maximum-likelihood, reconstruction, or perceptual objectives [19, 13, 20, 21, 22]. Recent work has explored RL-based post-training for world models [8, 23, 24], but current rewards remain largely low-level or verifiable, such as reconstruction and perceptual similarity, which are weak proxies for instruction following, physical plausibility, and long-horizon outcome correctness [8, 25, 26, 24]. Similar issues appear in physical spatiotemporal forecasting, where reconstruction-based training often produces oversmoothed predictions and misses rare but decision-relevant events [27, 28, 26]. In addition, autoregressive token-based world models suffer from long-horizon error accumulation at inference time [29, 15, 30, 31]. Our work studies reward-aligned post-training together with a simple strategy for stabilizing long-horizon rollouts.

Multimodal Judges and Rewards. Recent work has explored multimodal evaluators and reward models for video understanding and generation [32, 33, 34]. Compared with low-level visual metrics, these judges provide supervision that better reflects instruction following, physical plausibility, and higher-level video consistency [7, 14, 32, 33]. However, their cost and latency make them difficult to use directly as online rewards in reinforcement learning [7, 33, 14]. ROBOALIGN-R1 follows this direction, but targets robot video world models and distills multimodal judgment into a lightweight reward model for online reinforcement learning.

3 Method

ROBOALIGN-R1 addresses two key challenges in robot video world models: *reward misalignment* during training and *error accumulation* during inference. ROBOALIGN-R1 unifies reward-aligned post-training and stabilized long-horizon decoding within a single framework.

3.1 Token-Based Robot Video World Model

Our backbone (Figure 2) follows the *tokenize–predict–decode* paradigm. A dual-branch visual tokenizer $\mathcal{T}_{\text{vis}} = (E_c, E_d)$ maps a short clip to discrete tokens: E_c encodes a conditioning frame into N_c *context tokens* z^{ctx}, while E_d, cross-attended to E_c's feature pyramid, encodes each observation frame into N_d *dynamics tokens* z_t^{dyn} carrying only residual change. Robot actions $a_t \in \mathbb{R}^{d_a}$ are discretized into action-token blocks $\hat{a}_t$ and interleaved with visual tokens, forming a unified sequence $s = [z^{\text{ctx}}, z_1^{\text{dyn}}, \hat{a}_1, z_2^{\text{dyn}}, \hat{a}_2, \dots]$ (sizes N_c=1280, N_d=80, vocabulary K=4375; see Appendix B). A causal Transformer then models this sequence via standard next-token prediction:

$$\mathcal{L}_{\text{AR}} = -\sum_{t=1}^{T} \log p_\theta\big(z_t^{\text{dyn}} \mid z^{\text{ctx}}, z_{<t}^{\text{dyn}}, \hat{a}_{\le t}\big). \tag{1}$$

At inference time the predicted tokens $\hat{z}_{1:T}^{\text{dyn}}$ are decoded back to pixels via $\hat{x}_{1:T} = \mathcal{D}_{\text{vis}}(z^{\text{ctx}}, \hat{z}_{1:T}^{\text{dyn}})$. The resulting rollout is used for both reward-aligned post-training (§3.2) and sliding-window re-encoding (§3.3). In our implementation, a dual-branch FSQ tokenizer [35, 30] encodes the conditioning frame into context tokens and each subsequent observation into dynamics tokens, while continuous actions are normalized, uniformly binned, and assigned a disjoint vocabulary range. A 12-layer LLaMA decoder models the interleaved sequence under a causal mask with loss applied only to dynamics tokens. During inference, context tokens are encoded once and cached, and sliding-window re-encoding is triggered every W steps. More technical details can be found in Appendix B.

3.2 Reward-Aligned Post-Training

The pre-training objective (Eq. 1) captures surface-level statistical regularity but is agnostic to instruction following, physical plausibility, and action–outcome consistency. We close this gap via RL

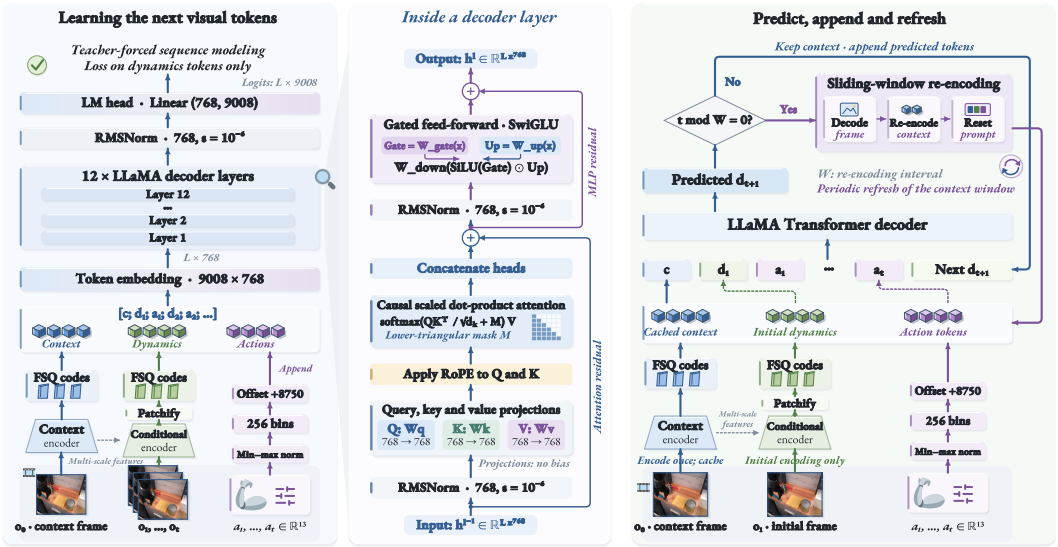


Figure 2: **Token-based robot video world model.** **(a)** Training: a dual-branch FSQ tokenizer produces context tokens c and dynamics tokens d_t; discretized action tokens are interleaved and modeled by a 12-layer LLaMA Transformer with loss on dynamics tokens only. **(b)** Inference: context tokens are encoded once and cached; the model autoregressively predicts $\hat{d}_{t+1}$ and triggers sliding-window re-encoding every W steps.

post-training with a structured multimodal reward, obtained in three stages: benchmark construction, teacher training, and reward distillation.

RobotWorldBench. Let l denote an instruction and v^+ a ground-truth manipulation video from established datasets. RobotWorldBench begins from a candidate pool drawn from four robot datasets and combines two annotation sources: rule-based degradations of ground-truth episodes and generated videos from open-source image-to-video or world-model baselines. From this pool we curate 10,000 annotated video–instruction pairs for judge training and reward distillation. Each retained pair $(l,\, v)$ is annotated with a raw score vector $\mathbf{r} = (r_1, \ldots, r_6)$ along six dimensions *instruction following*, *manipulation success*, *action–outcome consistency*, *temporal consistency*, *contact realism*, and *physics adherence* using the original rubric ranges $[3, 2, 1, 1, 1, 2]$, yielding

$$\mathcal{D}_{\text{bench}} = \left\{ (l_i,\ v_i,\ \mathbf{r}_i) \right\}_{i=1}^{N}. \tag{2}$$

Multimodal teacher judge. We fine-tune Qwen3-VL-8B-Thinking [36] as the teacher judge f_ϕ. Given $(l,\, v)$, the teacher produces structured raw scores $\hat{\mathbf{r}} = f_\phi(l, v)$ in the same six-dimensional rubric space as $\mathbf{r}$ via supervised fine-tuning on $\mathcal{D}_{\text{bench}}$:

$$\mathcal{L}_{\text{teacher}} \;=\; -\,\mathbb{E}_{(l,v,\mathbf{r})\,\sim\,\mathcal{D}_{\text{bench}}}\big[\log\, p_\phi(\mathbf{r} \mid l,\, v)\big]. \tag{3}$$

Student reward distillation. The teacher's autoregressive decoding cost precludes its use as an online RL reward. We distill its judgments into a lightweight student g_ψ—a compact visual–text encoder followed by a sigmoid-activated head emitting $g_\psi(l, v) \in [0, 1]^6$. The distillation set $\mathcal{D}_{\text{distill}}$ mixes annotated benchmark videos with teacher-scored rollouts from baseline/current world models. Teacher raw scores $f_\phi(l, v)$ are normalized dimension-wise via $\tilde{s}_k = [f_\phi(l, v)]_k / r_k^{\max}$ with $r_k^{\max} \in \{3, 2, 1, 1, 1, 2\}$ so both sides of the regression share the same $[0, 1]$ scale:

$$\mathcal{L}_{\text{distill}} \;=\; \mathbb{E}_{(l,v)\,\sim\,\mathcal{D}_{\text{distill}}} \sum_{k=1}^{6} \lambda_k\, \text{Huber}_{\delta_h}\big([g_\psi(l,\, v)]_k,\ \tilde{s}_k(l, v)\big), \tag{4}$$

with Huber threshold $\delta_h = 0.5$ (distinct from the quantization δ_q of Appendix C) and per-dimension balancing weights $\{\lambda_k\}$ (Appendix E); these differ from the reward-aggregation weights $\{w_k\}$ in Eq. 5. To counter reward hacking from distributional shift, we use *online iterative distillation*: every K policy updates, fresh rollouts are re-scored by the teacher to refresh the student.

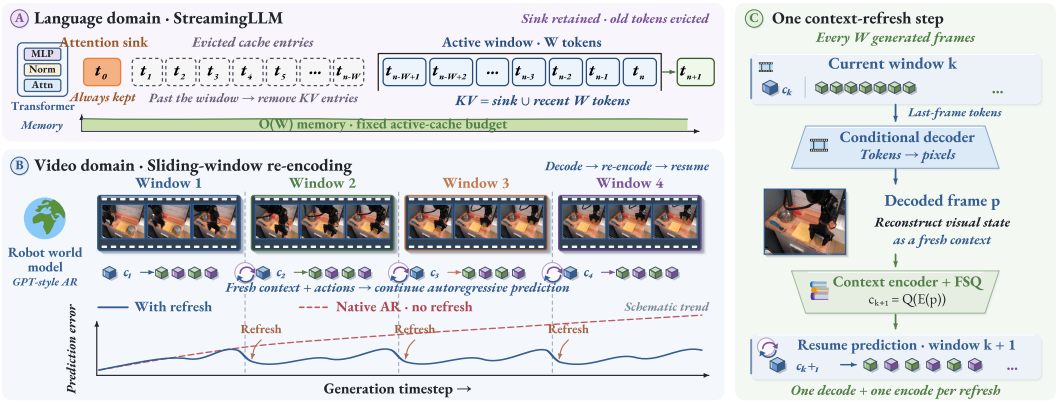


Figure 3: **Sliding window re-encoding.** *Top:* analogy to StreamingLLM [15], which retains an attention sink and a sliding KV-cache window in the language domain. *Bottom:* our approach periodically decodes the last predicted frame to pixel space, re-encodes it as fresh context tokens, and resets the autoregressive prompt, empirically limiting long-horizon drift while keeping the active KV-cache bounded by $O(W)$. *Right:* technical details of a single refresh step.

RL post-training with GRPO. The distilled student provides a composite reward over the six dimensions:

$$R(\hat{x}_{1:T}) \;=\; \sum_{k=1}^{6} w_k \,\big[g_\psi(l,\,\hat{x}_{1:T})\big]_k, \tag{5}$$

where $[g_\psi]_k \in [0,1]$ are the student's normalized scores (same scale as in Eq. 4) and $\{w_k\}$ are importance weights. We post-train with GRPO [37, 8]: given a group of G rollouts $\{\hat{x}^{(j)}\}_{j=1}^{G} \sim p_\theta$, the group-normalized advantage is

$$A^{(j)} \;=\; \frac{R\big(\hat{x}^{(j)}\big) - \mathrm{mean}_j\; R}{\mathrm{std}_j\; R}, \tag{6}$$

and the clipped policy-gradient objective reads

$$\mathcal{L}_{\mathrm{GRPO}} \;=\; -\,\mathbb{E}\!\left[\frac{1}{G}\sum_{j=1}^{G} \min\!\Big(\rho^{(j)} A^{(j)},\; \mathrm{clip}\big(\rho^{(j)},\,1{-}\epsilon,\,1{+}\epsilon\big)\,A^{(j)}\Big)\right] + \beta\, D_{\mathrm{KL}}\big(p_\theta \,\big\|\, p_{\theta_0}\big), \tag{7}$$

where $\rho^{(j)} = p_\theta(\hat{x}^{(j)}) / p_{\theta_{\mathrm{old}}}(\hat{x}^{(j)})$ with θ_{old} synchronized to θ at each GRPO iteration, and p_{θ_0} is the frozen pre-trained reference used for KL regularization. The student scores each rollout in one forward pass, reducing reward cost by over $10\times$ relative to the teacher while preserving high-level judgment fidelity.

3.3 Sliding Window Re-encoding

Under standard autoregressive decoding, each dynamics-token block $\hat{z}_t^{\mathrm{dyn}}$ is conditioned on all previously *predicted* tokens, so per-step errors compound and progressively degrade long-horizon rollouts. Inspired by the attention-sink mechanism of StreamingLLM [15] in the language domain, we introduce a training-free decoding strategy that periodically decodes recent predictions to pixel space and re-encodes them as fresh context, which can limit the carry-over of token-level drift across segments (Figure 3).

Segmented generation. A rollout of T frames is partitioned into $K = \lceil T/W \rceil$ segments of window size W (Figure 3, right). Within segment k, the model generates W dynamics-token blocks from context z_k^{ctx}:

$$\hat{z}^{\mathrm{dyn}}_{(k-1)W+1:\,kW} \;\sim \prod_{t=(k-1)W+1}^{kW} p_\theta\big(z_t^{\mathrm{dyn}} \mid z_k^{\mathrm{ctx}},\; \hat{z}^{\mathrm{dyn}}_{<t},\; \hat{a}_{\le t}\big), \tag{8}$$

where z_1^{ctx} is the original context encoding of the first conditioning frame.

Context refresh. At each segment boundary, the last predicted frame is decoded to pixel space and re-encoded as fresh context for the next segment (Figure 3, right):

$$\hat{x}_{kW} = \left[\mathcal{D}_{\text{vis}}\big(z_k^{\text{ctx}},\ \hat{z}^{\text{dyn}}_{(k-1)W+1:\,kW}\big)\right]_{t=kW}, \qquad \big(z_{k+1}^{\text{ctx}},\ z_0^{\text{dyn}}\big) = \mathcal{T}_{\text{vis}}\big(\hat{x}_{kW}\big), \tag{9}$$

where $\hat{x}_{kW}$ serves as both the new conditioning frame for E_c and a length-one dynamic clip for E_d; z_0^{dyn} seeds the next segment's prompt (Appendix B.2). This *decode–re-encode* cycle makes subsequent predictions depend on the current decoded frame rather than the entire raw token history.

Stylized stability analysis. We provide a simplified local error analysis that explains the qualitative window-size trade-off, rather than predicting exact gains. Let ε upper-bound the per-step token-space prediction error, δ_q the decode–re-encode quantization error, and $\alpha \in [0, 1)$ a local contraction factor on within-segment context-error amplification (precise assumptions in Appendix C).

Proposition 1. *Under the stylized model above, sliding-window re-encoding with window size W yields the bound*

$$\mathcal{E}_{\text{SWR}}(T) \ \le\ W\varepsilon + \frac{W\varepsilon + \delta_q}{1 - \alpha^W}, \qquad \forall\, T, \tag{10}$$

which does not grow explicitly with T; the $\alpha \to 0$ limit simplifies to $2W\varepsilon + \delta_q$. By contrast, vanilla AR without refresh admits a worst-case bound of order $\varepsilon/(1-\alpha)$, which blows up as $\alpha \to 1$ and degrades to $\mathcal{O}(T\varepsilon)$ in the non-contractive regime.

Proof sketch. Within segment k the context is fixed, so errors accumulate for $\le W$ steps, giving $\le W\varepsilon + \alpha^W \eta_k$ where η_k is the context error. The boundary cycle (Eq. 9) refreshes context at a one-time cost δ_q, so $\eta_{k+1} \le W\varepsilon + \alpha^W \eta_k + \delta_q$ converges to $\eta^* = (W\varepsilon + \delta_q)/(1 - \alpha^W)$; see Appendix C. □

Remark 1 (Role of α and choice of W). *Smaller W tightens the within-segment term $W\varepsilon$ but pays more δ_q per refresh; larger W leaves more room for within-segment drift. An asymptotic analysis of Eq. 10 (Corollary 1 in Appendix C) predicts that the optimal window grows logarithmically in δ_q/ε and in $1/(-\ln \alpha)$, i.e. more local drift or cheaper refresh both push W^* larger. Our ablation (Table 3) shows $W = 6$ hits the best balance, consistent with this prediction for our operating regime.*

4 Experiments

Datasets. We evaluate RoboAlign-R1 on RT-1 [38] and BridgeData V2 [39] in the main rollout experiments. RobotWorldBench draws candidate tasks and videos from four robot datasets, including RT-1, BridgeData V2, CALVIN [40], and LIBERO [41], and uses 10,000 annotated video–instruction pairs for judge training and reward distillation.

Baselines. We compare RoboAlign-R1 with three groups of approaches: closed-source video generation models, open-source video generation models, and embodied world models. Additional details are deferred to the Table 26.

Evaluation Metrics. We report both semantic/physical alignment metrics and pixel-level reconstruction metrics. The former are scored by RoboAlign-Judge over six dimensions and should be interpreted as an in-domain automated proxy rather than a substitute for large-scale human evaluation. We additionally report an external VLM-based cross-check and a small-scale blinded human study on held-out subsets to test whether the main ranking is preserved beyond the in-domain judge (Appendix J.7 and Appendix J.8). The pixel-level metrics include standard global measures and motion-mask-based ROI metrics; details are provided in Appendix I.

Implementation Details. RoboAlign-R1 is implemented in PyTorch, and all training and evaluation are conducted on NVIDIA A100 GPUs. In the RL stage, we adopt GRPO and distill the fine-tuned RoboAlign-Judge into a lightweight reward model.

4.1 RQ1: Does RoboAlign-R1 Achieve State-of-the-Art World-Model Quality?

Quantitative results. Tables 1–2 show that RoboAlign-R1 achieves the best overall performance under our evaluation protocol. It reaches an aggregate RoboAlign-Judge score of $\mathbf{8.52}_{\pm 0.15}$, exceeding the strongest baseline iVideoGPT ($7.74_{\pm 0.62}$) by $+10.1\%$ with 75.8% lower standard

deviation, while improving all six judged dimensions and surpassing all closed-source commercial models (e.g., Kling 2.6: 6.84) despite far fewer parameters. On low-level metrics, RoboAlign-R1 is also strongest on both datasets, reducing LPIPS by 4.9% on RT-1 and MSE by 8.7% on BridgeData V2 relative to the respective runners-up. The main ranking is preserved by the external VLM-based validation (Appendix J.7) and the blinded human study (Appendix J.8).

Qualitative analysis. Figures 4–5 show representative comparisons. RoboAlign-R1 produces more coherent manipulation sequences with accurate grasping and stable contact geometry. In contrast, iVideoGPT often produces blurry details, while Wan2.2-TI2V-5B (LoRA) struggles with temporal coherence and occasionally hallucinates object deformation. Across RT-1 and BridgeData V2, RoboAlign-R1 better preserves texture, shadows, and background stability, consistent with the quantitative gains above.

Table 1: Robot world-model performance on RobotWorldBench (mean ± std.).

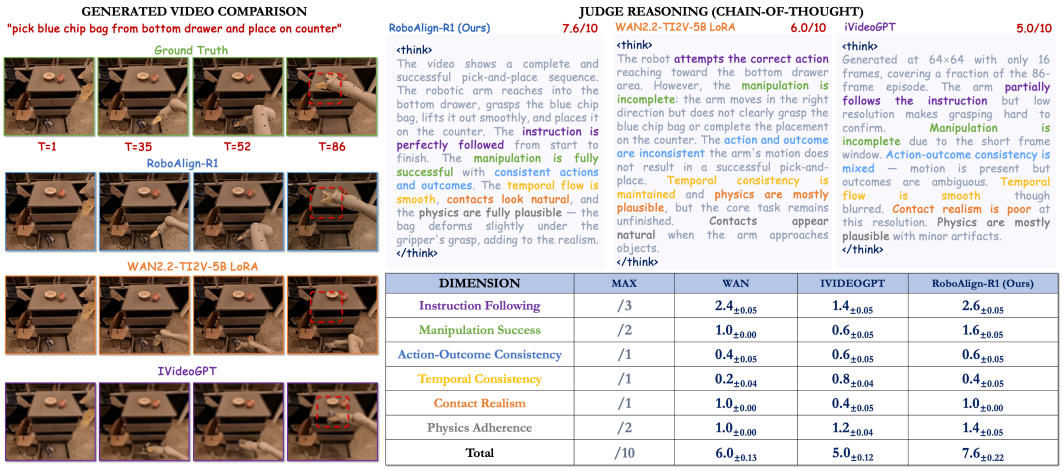
Method	Task Alignment			Physical Realism			Total
	Instr. ↑	Manip. ↑	Act.-Out. ↑	Temp. ↑	Contact ↑	Phys. ↑	↑
Real videos	3.0	2.0	1.0	1.0	1.0	2.0	10.0
Closed video models							
Kling 2.6	**2.42**$\pm$0.09 [3]	**1.38**$\pm$0.10 [3]	0.46$\pm$0.08	**0.58**$\pm$0.07 [3]	0.82$\pm$0.06	1.18$\pm$0.11	**6.84**$\pm$0.22 [3]
Runway Gen-4.5	2.34$\pm$0.10	1.29$\pm$0.09	0.43$\pm$0.07	0.55$\pm$0.08	0.79$\pm$0.06	1.12$\pm$0.10	6.52$\pm$0.24
MiniMax Hailuo 02	2.18$\pm$0.11	1.17$\pm$0.10	0.38$\pm$0.07	0.47$\pm$0.09	0.71$\pm$0.08	1.01$\pm$0.12	5.92$\pm$0.28
Luma Dream Machine	2.03$\pm$0.12	1.08$\pm$0.11	0.35$\pm$0.08	0.44$\pm$0.09	0.69$\pm$0.08	0.92$\pm$0.12	5.51$\pm$0.30
Open video models							
HunyuanVideo-I2V	1.20$\pm$0.28	0.40$\pm$0.22	0.24$\pm$0.08	0.56$\pm$0.15	**0.96**$\pm$0.08 [3]	**1.56**$\pm$0.15 [2]	4.92$\pm$0.63
LTX-Video	2.28$\pm$0.07	1.32$\pm$0.07	**0.52**$\pm$0.07 [3]	0.26$\pm$0.05	0.28$\pm$0.10	1.00$\pm$0.18	5.66$\pm$0.36
Stable Video Diffusion XT	1.84$\pm$0.15	0.98$\pm$0.13	0.46$\pm$0.08	0.06$\pm$0.08	0.36$\pm$0.20	0.58$\pm$0.20	4.28$\pm$0.74
Mochi-1	1.83$\pm$0.12	0.91$\pm$0.10	0.31$\pm$0.07	0.39$\pm$0.08	0.61$\pm$0.07	0.82$\pm$0.11	4.87$\pm$0.29
I2VGen-XL	1.56$\pm$0.15	0.66$\pm$0.14	0.32$\pm$0.04	0.00$\pm$0.00	0.44$\pm$0.19	0.56$\pm$0.16	3.54$\pm$0.57
CogVideoX-I2V	1.94$\pm$0.21	1.00$\pm$0.14	0.40$\pm$0.18	0.42$\pm$0.16	0.92$\pm$0.07	1.24$\pm$0.19	5.92$\pm$0.75
OpenSora-I2V	1.58$\pm$0.13	0.82$\pm$0.11	0.27$\pm$0.06	0.31$\pm$0.07	0.53$\pm$0.08	0.71$\pm$0.10	4.22$\pm$0.30
OpenSora-Plan-I2V	1.79$\pm$0.12	0.89$\pm$0.10	0.30$\pm$0.07	0.37$\pm$0.08	0.60$\pm$0.08	0.81$\pm$0.11	4.76$\pm$0.29
Embodied / interactive world-model baselines							
RLVR-World	2.29$\pm$0.10	1.31$\pm$0.09	0.44$\pm$0.07	0.43$\pm$0.08	0.84$\pm$0.06	1.19$\pm$0.10	6.54$\pm$0.23
iVideoGPT	**2.60**$\pm$0.11 [2]	**1.60**$\pm$0.11 [2]	**0.70**$\pm$0.11 [2]	**0.74**$\pm$0.14 [2]	0.56$\pm$0.10	**1.54**$\pm$0.15 [3]	**7.74**$\pm$0.62 [2]
RoboDreamer	2.02$\pm$0.16	1.02$\pm$0.16	0.34$\pm$0.05	0.32$\pm$0.12	0.04$\pm$0.05	0.70$\pm$0.17	4.44$\pm$0.56
Vid2World	2.18$\pm$0.10	1.08$\pm$0.04	0.40$\pm$0.09	0.22$\pm$0.07	**0.98**$\pm$0.04 [2]	1.04$\pm$0.14	5.90$\pm$0.30
Additional training baselines							
Wan2.2-TI2V-5B (LoRA)	2.40$\pm$0.09	1.02$\pm$0.11	0.41$\pm$0.08	0.24$\pm$0.10	**0.98**$\pm$0.05 [2]	0.96$\pm$0.12	6.01$\pm$0.24
RoboAlign-R1 (ours)	**2.72**$\pm$0.06 [1]	**1.72**$\pm$0.07 [1]	**0.72**$\pm$0.05 [1]	**0.78**$\pm$0.05 [1]	**1.00**$\pm$0.04 [1]	**1.58**$\pm$0.07 [1]	**8.52**$\pm$0.15 [1]

Table 2: Low-level metrics on full variable-length rollouts over RT-1 and BridgeData V2. We report mean values and, when available, standard deviations across repeated evaluations over the full generated rollout.

Method	RT-1				BridgeData V2			
	MSE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	MSE ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Real videos	0.000	—	1.000	0.000	0.000	—	1.000	0.000
Embodied / interactive world-model baselines								
RLVR-World	**0.0125**$\pm$0.0017 [2]	**19.47**$\pm$1.30 [2]	**0.736**$\pm$0.035 [2]	**0.182**$\pm$0.014 [2]	**0.0139**$\pm$0.0021 [2]	**19.06**$\pm$1.38 [2]	**0.721**$\pm$0.040 [2]	**0.178**$\pm$0.014 [2]
iVideoGPT	**0.0128**$\pm$0.0018 [3]	**19.36**$\pm$1.32 [3]	**0.732**$\pm$0.036 [3]	**0.188**$\pm$0.015 [3]	0.0438$\pm$0.0025	15.98$\pm$1.52	0.558$\pm$0.042	0.435$\pm$0.021
RoboDreamer	0.0171$\pm$0.0024	18.12$\pm$1.56	0.671$\pm$0.046	0.231$\pm$0.019	0.0184$\pm$0.0027	17.86$\pm$1.69	0.649$\pm$0.051	0.224$\pm$0.018
Vid2World	0.0138$\pm$0.0019	19.03$\pm$1.37	0.721$\pm$0.038	0.196$\pm$0.016	0.0145$\pm$0.0022	18.92$\pm$1.43	0.711$\pm$0.041	0.188$\pm$0.015
Additional training baselines								
Wan2.2-TI2V-5B (LoRA)	0.0131$\pm$0.0018	19.31$\pm$1.29	0.729$\pm$0.037	0.184$\pm$0.015	**0.0141**$\pm$0.0021 [3]	**19.02**$\pm$1.39 [3]	**0.718**$\pm$0.040 [3]	**0.182**$\pm$0.014 [3]
RoboAlign-R1 (ours)	**0.0121**$\pm$0.0016 [1]	**19.72**$\pm$1.26 [1]	**0.745**$\pm$0.032 [1]	**0.173**$\pm$0.013 [1]	**0.0127**$\pm$0.0019 [1]	**20.00**$\pm$1.31 [1]	**0.731**$\pm$0.037 [1]	**0.175**$\pm$0.013 [1]

4.2 RQ2: Do Distilled Multimodal Rewards Outperform Low-Level Alternatives?

We compare the distilled multimodal student reward with low-level alternatives under matched RL post-training settings. Figure 6(a) shows that the student reward performs best on all six semantic and physical dimensions on RobotWorldBench, reaching the highest aggregate score (8.52, +33.8%



DIMENSION	MAX	WAN	IVIDEOGPT	RoboAlign-R1 (Ours)
Instruction Following	/3	$2.4_{\pm 0.05}$	$1.4_{\pm 0.05}$	$2.6_{\pm 0.05}$
Manipulation Success	/2	$1.0_{\pm 0.00}$	$0.6_{\pm 0.05}$	$1.6_{\pm 0.05}$
Action-Outcome Consistency	/1	$0.4_{\pm 0.05}$	$0.6_{\pm 0.05}$	$0.6_{\pm 0.05}$
Temporal Consistency	/1	$0.2_{\pm 0.04}$	$0.8_{\pm 0.04}$	$0.4_{\pm 0.05}$
Contact Realism	/1	$1.0_{\pm 0.00}$	$0.4_{\pm 0.05}$	$1.0_{\pm 0.00}$
Physics Adherence	/2	$1.0_{\pm 0.00}$	$1.2_{\pm 0.04}$	$1.4_{\pm 0.05}$
Total	/10	$6.0_{\pm 0.13}$	$5.0_{\pm 0.12}$	$7.6_{\pm 0.22}$

Figure 4: **Qualitative comparison on a representative manipulation case.** ROBOALIGN-R1 generates physically coherent sequences with accurate grasping.

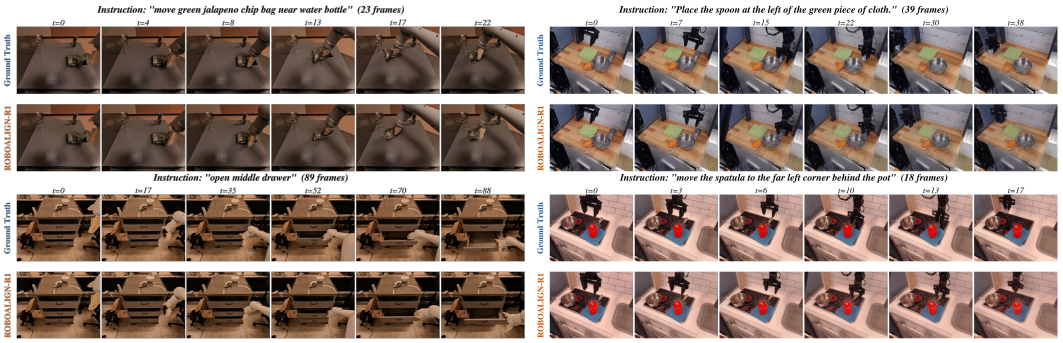


Figure 5: **Qualitative case study of ROBOALIGN-R1 on RT-1 and BridgeData V2**, showing improved texture, shadow consistency, and background stability.

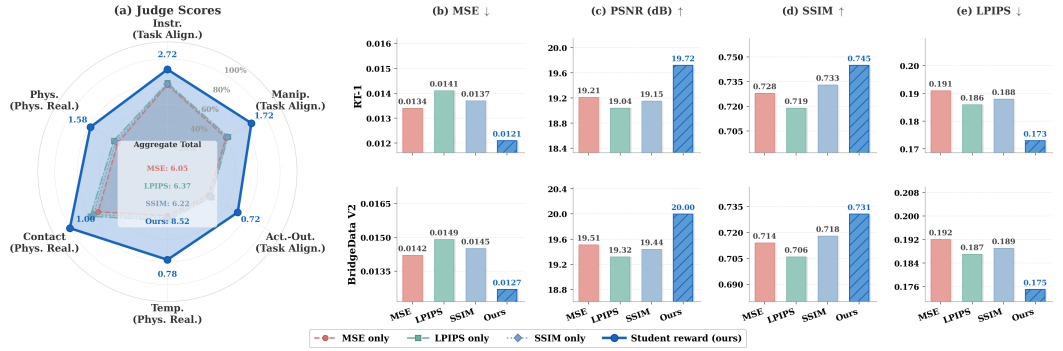


Figure 6: **Reward-type ablation for RL post-training on ROBOTWORLDBENCH**. Panel (a) shows that the distilled student reward yields the strongest judge-aligned improvement across semantic and physical dimensions. Panels (b)–(e) show that it also delivers the best full-rollout low-level metrics on RT-1 and BridgeData V2.

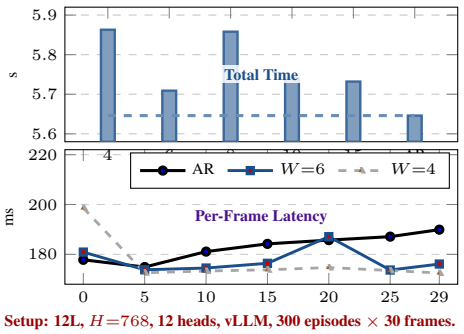
over the best single-metric reward baseline, LPIPS). Figure 6(b)–(e) further shows that it also yields the best full-rollout low-level metrics on both RT-1 and BridgeData V2. Overall, the distilled reward is better aligned with our in-domain evaluator and remains beneficial to pixel fidelity.

4.3 RQ3: Does SWR Improve Long-Horizon Quality Without Heavy Overhead?

We compare SWR with default autoregressive (AR) decoding on RT-1. Table 3(a) shows that SWR with $W{=}6$ improves long-horizon generation quality, yielding $+2.8\%$ SSIM, $+0.62$ dB PSNR, and -9.8% LPIPS. The gains are larger in the region of interest, where ROI-LPIPS drops by -12.2%. These results are consistent with SWR reducing accumulated drift by periodically refreshing the

Table 3: **SWR empirical summary** on RT-1. **(a)** Default autoregressive decoding vs. SWR with $W{=}6$ (mean $\pm$ std.; Δ vs. Default AR). **(b)** Total wall-clock time vs. W and per-frame latency on a keyframe subsample.

Metric	Default AR	SWR ($W{=}6$)	Δ
Quality			
SSIM ↑	$0.7526_{\pm 0.084}$	$0.7735_{\pm 0.072}$	$+2.8\%$
PSNR (dB) ↑	$20.49_{\pm 3.22}$	$21.11_{\pm 3.23}$	$+0.62$ dB
LPIPS ↓	$0.2078_{\pm 0.071}$	$0.1875_{\pm 0.064}$	-9.8%
ROI-PSNR ↑	$16.86_{\pm 3.49}$	$17.48_{\pm 3.53}$	$+0.62$ dB
ROI-LPIPS ↓	$0.1027_{\pm 0.042}$	$0.0902_{\pm 0.039}$	-12.2%
Efficiency			
Infer. (s)	$5.646_{\pm 0.01}$	$5.709_{\pm 0.07}$	$+1.1\%$
Throughput (FPS)	5.31	5.26	-0.9%
Peak mem. (GB)	33.4	32.0	-4.2%
Max seq. len.	4,070	1,838	-54.8%
KV-cache growth	$O(T)$	$O(W)$	bounded
Re-enc. overhead	—	73.8 ms	1.3%



Setup: 12L, $H{=}768$, 12 heads, vLLM, 300 episodes $\times$ 30 frames.

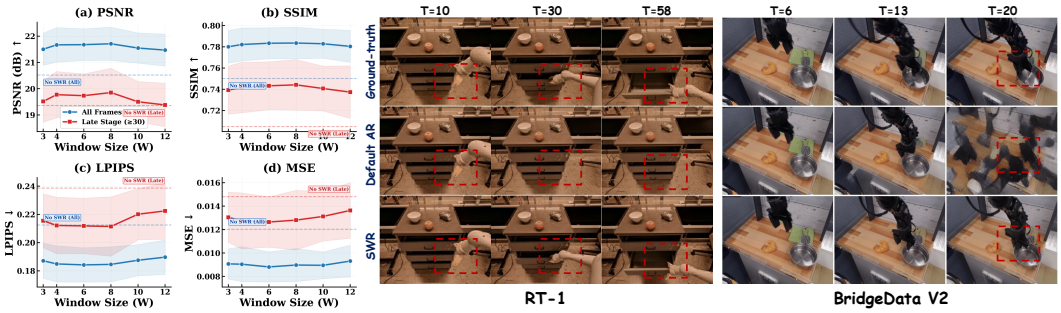


Figure 7: **Qualitative effect of SWR.** SWR mitigates error accumulation and maintains visual fidelity in long-horizon generation by periodically re-encoding recent context.

context (Figure 7), and $W{=}6$ provides the best quality–efficiency trade-off among the tested window sizes. These gains come with only modest overhead. Table 3(b) shows that SWR keeps total inference time (5.709 s vs. 5.646 s) and throughput (5.26 FPS vs. 5.31 FPS) close to AR, with about 1.1% extra wall-clock time. SWR also bounds KV-cache growth to $O(W)$, reducing maximum sequence length by 54.8% and peak memory by 4.2%. As a result, SWR maintains a near-flat per-frame latency profile, whereas AR shows a $+6.8\%$ latency drift by frame 29.

4.4 RQ4: Does Online Iterative Distillation Help Maintain Reward Alignment?

Table 4 shows the effect of online iterative distillation under the same student architecture and evaluation protocol. Online iterative distillation improves the aggregate score from 8.09 to 8.52 and *Manipulation Success* from 1.58 to 1.72, while slightly improving RT-1 LPIPS from 0.176 to 0.173. This supports the motivation in Section 3.2: refreshing the student during RL helps maintain reward alignment under distribution shift.

Table 4: **Iterative distillation.**

Setting	Total ↑	Manip. ↑	LPIPS ↓
One-shot distill	$8.09_{\pm 0.23}$	$1.58_{\pm 0.09}$	$0.176_{\pm 0.014}$
Online iterative	$\mathbf{8.52}_{\pm 0.15}$ [1]	$\mathbf{1.72}_{\pm 0.07}$ [1]	$\mathbf{0.173}_{\pm 0.013}$ [1]

5 Conclusion & Limitation

We present RoboAlign-R1, a framework that improves robot video world models through reward-aligned post-training and stabilized long-horizon decoding. By constructing a robot-centric benchmark to train a multimodal teacher judge and distilling it into a lightweight student reward model, RoboAlign-R1 enables efficient RL post-training that lifts the aggregate in-domain evaluation score by $+10.1\%$ over the strongest baseline under our current protocol, with the overall ranking further supported by external VLM-based validation and a small-scale blinded human study; the proposed sliding window re-encoding further yields $+2.8\%$ SSIM and -9.8% LPIPS gains at about 1% additional latency. Our current evaluation is still limited to single-arm tabletop and countertop manipulation, and downstream policy improvement remains to be validated. Extending to diverse embodiments, ensembling multiple judges, and measuring control-level gains are promising directions for future work. Code, datasets, models, and video samples are available at the project website.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (42125503, 42430602).

References

[1] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

[2] Jingcheng Ni, Yuxin Guo, Yichen Liu, Rui Chen, Lewei Lu, and Zehuan Wu. Maskgwm: A generalizable driving world model with video mask reconstruction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22381–22391, 2025.

[3] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Leo Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. *OpenAI Blog*, 1(8):1, 2024.

[4] Wentao Tan, Lei Zhu, Bowen Wang, Enci Xie, Baixu Ji, Zengrong Lin, Wenjie Yang, Jingjing Li, and Heng Tao Shen. Towards generalist embodied ai: A survey on world models for vla agents. *Authorea Preprints*, 2026.

[5] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

[6] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[7] Dacheng Li, Yunhao Fang, Yukang Chen, Shuo Yang, Shiyi Cao, Justin Wong, Michael Luo, Xiaolong Wang, Hongxu Yin, Joseph E Gonzalez, et al. Worldmodelbench: Judging video generation models as world models. *arXiv preprint arXiv:2502.20694*, 2025.

[8] Jialong Wu, Shaofeng Yin, Ningya Feng, and Mingsheng Long. Rlvr-world: Training world models with reinforcement learning. *arXiv preprint arXiv:2505.13934*, 2025.

[9] Leonardo Barcellona, Andrii Zadaianchuk, Davide Allegro, Samuele Papa, Stefano Ghidoni, and Efstratios Gavves. Dream to manipulate: Compositional world models empowering robot imitation learning with imagination. *arXiv preprint arXiv:2412.14957*, 2024.

[10] Jing-Cheng Pang, Nan Tang, Kaiyuan Li, Yuting Tang, Xin-Qiang Cai, Zhen-Yu Zhang, Gang Niu, Masashi Sugiyama, and Yang Yu. Learning view-invariant world models for visual robotic manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[11] Siqiao Huang, Jialong Wu, Qixing Zhou, Shangchen Miao, and Mingsheng Long. Vid2world: Crafting video diffusion models to interactive world models. *arXiv preprint arXiv:2505.14357*, 2025.

[12] Jiageng Mao, Sicheng He, Hao-Ning Wu, Yang You, Shuyang Sun, Zhicheng Wang, Yanan Bao, Huizhong Chen, Leonidas Guibas, Vitor Guizilini, et al. Robot learning from a physical world model. *arXiv preprint arXiv:2511.07416*, 2025.

[13] Jingtao Ding, Yunke Zhang, Yu Shang, Yuheng Zhang, Zefang Zong, Jie Feng, Yuan Yuan, Hongyuan Su, Nian Li, Nicholas Sukiennik, et al. Understanding world or predicting future? a comprehensive survey of world models. *ACM Computing Surveys*, 58(3):1–38, 2025.

[14] Tianyi Xiong, Shihao Wang, Guilin Liu, Yi Dong, Ming Li, Heng Huang, Jan Kautz, and Zhiding Yu. Phycritic: Multimodal critic models for physical ai. *arXiv preprint arXiv:2602.11124*, 2026.

[15] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[16] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[17] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 1 (2):6, 2023.

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

[19] Peng-Fei Zhang, Ying Cheng, Xiaofan Sun, Shijie Wang, Fengling Li, Lei Zhu, and Heng Tao Shen. A step toward world models: A survey on robotic manipulation. *arXiv preprint arXiv:2511.02097*, 2025.

[20] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[21] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver 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.

[22] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612, 2004.

[23] Mihir Prabhudesai, Russell Mendonca, Zheyang Qin, Katerina Fragkiadaki, and Deepak Pathak. Video diffusion alignment via reward gradients. *arXiv preprint arXiv:2407.08737*, 2024.

[24] Anonymous. Spatiotemporal forecasting as planning: A model-based reinforcement learning approach with generative world models. *Under review as a conference paper at ICLR 2026*, 2026.

[25] Hao Wu, Xingjian Shi, Ziyue Huang, Penghao Zhao, Wei Xiong, Jinbao Xue, Yangyu Tao, Xiaomeng Huang, and Weiyan Wang. Beamvq: aligning space-time forecasting model via self-training on physics-aware metrics. *arXiv preprint arXiv:2405.17051*, 2024.

[26] Weiyan Wang, Xingjian Shi, Ruiqi Shu, Yuan Gao, Rui Ray Chen, Kun Wang, Fan Xu, Jinbao Xue, Shuaipeng Li, Yangyu Tao, et al. Beamvq: Beam search with vector quantization to mitigate data scarcity in physical spatiotemporal forecasting. *arXiv preprint arXiv:2502.18925*, 2025.

[27] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, et al. Skilful precipitation nowcasting using deep generative models of radar. *Nature*, 597(7878):672–677, 2021.

[28] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems*, 29, 2016.

[29] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

[30] Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. ivideogpt: Interactive videogpts are scalable world models. *Advances in Neural Information Processing Systems*, 37:68082–68119, 2024.

[31] Yunbo Wang, Haixu Wu, Jianjin Zhang, Zhifeng Gao, Jianmin Wang, Philip S Yu, and Mingsheng Long. Predrnn: A recurrent neural network for spatiotemporal predictive learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2208–2225, 2022.

[32] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024.

[33] Xuan He, Dongfu Jiang, Ge Zhang, Max Ku, Achint Soni, Sherman Siu, Haonan Chen, Abhranil Chandra, Ziyan Jiang, Aaran Arulraj, et al. Videoscore: Building automatic metrics to simulate fine-grained human feedback for video generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2105–2123, 2024.

[34] Tianyi Xiong, Xiyao Wang, Dong Guo, Qinghao Ye, Haoqi Fan, Quanquan Gu, Heng Huang, and Chunyuan Li. Llava-critic: Learning to evaluate multimodal models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 13618–13628, 2025.

[35] Fabian Mentzer, David Minnen, Eirikur Agustsson, and George Toderici. Finite scalar quantization: VQ-VAE made simple. In *International Conference on Learning Representations*, 2024.

[36] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[37] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[38] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gober, Karol Hausman, Alexander Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2023.

[39] Homer Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. *arXiv preprint arXiv:2308.12952*, 2023.

[40] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. In *IEEE Robotics and Automation Letters*, volume 7, pages 7327–7334, 2022.

[41] Bo Liu, Yifeng Zhu, Chongkai Gao, Yizhou Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

[42] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Appendix Contents

A **Extended Related Work** . . . 14

B **Training Details of the Video World Model** . . . 15

B.1 Notation . . . 15

B.2 Stage 1: Context-Aware Compression Tokenizer . . . 15

B.3 Stage 2: Autoregressive Transformer World Model . . . 17

B.4 Dataset: Bridge V2 Instantiation . . . 19

B.5 Reproduction Protocol . . . 20

B.6 Optimization Infrastructure . . . 21

B.7 End-to-End Quantitative Summary . . . 22

B.8 TensorBoard Training Curves . . . 22

C **Stylized derivation for Proposition 1** . . . 27

D **Pseudocode for Reward-Aligned Post-Training** . . . 30

E **Student Reward Model Details** . . . 31

F **Pseudocode and Implementation of Sliding Window Re-encoding** . . . 34

G **SWR window-size ablation** . . . 36

H **SWR qualitative visualizations** . . . 38

I **Evaluation Metrics** . . . 42

I.1 Semantic and Physical Alignment Metrics . . . 42

I.2 Pixel-Level Reconstruction Metrics . . . 42

I.3 Motion-Mask-Based ROI Metrics . . . 42

J **RoboAlign-Judge: Multimodal Teacher Judge Training Details** . . . 44

J.1 Motivation and Design Rationale . . . 44

J.2 Training Data Construction . . . 44

J.3 Six-Dimension Scoring Rubric . . . 48

J.4 Model Architecture and LoRA Fine-Tuning . . . 48

J.5 Chain-of-Thought Reasoning . . . 49

J.6 From Teacher Judge to Reward Signal . . . 50

J.7 Comparison with Alternative VLM Judges . . . 50

J.8 Small-scale Blinded Human Evaluation . . . 53

J.9 Comparison with Alternative Judge Backbones . . . 54

J.10 Limitations and Future Directions . . . 54

J.11 Judge Prompt Templates . . . 54

A Extended Related Work

Robot Video World Models. World models provide learnable internal simulators for planning, policy evaluation, and long-horizon decision making by predicting how environments evolve under candidate actions [1, 5]. With the rise of video generation and embodied learning, recent work has increasingly explored video-based world models for robotics, including approaches that model manipulation dynamics, interaction outcomes, and future observations through video prediction or generative simulation [3, 16, 17, 18, 11, 12]. These studies highlight the promise of video world models for embodied intelligence, but most still emphasize generation quality, dynamics modeling, or representation learning, and are largely trained with maximum-likelihood or reconstruction-style objectives [19, 13]. In contrast, our work focuses on reward-aligned post-training for robot video world models.

RL Post-Training for World Models. Recent work has begun to explore reinforcement learning as a post-training mechanism for world models, showing that world-model behavior can be further aligned with downstream objectives beyond standard supervised fitting [8, 23, 24]. However, existing approaches still rely mainly on low-level or verifiable rewards, such as pixel reconstruction error or perceptual similarity metrics [8, 20, 25, 21, 22, 26]. While these rewards are stable and easy to compute, they remain weak proxies for the properties that matter in robot prediction, such as correct instruction execution, physically plausible contact dynamics, and consistent long-horizon action outcomes. Similar limitations have also been observed in physical spatiotemporal forecasting [27, 28], where optimizing average reconstruction objectives often leads to oversmoothed predictions and poor coverage of rare but decision-critical events [26, 24]. Long-horizon rollout quality is also shaped by inference-time generation strategy: in autoregressive token-based world models [29, 15, 30, 31], prediction errors can accumulate over time and progressively degrade later predictions. Our work therefore complements reward-aligned post-training with a simple decoding strategy for stabilizing long-horizon rollouts.

Multimodal Judges and Rewards. Another relevant line of research studies multimodal evaluators and reward models for video understanding and generation [32, 33, 34]. Prior work shows that multimodal judges can provide richer supervision than low-level visual metrics by explicitly assessing instruction following, physical plausibility, and higher-level video consistency [7, 14, 32, 33]. Such evaluators are much better aligned with human judgment, but their inference cost and latency make them difficult to use directly as online rewards in reinforcement learning [7, 33, 14]. Our method is closely related in spirit to this line of work, but differs in two important ways: we focus on robot video world models rather than general video evaluation, and we distill high-capacity multimodal judgment into a lightweight reward model that is practical for online RL. This makes multimodal reward supervision not only evaluative, but directly usable for post-training robot world models.

B Training Details of the Video World Model

This appendix details the implementation of the two-stage video world model used as the backbone throughout our experiments. Stage 1 trains a context-aware compression tokenizer that maps raw robot video clips into discrete index sequences; on top of a frozen Stage 1 tokenizer, Stage 2 trains an autoregressive Transformer that jointly models visual and action tokens. The two stages share the same clip length, spatial resolution, and data pipeline, and communicate exclusively through discrete tokens.

B.1 Notation

Unless stated otherwise, the symbols in Table 5 are shared between both stages.

Table 5: Symbols shared between Stage 1 and Stage 2.

Symbol	Meaning	Value
T	Clip length (frames)	8
t_c	Context frames	1
$t_d = T - t_c$	Dynamic frames	7
$H \times W$	Spatial resolution	256×320
p	Dynamic-branch patch size	4
D	FSQ quantization dimension	5
$\mathbf{L}$	FSQ level vector	$(7, 5, 5, 5, 5)$
$K = \prod_i L_i$	Codebook size	4375
N_c	Context tokens per clip	1280
N_d	Dynamic tokens per frame	80
D_a	Action-vector dimensionality (Bridge V2)	13
B_a	Per-dimension action bins	256
V	Joint vocabulary size (Stage 2)	9008
S	Transformer sequence length (Stage 2)	1931
ϕ_l	VGG-16 feature map at layer l (LPIPS)	—

B.2 Stage 1: Context-Aware Compression Tokenizer

Stage 1 compresses a clip $\{x_t\}_{t=1}^{T} \in \mathbb{R}^{T\times 3\times H\times W}$ into 1840 discrete indices using two asymmetric branches: a *Full VAE* over the context frame $x_c := x_1$ and a *Conditional VAE* over the dynamic frames $x_d := \{x_t\}_{t=2}^{T}$. Quantization is carried out by Finite Scalar Quantization (FSQ); no learnable codebook is used.

Architecture. The *context branch* uses a convolutional encoder E_c that produces a latent feature map together with an intermediate feature pyramid $\{f_l^{(E)}\}_{l=1}^{L}$:

$$\big(h,\ \{f_l^{(E)}\}_{l=1}^{L}\big) = E_c(x_c), \qquad h \in \mathbb{R}^{256\times 32\times 40}.$$

A 1×1 convolution projects h into the quantization space, $z_c = \mathrm{QuantConv}(h) \in \mathbb{R}^{5\times 32\times 40}$. No spatial patchification is applied, so this branch emits $N_c = 32 \cdot 40 = 1280$ tokens, each covering an 8×8 pixel block of the input.

The *dynamic branch* uses a conditional encoder E_d whose downsampling stages are interleaved with cross-attention layers that inject the context feature pyramid,

$$s_{l+1} = \mathrm{CrossAttn}\big(\mathrm{DownBlock}_l(s_l),\ f_{l+1}^{(E)}\big),$$

so that E_d encodes only the residual dynamics relative to the static context. Its output $d' \in \mathbb{R}^{64\times 32\times 40}$ is patchified with $p = 4$ and linearly projected,

$$d'' = \mathrm{Patchify}_p(d') \in \mathbb{R}^{80\times 1024}, \qquad z_d = W_q\, d'' + b_q \in \mathbb{R}^{80\times 5}, \quad W_q \in \mathbb{R}^{5\times 1024},$$

producing $N_d = 80$ tokens per dynamic frame, each covering a 32×32 pixel block.

FSQ quantization. For a vector $z \in \mathbb{R}^D$ with $D = 5$ and $\mathbf{L} = (7, 5, 5, 5, 5)$, FSQ performs per-dimension bounding, straight-through rounding, and mixed-radix encoding:

$$\tilde{z}_i = \tfrac{L_i}{2} \tanh(z_i + s_i) - o_i, \qquad \hat{z}_i = \mathrm{round}(\tilde{z}_i) + \big(\tilde{z}_i - \mathrm{sg}[\mathrm{round}(\tilde{z}_i)]\big),$$

$$\mathrm{idx}(\hat{z}) \;=\; \sum_{i=1}^{D} \Big(\hat{z}_i + \lfloor L_i/2 \rfloor\Big) \prod_{j<i} L_j \;\in\; \{0, \ldots, K-1\}, \qquad K = \prod_i L_i = 4375.$$

Because the index set is analytically fixed by $\mathbf{L}$ and every codeword is reachable by construction, no commitment loss is required.

Decoding. Symmetric decoders D_c and D_d reconstruct the clip,

$$\hat{x}_c = D_c\big(\mathrm{PostQuantConv}(\mathcal{Q}(z_c))\big), \qquad \hat{x}_d = D_d\big(\mathrm{UnPatchify}(\mathrm{PostQuantLinear}(\mathcal{Q}(z_d)));\ \{f_l^{(D)}\}\big),$$

with cross-attention inserted at every upsampling stage of D_d to read the decoder-side feature pyramid $\{f_l^{(D)}\}$ of D_c. *No skip connection bypasses the quantization bottleneck*: the decoders depend solely on the discrete indices. Each clip therefore carries $N_c + t_d \cdot N_d = 1280 + 7 \cdot 80 = 1840$ tokens, corresponding to a compression ratio of roughly $1070\times$ relative to the $8{\cdot}3{\cdot}256{\cdot}320 = 1{,}966{,}080$ raw pixel values.

End-to-end data flow. Figure 8 summarizes the Stage 1 forward pipeline, making explicit how the two branches share the context feature pyramid yet quantize independently through FSQ, and how the decoders reconstruct the clip from discrete indices alone.

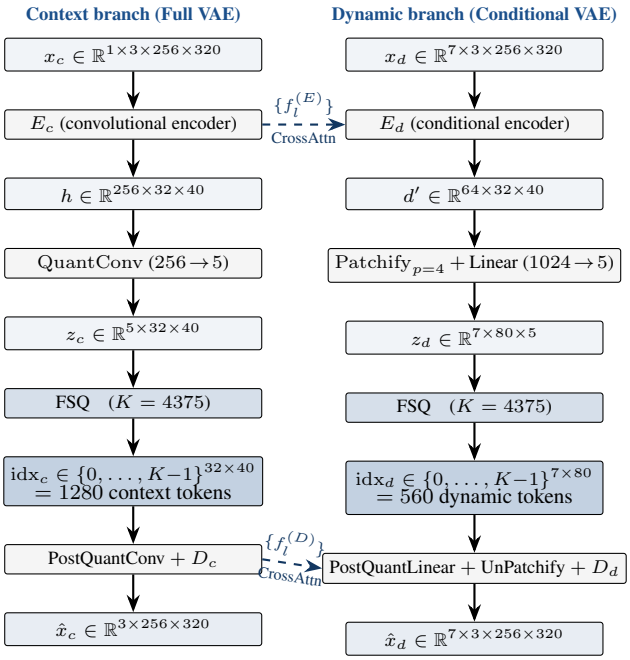


Figure 8: **End-to-end data flow of the Stage 1 tokenizer.** The context frame x_c passes through a Full VAE and is quantized into 1280 context tokens at 8×8 granularity; the 7 dynamic frames x_d pass through a Conditional VAE whose encoder and decoder are cross-attention-conditioned on the context feature pyramids $\{f_l^{(E)}\}$ and $\{f_l^{(D)}\}$ (dashed arrows), producing 560 dynamic tokens at 32×32 granularity. FSQ uses a shared codebook of size $K = 4{,}375$ without any commitment loss, and both decoders depend solely on the discrete indices – no skip connection bypasses the quantization bottleneck.

Training objective. Let $\hat{x} := [\hat{x}_c;\, \hat{x}_d]$. The generator loss combines reconstruction, perceptual, and adversarial terms,

$$\mathcal{L}_G \;=\; \lambda_r\big(\mathcal{L}^d_{\mathrm{rec}} + \mathcal{L}^c_{\mathrm{rec}}\big) + \lambda_p\big(\mathcal{L}^d_{\mathrm{perc}} + \mathcal{L}^c_{\mathrm{perc}}\big) + \lambda_a \cdot \mathbb{1}[t \geq T_{\mathrm{adv}}] \cdot \mathcal{L}^G_{\mathrm{adv}},$$

with

$$\mathcal{L}^{d}_{\text{rec}} = \tfrac{1}{t_d\,3HW}\sum_{k=1}^{t_d}\|x_d^{(k)} - \hat{x}_d^{(k)}\|_1, \quad \mathcal{L}^{c}_{\text{rec}} = \tfrac{1}{3HW}\|x_c - \hat{x}_c\|_1,$$
$$\mathcal{L}^{\bullet}_{\text{perc}} = \textstyle\sum_l w_l\|\phi_l(\tilde{x}^{\bullet}) - \phi_l(\tilde{\hat{x}}^{\bullet})\|_2^2, \quad \tilde{x} = 2x - 1, \qquad \mathcal{L}^{G}_{\text{adv}} = -\mathbb{E}\big[D_\psi(\hat{x})\big],$$

where ϕ_l denotes ImageNet-pretrained VGG-16 features with LPIPS linear weights $\{w_l\}$. The discriminator D_ψ is a depth-6 PatchGAN trained with the hinge objective and an R_1-style gradient penalty,

$$\mathcal{L}_D = \mathbb{E}\big[\max(0,\,1 + D_\psi(\hat{x}))\big] + \mathbb{E}\big[\max(0,\,1 - D_\psi(x))\big] + \gamma\,\mathbb{E}_x\big[(\|\nabla_x D_\psi(x)\|_2 - 1)^2\big].$$

The adversarial term is disabled for the first $T_{\text{adv}} = 10{,}000$ steps; thereafter G and D are updated alternately under a shared global-step counter.

Data and augmentation. Training follows the Open X-Embodiment (OXE) mixed-dataset protocol, with mixing weights proportional to the sample counts of each sub-dataset; the numbers reported in this appendix are obtained on the Bridge V2 subset for reproducibility. For each trajectory we sample a start index uniformly and extract a clip of length $T = 8$ with stride 1. Photometric and geometric augmentations are sampled once per clip and shared across all T frames (Table 6).

Table 6: Clip-wise augmentation ranges used by the Stage 1 tokenizer.

Augmentation	Range
Brightness / contrast / saturation	$[0.9,\ 1.1]$
Hue	$[-0.05,\ 0.05]$
`RandomResizedCrop` scale	$[0.8,\ 1.0]$
`RandomResizedCrop` aspect ratio	$[1.0,\ 1.4]$

Hyperparameters. Table 7 lists the Stage 1 hyperparameters. Generator components, discriminator components, and validation-set metrics are logged as separate TensorBoard scalar groups. Validation is run every 1,000 steps over 100 batches, reporting mean L1 and LPIPS together with a few reconstruction visualizations.

Table 7: Stage 1 tokenizer hyperparameters.

Item	Value
Resolution $H \times W$	256×320
Clip length T / context t_c	$8\ /\ 1$
Context latent channels C_h	256
Dynamic latent channels	64
Patch size p	4
FSQ $(D, \mathbf{L}, K)$	$(5,\ (7,5,5,5,5),\ 4375)$
Tokens per clip	$1280 + 7 \times 80 = 1840$
Discriminator depth	6
Reconstruction loss	L1
Loss weights $(\lambda_r, \lambda_p, \lambda_a)$	$(1.0,\ 1.0,\ 0.1)$
Gradient-penalty coefficient γ	10
Adversarial start step T_{adv}	10,000
Optimizer	AdamW, $(\beta_1, \beta_2, \varepsilon) = (0.9, 0.999, 10^{-8})$, weight decay 0
Learning rate (G / D)	5×10^{-4} / 5×10^{-4}
LR schedule (G / D)	cosine / constant-with-warmup, warmup 5,000 steps
Gradient clipping	$\|\nabla\|_2 \le 1.0$
Mixed precision	bf16
Gradient checkpointing	enabled
Per-GPU batch / accumulation	$1\ /\ 4$
GPUs	$8\times$ A100-40GB
Global effective batch	32
Total training steps	60,000
Checkpoint / validation frequency	every 2,000 / 1,000 steps

B.3 Stage 2: Autoregressive Transformer World Model

Stage 2 trains a causal Transformer π_θ on top of the *frozen* Stage 1 tokenizer. With $\mathbf{c} \in \{K, \ldots, 2K-1\}^{N_c}$ the context tokens, $\mathbf{d}_t \in \{0, \ldots, K-1\}^{N_d}$ the dynamic tokens of frame t,

and $\mathbf{a}_t \in \{2K, \ldots, 2K + B_a - 1\}^{D_a}$ the action tokens at step t, the model factorizes

$$p_\theta\big(\mathbf{d}_2, \mathbf{d}_3, \ldots, \mathbf{d}_T \;\big|\; \mathbf{c},\, \mathbf{d}_1,\, \mathbf{a}_1, \ldots, \mathbf{a}_{T-1}\big).$$

Joint vocabulary. Visual and action tokens are merged into a single vocabulary of size V via fixed offsets, so that a standard LM head can emit either modality (Table 8). The two visual segments share the same FSQ codebook but are disambiguated by an offset of K, allowing the embedding matrix $W_E \in \mathbb{R}^{V \times d}$ to learn separate embeddings for context and dynamic semantics.

Table 8: Stage 2 joint vocabulary.

Range	Semantics	Size
$[0,\ K)$	Dynamic visual tokens (FSQ indices of E_d)	$K = 4375$
$[K,\ 2K)$	Context visual tokens (FSQ indices of E_c, offset by K)	$K = 4375$
$[2K,\ 2K + B_a)$	Action tokens	$B_a = 256$
$\{2K + B_a\}$	BOS (reserved)	1
$\{2K + B_a + 1\}$	EOS (reserved)	1
Total V		**9008**

Action discretization. Each dimension of $\mathbf{a}_t \in \mathbb{R}^{D_a}$ is independently mapped into $B_a = 256$ equal-width bins. The per-dimension lower and upper bounds $(a_i^{\min},\ a_i^{\max})$ are pre-computed on the training split and persisted as an offline lookup table shared between training and inference:

$$\mathrm{bin}_i(a_{t,i}) = \left\lfloor B_a \cdot \frac{a_{t,i} - a_i^{\min}}{a_i^{\max} - a_i^{\min} + \varepsilon} \right\rfloor \in \{0, \ldots, B_a - 1\}, \qquad a_{t,i}^{\text{tok}} = \mathrm{bin}_i(a_{t,i}) + 2K.$$

Sequence layout and loss masking. The data module concatenates the segments into

$$\mathbf{x} = \big[\underbrace{\mathbf{c}}_{1280} \,\|\, \underbrace{\mathbf{d}_1 \| \mathbf{a}_1}_{80+13} \,\|\, \underbrace{\mathbf{d}_2 \| \mathbf{a}_2}_{80+13} \,\|\, \cdots \,\|\, \underbrace{\mathbf{d}_T \| \mathbf{a}_T}_{80+13}\big] \in \{0, \ldots, V-1\}^S,$$

where $\mathbf{a}_T$ is filled with the action associated with the last transition. The total sequence length is

$$S = N_c + t_d \cdot (N_d + D_a) = 1280 + 7 \cdot (80 + 13) = 1931.$$

The label sequence has the same shape as $\mathbf{x}$; all positions except the dynamic tokens of frames $2, \ldots, T$ are set to the ignore index and excluded from the cross-entropy loss (Table 9). The number of supervised positions per sample is

$$|\mathcal{M}| = (t_d - 1) \cdot N_d = 6 \cdot 80 = 480.$$

Since a single causal forward pass simultaneously supervises the predictions of frames $2, \ldots, T$, this objective is equivalent to $t_d - 1 = 6$ one-step next-frame prediction subtasks that share attention compute, i.e. multi-step prediction (MSP).

Table 9: Loss-mask layout of a single Stage 2 sample.

Segment	Label
Context tokens ($N_c = 1280$)	-100 (condition)
Dynamic tokens at frame 1 ($N_d = 80$)	-100 (initial state)
Dynamic tokens at frames $2{:}T$ ($6 \cdot 80 = 480$)	ground-truth token ids
All action tokens ($7 \cdot 13 = 91$)	-100 (condition)

Backbone. π_θ is a causal LLaMA configured as in Table 10. The causal mask is realized inside the FlashAttention-2 kernel; weights, activations, and gradients are all stored in bf16, since FlashAttention-2 is incompatible with fp32.

Training objective. Let $h_{i-1} \in \mathbb{R}^d$ denote the last-layer hidden state at position $i-1$ and $W_O \in \mathbb{R}^{d \times V}$ the LM head, so that

$$p_\theta(x_i \mid x_{<i}) = \big[\mathrm{softmax}(W_O\, h_{i-1})\big]_{x_i}.$$

The training loss is the causal cross-entropy restricted to the supervised positions $\mathcal{M}$,

$$\mathcal{L}_{\text{VGPT}}(\theta) = -\frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \log p_\theta(x_i \mid x_{<i}),$$

and the perplexity $\mathrm{PPL} = \exp(\mathcal{L}_{\text{VGPT}}) \in [1,\ V]$ serves as an interpretable monitoring metric.

Table 10: Stage 2 Transformer backbone.

Component	Value
Layers	12
Hidden size d	768
Attention heads	12 (no GQA)
FFN size	3072
Activation	SwiGLU
Normalization	RMSNorm, $\varepsilon = 10^{-6}$
Positional encoding	RoPE
Max positions	8192
Tied word embeddings	disabled
Parameters (incl. embeddings)	$\approx 1.38 \times 10^8$
Attention kernel	FlashAttention-2
Compute precision	bf16

Tokenization during training. Tokenization is performed inside the data pipeline with all Stage 1 parameters frozen and detached from the computation graph:

$$(\mathbf{c}, \mathbf{d}_{1:T}) = \mathcal{T}_{\text{Stage1}}(x_{1:T}), \qquad \mathbf{a}_{1:T-1} = \text{discretize}(a_{1:T-1}),$$

followed by the offsetting and concatenation above. The Transformer therefore only ever consumes integer token ids, and gradients flow exclusively through π_θ.

Hyperparameters. Stage 2 hyperparameters are summarized in Table 11. Validation computes $\mathcal{L}_{\text{VGPT}}$ and PPL on a held-out split every 5,000 steps. At inference, autoregressive sampling uses temperature $\tau = 1.0$ with top-k and top-p truncation; beam search is not used.

Table 11: Stage 2 autoregressive Transformer hyperparameters.

Item	Value
Backbone	LLaMA-12L-768d-12h (FlashAttention-2, bf16)
Vocabulary size V	9008
Sequence length S	1931 (max positions 8192)
Supervised positions $\lvert\mathcal{M}\rvert$	480
Frozen tokenizer	Stage 1 checkpoint at 60,000 steps
Data module	Context + MSP processor
Optimizer	AdamW, $(\beta_1, \beta_2, \varepsilon) = (0.9, 0.999, 10^{-8})$, weight decay 0
Learning rate	5×10^{-5}
LR schedule	constant-with-warmup, warmup 5,000 steps
Gradient clipping	$\lVert\nabla\rVert_2 \leq 1.0$
Dropout / stochastic depth	0
Per-GPU batch / accumulation	4 / 1
GPUs	8× A100-40GB
Global effective batch	32
Max steps $T_{\max}$	1,000,000
Checkpoint / validation frequency	every 10,000 / 5,000 steps

B.4 Dataset: Bridge V2 Instantiation

The two-stage pipeline is dataset-agnostic and only assumes synchronized image observations paired with low-dimensional action sequences. For reproducibility we instantiate it on **Bridge V2** (Walke *et al.*, 2023) as a representative case; any dataset satisfying the same interface can be substituted without further changes.

Bridge V2 is a large-scale real-robot manipulation dataset collected with a WidowX-250 six-DoF arm across 24 indoor kitchen and tabletop environments. It contains approximately 60,096 human-teleoperated trajectories covering 13 skill categories (grasping, placing, pushing, pulling, opening and closing drawers, flipping, stacking, and so on). Each trajectory logs synchronized third-person RGB, end-effector pose, and gripper state at 5 Hz. The dataset is distributed in the TensorFlow Datasets (TFDS) format; we use version 1.0.0.

Raw sample structure. Each raw trajectory is a variable-length sequence with the fields summarized in Table 12, where L is the trajectory length (typically 20–60 frames). The data is distributed as TFRecord shards and split into training and validation partitions.

Table 12: Core fields of a Bridge V2 trajectory.

Field	Shape	Dtype	Description
Primary RGB	$(L,\ 480,\ 640,\ 3)$	uint8	Third-person primary-camera observation
Proprio. state	$(L,\ 7)$	float32	End-effector $(x, y, z,$ roll, pitch, yaw, gripper$)$
Action	$(L,\ 7)$	float32	EE increments $(\Delta x, \Delta y, \Delta z, \Delta$roll, Δpitch, Δyaw, gripper_cmd$)$
Language instruction	string	—	Natural-language task description (unused here)
Terminal flag	bool	—	Episode-end indicator

Pre-processing (TFDS → per-trajectory arrays). The TFDS source is converted offline into per-trajectory array files once, so that the downstream data pipeline does not need to touch TFRecords again:

- **Image resampling.** $480\times640 \to H\times W = 256\times320$ via aspect-ratio-preserving resize followed by center cropping, stored as uint8.
- **Action expansion.** The raw $\mathbb{R}^7$ action is expanded to $\mathbb{R}^{13}$ by appending several dimensions of the previous end-effector state, matching the shared OXE action layout ($D_a = 13$).
- **Action-range statistics.** The training split is scanned once to record per-dimension $(a_i^{\min}, a_i^{\max})$; the resulting table is persisted and reused by the equal-width binning in §B.3.

After conversion, each trajectory becomes an independent array file containing an image tensor $(L,\ 256,\ 320,\ 3)$ in uint8, an action tensor $(L,\ 13)$ in float32, and a proprioceptive state $(L,\ 7)$ in float32. The resulting corpus has ≈60,096 trajectories, over 1.92 M frames, and occupies roughly 260 GB on disk.

Sampling and slicing. Each optimization step samples a clip according to:

1. Uniformly sample an episode from the training split.
2. Uniformly sample a start frame $s \in [0,\ L - T]$.
3. Take $\{I_s, \ldots, I_{s+T-1}\}$ and the paired actions $\{a_s, \ldots, a_{s+T-1}\}$ with $T = 8$ and stride 1.
4. Apply the clip-wise photometric and geometric augmentations of Table 6, with parameters shared across the T frames to preserve temporal consistency.
5. Normalize images to $[0,\ 1]$ before feeding them to the tokenizer.

Stage 1 consumes only the pixels $\{I_{s+k}\}_{k=0}^{T-1}$; Stage 2 additionally consumes the paired actions to construct the joint token sequence of §B.3. The two stages share identical sampling and augmentation logic, and differ only in their collate-level outputs.

B.5 Reproduction Protocol

We describe the end-to-end reproduction protocol in terms of inputs, computation, and outputs, deliberately avoiding prescriptive script layouts or launch commands.

Software environment. The implementation is built on PyTorch with the version constraints in Table 13. Distributed training uses Accelerate over the NCCL backend; both stages train under bf16 mixed precision to satisfy the dtype constraints of FlashAttention-2.

Step 0: Offline pre-processing of raw data. This step is CPU-only and converts TFRecord shards into the per-trajectory array files of §B.4, reporting the total number of converted trajectories as a consistency check. It runs once per dataset release and takes ≈2–3 hours on a single CPU node at the scale of the example dataset.

Step 1: Train the context-aware tokenizer. Stage 1 is trained on the pre-processed per-trajectory data with the hyperparameters of §B.2: global effective batch 32, base learning rate 5×10^{-4},

Table 13: Key software dependencies.

Component	Version
PyTorch	2.3
FlashAttention	2.5
HuggingFace Accelerate	0.30
HuggingFace Transformers	latest stable
HuggingFace Diffusers	latest stable
TensorFlow (only for TFDS I/O)	2.15
LPIPS / timm / einops	latest stable

discriminator start at step $10{,}000$, 6×10^4 total steps, checkpoint every $2{,}000$ steps, and validation every $1{,}000$ steps on a held-out split (L1 and LPIPS). Activation checkpointing is enabled to reduce memory. The final checkpoint becomes the frozen backbone of Stage 2. Training takes $\approx$72 hours on a single $8\times$ A100-40GB node.

Step 2: Train the autoregressive Transformer. Stage 2 is trained on the joint token sequence of §B.3 with the hyperparameters therein: per-GPU batch 4, no accumulation, global effective batch 32, learning rate 5×10^{-5} with a $5{,}000$-step linear warmup, gradient clipping $\|\nabla\|_2 \leq 1.0$, 10^6 total steps, checkpoint every $10{,}000$ steps, and validation every $5{,}000$ steps. The Stage 1 tokenizer is loaded read-only and remains frozen throughout. Training throughput is $\approx$3.5 it/s on a single $8\times$ A100-40GB node, totalling $\approx$80 hours for the full 10^6 steps.

Monitoring and sanity checks. All scalar and image summaries are logged to TensorBoard. In practice we monitor three families of health indicators: *(i) mixed precision*—bf16 must be explicitly enabled, since FlashAttention-2 refuses to execute under fp32; *(ii) hardware utilization*—compute utilization should remain consistently above 90% with near-uniform per-GPU memory; otherwise the bottleneck typically lies in data loading or NCCL communication; *(iii) training stability*—a smooth generator-loss transition around discriminator activation in Stage 1, and long-term stability of the gradient norm together with a monotone decrease of the held-out PPL in Stage 2. Common failure modes and mitigations are listed in Table 14.

Table 14: Common failure modes observed during training.

Symptom	Root cause	Fix
FlashAttention only supports fp16 / bf16	Mixed precision misconfigured as fp32	Force the mixed-precision policy to bf16
NVML / NVLink symbol missing	Legacy node driver (e.g. the 450.x series)	Disable framework-side NVML probing and turn off NCCL P2P/NVLS fast paths
Nested snapshots and abnormal disk usage under the checkpoint directory	Residuals from previous smoke-test runs	Clean the smoke-test directory before starting the production run

B.6 Optimization Infrastructure

Table 15: Optimization infrastructure shared by Stage 1 and Stage 2.

Item	Value
Framework	PyTorch + HuggingFace Accelerate (multi-GPU)
Distributed backend	NCCL
Mixed precision	bf16 (both stages)
Activation checkpointing	Stage 1 enabled, Stage 2 disabled
Logging	TensorBoard scalar and image summaries
Random seed	set per experiment via Accelerate
Checkpoint format	HuggingFace Safetensors (unwrapped)

Stage 1 produces tokenizer checkpoints at a fixed cadence; Stage 2 loads a specific checkpoint in read-only mode as its frozen backbone. No optimizer state, LR scheduler state, or data-loader state is shared between the two stages.

Table 16: Side-by-side comparison of the two training stages.

Dimension	Stage 1	Stage 2
Input	$x_{1:T} \in \mathbb{R}^{T\times 3\times H\times W}$	$\mathbf{x} \in \{0,\dots,V-1\}^S$
Output	Reconstructions $\hat{x}_{1:T}$ and FSQ indices	Logits $\in \mathbb{R}^{S\times V}$
Objective	$\mathcal{L}_G$ (L1 + LPIPS + hinge-GAN)	$\mathcal{L}_{\mathrm{VGPT}}$ (masked causal CE)
Quantization	FSQ, $K = 4375$	—
Tokens per clip	1840	1931 (supervised 480)
Trainable parameters	Stage 1 only	Stage 2 only (Stage 1 frozen)
Total steps	6×10^4	10^6
Effective batch	32	32
Mixed precision	bf16	bf16
Hardware	8× A100-40GB	8× A100-40GB

B.7 End-to-End Quantitative Summary

Compact formula card. For reference, the end-to-end forward and training equations are collected below:

$$\text{Stage 1 encoders:}\quad z_c = \mathrm{QuantConv}\big(E_c(x_c)\big),\quad z_d = W_q\,\mathrm{Patchify}_p\big(E_d(x_d;\{f_l^{(E)}\})\big),$$

$$\text{Stage 1 quantization:}\quad \hat{z} = \mathcal{Q}_{\mathbf{L}}(z),\quad \mathrm{idx}(\hat{z}) = \sum_i \big(\hat{z}_i + \lfloor L_i/2\rfloor\big)\prod_{j<i} L_j,$$

$$\text{Stage 1 decoders:}\quad \hat{x}_c = D_c\big(\mathcal{Q}(z_c)\big),\quad \hat{x}_d = D_d\big(\mathcal{Q}(z_d);\{f_l^{(D)}\}\big),$$

$$\begin{aligned}\text{Stage 1 loss:}\quad \mathcal{L}_G &= \lambda_r\big(\|x_d-\hat{x}_d\|_1 + \|x_c-\hat{x}_c\|_1\big)\\ &\quad + \lambda_p\big(\mathrm{LPIPS}(x_d,\hat{x}_d) + \mathrm{LPIPS}(x_c,\hat{x}_c)\big)\\ &\quad - \lambda_a\,\mathbb{1}[t\ge T_{\mathrm{adv}}]\,\mathbb{E}\big[D_\psi(\hat{x})\big],\end{aligned}$$

$$\text{Stage 2 tokens:}\quad \mathbf{c}\in[K,2K)^{N_c},\ \mathbf{d}_t\in[0,K)^{N_d},\ \mathbf{a}_t\in[2K,2K+B_a)^{D_a},$$

$$\text{Stage 2 sequence:}\quad \mathbf{x} = \big[\mathbf{c}\,\|\,\mathbf{d}_1\|\mathbf{a}_1\,\|\,\cdots\,\|\,\mathbf{d}_T\|\mathbf{a}_T\big],\quad S = 1931,$$

$$\text{Stage 2 model:}\quad p_\theta(x_i\mid x_{<i}) = \big[\mathrm{softmax}(W_O\,h_{i-1})\big]_{x_i},$$

$$\text{Stage 2 loss:}\quad \mathcal{L}_{\mathrm{VGPT}} = -\tfrac{1}{|\mathcal{M}|}\sum_{i\in\mathcal{M}}\log p_\theta(x_i\mid x_{<i}),\ \ \mathcal{M} = \{\text{positions of }\mathbf{d}_2,\dots,\mathbf{d}_T\},\ |\mathcal{M}| = 480.$$

Constants: $T=8,\ t_c=1,\ t_d=7,\ H\times W = 256\times 320,\ p=4,\ D=5,\ \mathbf{L}=(7,5,5,5,5),\ K=4375,\ N_c=1280,\ N_d=80,\ D_a=13,\ B_a=256,\ V=9008,\ S=1931,\ |\mathcal{M}|=480.$

B.8 TensorBoard Training Curves

This section summarizes the convergence curves of both stages on the Bridge V2 instantiation. All curves are exported directly from TensorBoard scalars without smoothing, with global optimization steps on the horizontal axis.

B.8.1 Stage 1: Tokenizer (GAN + Reconstruction + Perception)

Stage 1 logs four prefix-grouped scalar families. Tables 17–19 report the trends observed over the first 60,000 steps; each TensorBoard tag (quoted verbatim) maps one-to-one to the exported CSV file.

Optimization and throughput. The auxiliary scalar `lr` tracks the generator learning rate (warmup for 5,000 steps, followed by cosine decay); `samples_sec_gpu`, `batch_time`, and `data_time` respectively record per-GPU throughput, per-step wall time, and data-loading time. Reconstruction previews `images/reconstruction_*` are saved every 1,000 steps for qualitative inspection of high-frequency texture fidelity.

Key training phases. **(i)** Steps 0–10,000: pure L1 + LPIPS supervision; reconstruction and perceptual curves descend fastest. **(ii)** Step 10,000: discriminator activation (`disc_start` = 10,000); `gen_loss/gan_loss` becomes nonzero and `step_gen_loss` shows a mild $+0.05$-magnitude rebound. **(iii)** Steps 10,000–60,000: reconstruction and adversarial terms decrease jointly, and `val_loss/*` continues to drop monotonically.

Table 17: Generator loss $\mathcal{L}_G$ and its components (`gen_loss/*`).

Curve	Term	Observed trend ($0 \to 60{,}000$ steps)
`gen_loss/recon_loss`	$\mathcal{L}_{\text{rec}}$ (dynamic-frame L1)	Monotone drop from $\sim$0.11 to $\mathbf{0.0169}$
`gen_loss/perceptual_loss`	$\mathcal{L}_{\text{perc}}$ (dynamic-frame LPIPS)	Monotone drop from $\sim$0.30 to $\mathbf{0.0607}$
`gen_loss/ref_recon_loss`	Context-frame L1	Same magnitude as `recon_loss`; final $\sim$0.02
`gen_loss/ref_perceptual_loss`	Context-frame LPIPS	Tracks `perceptual_loss`; final $\sim$0.07
`gen_loss/gan_loss`	$\mathcal{L}_{\text{adv}}^{G}$ (hinge, $\lambda_a = 0.1$)	0 for step $\leq 10{,}000$; afterwards oscillates in $[0.10,\ 0.40]$; final $\mathbf{0.2546}$
`gen_loss/commit_loss`	VQ commitment (exactly 0 under FSQ)	0 throughout
`gen_loss/dyna_commit_loss`	Dynamic-branch commitment (exactly 0 under FSQ)	0 throughout
`step_gen_loss`	Generator total loss $\mathcal{L}_G$	Drops from $\sim$0.50 to $\mathbf{0.1630}$

Table 18: Discriminator loss $\mathcal{L}_D$ and logits (`disc_loss/*`; logged only for step $\geq 10{,}000$).

Curve	Term	Observed trend
`disc_loss/real_logits`	$\mathbb{E}[D_\psi(x)]$	Mild drift in $[-0.30,\ 0.10]$; final $\mathbf{-0.2297}$
`disc_loss/fake_logits`	$\mathbb{E}[D_\psi(\hat{x})]$	Drifts synchronously with `real_logits`; final $\mathbf{-0.2463}$
`disc_loss/logit_diff`	$\mathbb{E}[D_\psi(x) - D_\psi(\hat{x})]$	Hovers around 0 with amplitude < 0.05, indicating D does not overwhelm G
`step_discr_loss`	Discriminator total loss $\mathcal{L}_D$	Quickly settles into the $[1.5,\ 2.2]$ band after activation; final $\mathbf{1.9864}$, consistent with a healthy GAN balance against $\mathcal{L}_{\text{adv}}^{G}$

Training-curve visualization. Figure 9 shows the four generator-side training curves, and Figure 10 shows the discriminator logits, validation metrics, and learning-rate schedule. In each panel the light trace is the raw sampled values and the dark trace an exponential moving average ($\alpha = 0.08$); a dash-dotted vertical line marks discriminator activation at step 10,000.

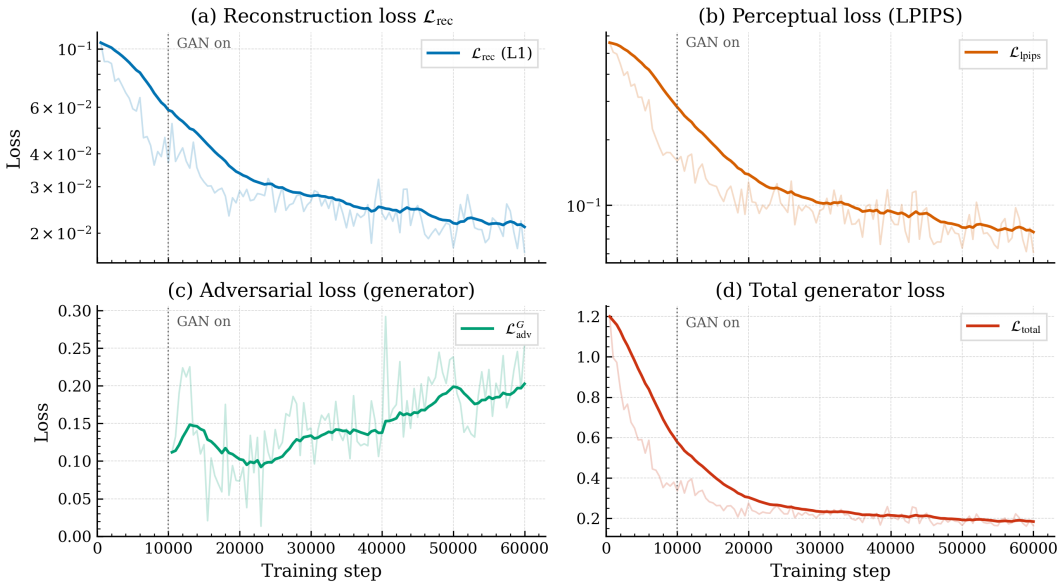


Figure 9: **Stage 1 generator-side training dynamics.** (a) Reconstruction loss $\mathcal{L}_{\text{rec}}$ (`gen_loss/recon_loss`, L1); (b) perceptual loss $\mathcal{L}_{\text{lpips}}$ (`gen_loss/perceptual_loss`); (c) adversarial loss $\mathcal{L}_{\text{adv}}^{G}$ (`gen_loss/gan_loss`); (d) generator total loss $\mathcal{L}_G$ (`step_gen_loss`). Panels (a) and (b) descend fastest before discriminator activation and continue to decrease monotonically afterwards; panel (c) stabilizes in $[0.10,\ 0.40]$ after activation, reflecting the dynamic G–D equilibrium.

Table 19: Validation scalars (`val_loss/*`; averaged over 100 batches every 1,000 steps).

Curve	Meaning	Observed trend
`val_loss/recon_loss`	Mean pixel L1	Monotone drop; best **0.01944** at step 50,001, final 0.01968 at step 59,001
`val_loss/perceptual_loss`	Mean LPIPS	Monotone drop; best **0.06840** at step 50,001, final 0.07147 at step 59,001

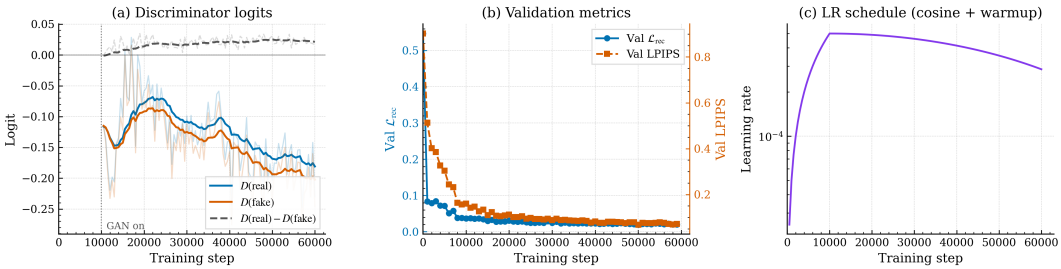


Figure 10: **Stage 1 adversarial balance, validation metrics, and optimization schedule.** (a) Discriminator logits on real and fake samples (`disc_loss/real_logits`, `disc_loss/fake_logits`) and their difference `disc_loss/logit_diff` $= D(\text{real}) - D(\text{fake})$, which hovers around 0 with a mild positive drift and indicates that D does not overwhelm G; (b) validation L1 (`val_loss/recon_loss`, left axis) and LPIPS (`val_loss/perceptual_loss`, right axis), both decreasing monotonically; (c) generator learning rate `lr`, which undergoes a 5,000-step warmup followed by cosine decay.

Reconstruction evolution. To visualize how the numerical trends translate into perceptual quality, Figure 11 shows pixel-space reconstructions from the Stage 1 tokenizer at six representative training steps ($\{1,\ 2{,}500,\ 10{,}000,\ 25{,}000,\ 50{,}000,\ 60{,}000\}$). Each training step is rendered as a pair of adjacent rows: the upper row shows the ground-truth frames x and the lower row the decoder outputs $\hat{x}$; columns uniformly sample 4 temporal positions ($t_1,\ t_3,\ t_5,\ t_7$) out of the $T = 8$ clip frames, so that both spatial fidelity and temporal consistency become visually apparent.

B.8.2 Stage 2: Autoregressive Transformer (Causal CE + Perplexity)

Stage 2 is monitored with TensorBoard scalars centered on the masked causal cross-entropy objective and its induced perplexity. Figure 12 reports the exported raw traces together with an exponential moving average (EMA, $\alpha = 0.9$), avoiding manually tabulated estimates from the plots.

Observed convergence. The training loss drops sharply at the beginning of optimization and then enters a slower refinement regime, with the EMA curve continuing to decrease over the full training horizon. The held-out loss follows the same overall pattern: a rapid early reduction followed by a gradual flattening, without a late-stage upward trend. The evaluation perplexity is consistent with the held-out cross-entropy curve, decreasing rapidly in the early phase and then approaching a stable low-variance regime. Together, these curves indicate that the Stage 2 Transformer learns the token dynamics early and continues to refine its next-token distribution over long training, while the validation metrics do not show visible signs of divergence from the training trajectory.

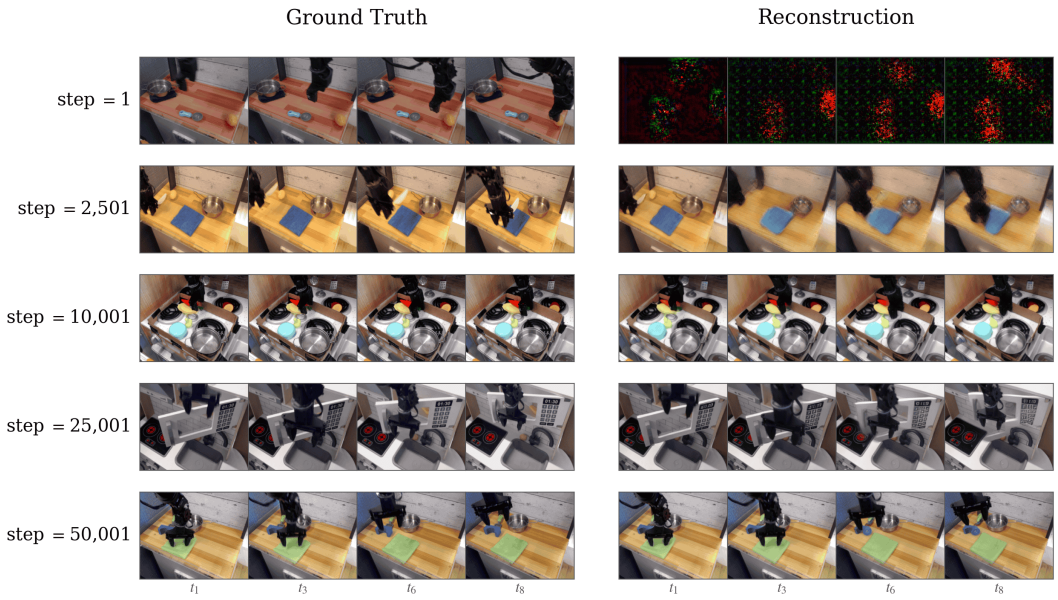


Figure 11: **Evolution of Stage 1 reconstructions across training steps.** Horizontal axis: temporal frame position t_i inside a clip. Vertical axis: training steps from early to late; every two rows form a comparison pair, with the upper row showing the ground-truth frame and the lower row showing the decoded $\hat{x}$ from the same FSQ indices. At step $\leq 10^3$ the dynamic branch fails to recover high-frequency textures occluded by the manipulator, and reconstructions exhibit global blur and mild colour bias; around step 10^4 (discriminator activation) texture sharpness improves noticeably, yet locally visible grid-like artefacts still contribute a non-negligible LPIPS gap; beyond step 2.5×10^4, reconstructions become visually indistinguishable from ground truth, with residual errors concentrated on the narrow contact region between the end-effector and the manipulated object. This qualitative evolution is step-aligned with the monotone L1/LPIPS curves in Figures 9–10 and with the best-validation step (5×10^4) of `val_loss/*`.

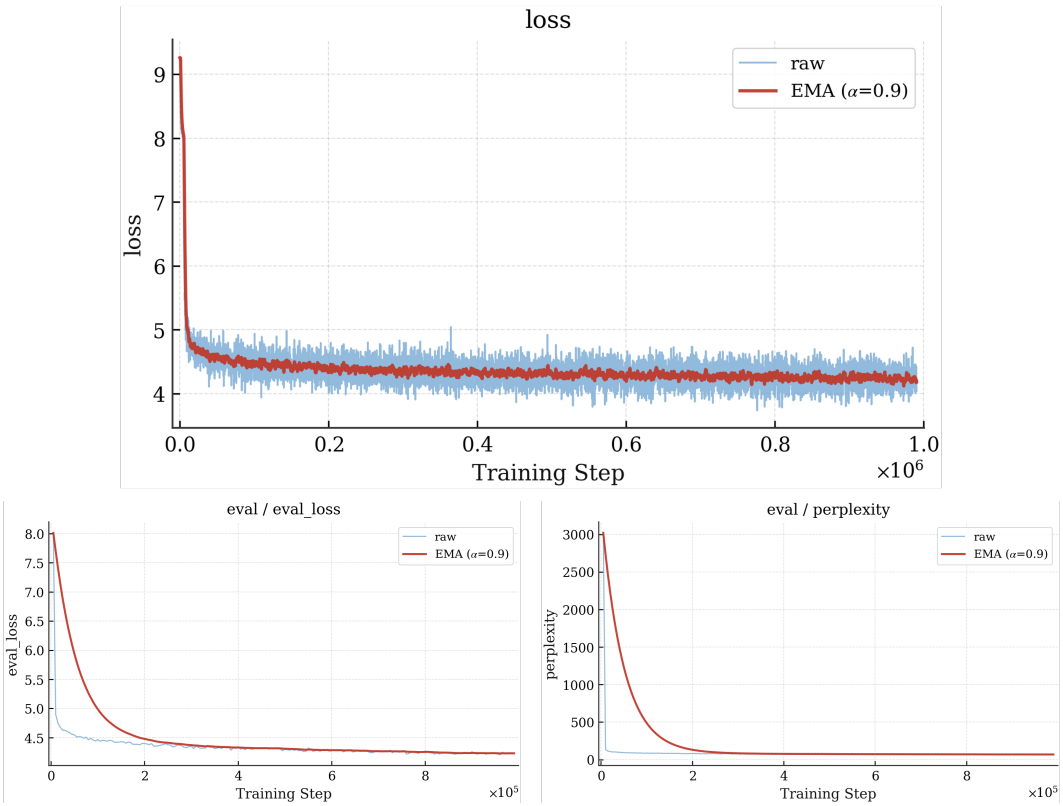


Figure 12: **Stage 2 autoregressive Transformer training curves.** Top: training masked causal cross-entropy loss. Bottom left: held-out evaluation loss. Bottom right: held-out evaluation perplexity. Each panel shows both the raw TensorBoard trace and an EMA-smoothed curve with $\alpha = 0.9$.

C Stylized derivation for Proposition 1

We provide the simplified derivation underlying the main-text stability discussion. The goal is to illustrate why periodic refresh can limit long-horizon drift under a local contraction assumption, rather than to claim a complete dynamical model of the tokenizer–decoder pair or a quantitative predictor of the exact gains observed in experiments.

Setup and notation. Consider a token-based autoregressive world model generating a rollout of T frames partitioned into $K = \lceil T/W \rceil$ segments of window size W. Let x_t^* denote the ground-truth frame at step t and $\hat{x}_t$ the predicted frame. We define the frame-level error as $e_t = \|\hat{x}_t - x_t^*\|$. We make two assumptions:

A1. **Bounded per-step error.** Conditioned on a correct context, the single-step prediction error satisfies $e_t \le \varepsilon$ for all t. More generally, if the context carries an error η, the prediction error at the next step satisfies $e_t \le \varepsilon + \alpha\,\eta$ for some contraction factor $\alpha \in [0, 1)$.

A2. **Bounded quantization error.** The decode–re-encode cycle introduces a quantization error bounded by δ_q: for any predicted frame $\hat{x}$, $\|\mathcal{D}_{\text{vis}}(\mathcal{T}_{\text{vis}}(\hat{x})) - \hat{x}\| \le \delta_q$.

Standard autoregressive generation (no re-encoding). Under native AR decoding, the context is never refreshed, so context-carried error keeps accumulating across the full horizon. If a contraction assumption analogous to **A1** held globally, one would obtain the geometric series

$$e_t \;\le\; \varepsilon \sum_{i=0}^{t-1} \alpha^i \;=\; \frac{\varepsilon\,(1-\alpha^t)}{1-\alpha}.$$

For any fixed $\alpha < 1$, this is bounded by $\varepsilon/(1-\alpha)$, but this constant can still blow up as $\alpha \to 1$. In the non-contractive worst case, the accumulation becomes linear in the horizon, yielding the coarse comparison bound

$$\mathcal{E}_{\text{AR}}(T) \;=\; \max_{t \le T} e_t \;\le\; T\,\varepsilon. \tag{11}$$

The AR regime therefore either pays a horizon-independent but potentially large constant $\varepsilon/(1-\alpha)$, or a linear-in-T bound when contraction fails; SWR will instead replace this with a window-size-controlled constant.

Sliding-window re-encoding. With SWR, at each segment boundary $t = kW$, the model decodes the last predicted frame $\hat{x}_{kW}$ to pixel space and re-encodes it as a fresh context/state pair $(z_{k+1}^{\text{ctx}},\, z_0^{\text{dyn}}) = \mathcal{T}_{\text{vis}}(\hat{x}_{kW})$; concretely, $\hat{x}_{kW}$ is treated both as the new conditioning frame for the Full-VAE branch E_c and as a length-one dynamic clip for the Conditional-VAE branch E_d (Appendix B.2).

Step 1: Within-segment error. Within segment k, the context z_k^{ctx} is fixed and the model generates at most W frames. By **A1**, the error at relative step $j \in \{1, \dots, W\}$ within the segment satisfies:

$$e_{(k-1)W+j} \;\le\; \varepsilon \sum_{i=0}^{j-1} \alpha^i + \alpha^j\,\eta_k,$$

where η_k is the error carried by the context z_k^{ctx}. For $j \le W$, using $\sum_{i=0}^{j-1} \alpha^i = \frac{1-\alpha^j}{1-\alpha} \le j \le W$ and $\alpha^j \le 1$:

$$e_{(k-1)W+j} \;\le\; \frac{\varepsilon(1-\alpha^W)}{1-\alpha} + \alpha^W \eta_k \;\le\; W\varepsilon + \eta_k. \tag{12}$$

Taking $j = W$ gives the tighter intermediate bound $e_{kW} \le W\varepsilon + \alpha^W \eta_k$, which we use in Step 2.

Step 2: Cross-segment error refresh. At the boundary, the context error of segment $k{+}1$ is:

$$\eta_{k+1} = \|z_{k+1}^{\text{ctx}} - z_{k+1}^{\text{ctx},*}\| \le \|\hat{x}_{kW} - x_{kW}^*\| + \delta_q = e_{kW} + \delta_q.$$

Crucially, z_{k+1}^{ctx} is obtained by re-encoding the *decoded* frame, so the next segment depends on the current decoded observation rather than the entire raw token history of all prior segments. Substituting the tighter form of Eq. 12 at $j = W$:

$$\eta_{k+1} \;\le\; W\varepsilon + \alpha^W \eta_k + \delta_q.$$

Step 3: Steady-state bound. Since $\alpha^W < 1$, the recurrence $\eta_{k+1} \le W\varepsilon + \alpha^W \eta_k + \delta_q$ converges to a fixed point:

$$\eta^* = \frac{W\varepsilon + \delta_q}{1 - \alpha^W}.$$

Starting from $\eta_1 = 0$ (ground-truth conditioning frame), by induction $\eta_k \le \eta^*$ for all k. Combining with the loose within-segment bound $e_{(k-1)W+j} \le W\varepsilon + \eta_k$ from Eq. 12 (which upper-bounds $\alpha^W \eta_k$ by η_k), the maximum frame-level error in any segment is therefore:

$$\mathcal{E}_{\mathrm{SWR}}(T) = \max_{t \le T} e_t \;\le\; W\varepsilon + \eta^* = W\varepsilon + \frac{W\varepsilon + \delta_q}{1 - \alpha^W}. \tag{13}$$

In the simplified case $\alpha \to 0$ (errors do not propagate within the causal Transformer's effective receptive field beyond one step), $\alpha^W \to 0$ and $\eta^* \to W\varepsilon + \delta_q$, so the bound becomes $\mathcal{E}_{\mathrm{SWR}}(T) \le 2W\varepsilon + \delta_q$. More generally, for any $\alpha \in [0, 1)$ and any window size W such that $\alpha^W < 1$, the bound in Eq. 13 is finite and does not grow explicitly with T, showing that under this stylized local model the effect of SWR is governed by the refresh window and the local error parameters rather than the rollout horizon alone. We do not estimate α or δ_q from the trained model; instead, the derivation is meant to justify the qualitative window-size trade-off observed in Table 3 and Appendix G, where moderate refresh intervals outperform both overly frequent and overly infrequent refresh. □

Asymptotic optimum of the window size. Treating W as a positive real variable, let

$$f(W) \;=\; W\varepsilon + \frac{W\varepsilon + \delta_q}{1 - \alpha^W}.$$

This is the SWR bound of Eq. 13 written as a function of W at fixed $(\varepsilon, \delta_q, \alpha)$. The qualitative trade-off discussed in Remark 1 in the main text can be made precise by analyzing the first-order stationarity of f.

Corollary 1 (Asymptotic optimum W^*). *Assume $\alpha \in (0, 1)$ and $\delta_q \ge 0$. In the moderate-window regime $\alpha^W \ll 1$, the bound $f(W)$ is quasi-convex in W and attains its minimum at*

$$W^* \;\approx\; \frac{1}{-\ln \alpha} \ln\!\left(\frac{(-\ln \alpha)\, \delta_q}{2\,\varepsilon} \right), \tag{14}$$

provided the argument of the logarithm exceeds 1; otherwise the minimum is achieved at the smallest admissible W (i.e. the quantization cost δ_q is too small relative to ε to justify any batching). In particular, W^* is logarithmic in δ_q/ε and in $1/(-\ln \alpha)$.

Proof. In the regime $\alpha^W \ll 1$, we expand $1/(1 - \alpha^W) \approx 1 + \alpha^W + \mathcal{O}(\alpha^{2W})$ in Eq. 13, giving

$$f(W) \;\approx\; 2W\varepsilon + \delta_q \;+\; \alpha^W \,(W\varepsilon + \delta_q) \;+\; \mathcal{O}(\alpha^{2W}).$$

Differentiating with respect to W and retaining leading-order terms (using $\frac{d}{dW}\alpha^W = \alpha^W \ln \alpha$ and discarding the subleading $\alpha^W \varepsilon$ contribution),

$$\frac{df}{dW} \;\approx\; 2\varepsilon \;+\; \alpha^W \,\ln \alpha \,\big(W\varepsilon + \delta_q\big).$$

Setting $df/dW = 0$ and using $\ln \alpha < 0$,

$$2\varepsilon \;=\; \alpha^W \,(-\ln \alpha)\,\big(W\varepsilon + \delta_q\big).$$

In the typical regime $\delta_q \gg W\varepsilon$ (single-step token error small relative to the decode–re-encode distortion, as observed in our token-level probes), the right-hand side is dominated by δ_q, giving

$$\alpha^W \;\approx\; \frac{2\varepsilon}{(-\ln \alpha)\,\delta_q}, \qquad W^* \;=\; \frac{1}{-\ln \alpha} \ln\!\Big(\tfrac{(-\ln \alpha)\,\delta_q}{2\,\varepsilon} \Big),$$

which is Eq. 14. The second-derivative test at this point gives $d^2 f/dW^2 \approx -(\ln \alpha)^2 \alpha^W (W\varepsilon + \delta_q) > 0$, so f is locally convex and W^* is a minimum. When the argument of the outer logarithm drops below 1, i.e. $\delta_q \le 2\varepsilon/(-\ln \alpha)$, the stationarity condition has no interior solution and f is monotone decreasing over its admissible range; the constrained minimum then lies at the smallest W permitted by the inference hardware or the decoder receptive field. □

Physical interpretation. Corollary 1 attaches a concrete meaning to the three parameters of the stylized model. **(i) Role of the contraction factor α.** As $\alpha \to 1^-$ (within-segment error is barely damped), the prefactor $1/(-\ln \alpha)$ diverges and W^* grows, i.e. the model benefits from *longer* segments that amortize the one-time refresh cost; in the opposite limit $\alpha \to 0^+$ (errors decay within a single step), short segments suffice and W^* is small. **(ii) Role of the noise ratio δ_q/ε.** Large δ_q/ε (each refresh is comparatively expensive) pushes W^* up, while small δ_q/ε (refresh is nearly free) pushes W^* down; the logarithmic dependence means that W^* is insensitive to the precise value of δ_q and ε within an order of magnitude. **(iii) Consistency with the ablation.** In our setting we expect the tokenizer–decoder cycle to be moderately contractive and the decode–re-encode distortion δ_q to dominate the single-step token error ε, so Eq. 14 predicts W^* to lie in the small-integer regime. Empirically, Appendix G finds $W = 6$ optimal among $\{4, 6, 8, 10, 15\}$, which is consistent with this regime; $W = 4$ pays too much refresh overhead while $W \geq 10$ leaves too much within-segment drift, as Corollary 1 predicts. These conclusions are meant only to explain the *shape* of the W–quality trade-off under the stylized assumptions **A1**–**A2**; we do not claim that Eq. 14 is a numerical predictor of the exact optimum, nor that a single global $(\alpha, \varepsilon, \delta_q)$ triple suffices to describe every manipulation task in the benchmark.

Comparison with vanilla AR decoding. The two regimes can be placed on a common axis by viewing $\mathcal{E}_{\mathrm{AR}}(T)$ of Eq. 11 against $\mathcal{E}_{\mathrm{SWR}}(T)$ of Eq. 13. This yields the following direct consequence, which makes the T–dependence gap explicit rather than only qualitative.

Corollary 2 (SWR versus AR asymptotic gap). *Under Assumptions **A1**–**A2** and for any W with $\alpha^W < 1$,*

$$\mathcal{E}_{\mathrm{AR}}(T) \;-\; \mathcal{E}_{\mathrm{SWR}}(T) \;\geq\; T\varepsilon \;-\; \Big(W\varepsilon + \tfrac{W\varepsilon+\delta_q}{1-\alpha^W}\Big), \tag{15}$$

so the advantage of SWR grows at least linearly in the rollout horizon T once T exceeds the horizon-independent SWR constant. In the contractive regime $\alpha < 1$ with $\alpha^W \ll 1$, the SWR constant simplifies to $\approx 2W\varepsilon + \delta_q$, and the crossover horizon above which SWR strictly dominates AR is $T^\dagger \approx W + \delta_q/\varepsilon$.

Proof. Subtract Eq. 13 from the non-contractive-regime bound of Eq. 11 directly. The simplified constant follows from $(W\varepsilon + \delta_q)/(1 - \alpha^W) \to W\varepsilon + \delta_q$ as $\alpha^W \to 0$, and solving $T^\dagger\varepsilon \geq 2W\varepsilon + \delta_q$ for $T^\dagger$ gives the stated crossover. □

Eq. 15 turns the qualitative "does not grow with T" statement in the main text into an order-of-magnitude prediction: the longer the rollout, the larger the SWR advantage, with the crossover controlled by the ratio δ_q/ε rather than by T itself.

Discussion of assumptions. Assumption **A1** replaces the true non-stationary rollout dynamics with a locally linear contraction around the on-policy trajectory; this is standard in stylized error-propagation analyses of autoregressive generation (see, e.g., the attention-sink regime of [15]) and is used here only to obtain a closed-form window-size trade-off. A global contraction assumption on the full horizon would in general fail for autoregressive token models, which is precisely the empirical observation that motivates SWR; our analysis therefore imposes contraction only *within* a segment of length W, consistent with the segmented generation procedure of Eq. 9. Assumption **A2** treats the decode–re-encode cycle as a bounded, input-independent distortion. This is a conservative over-approximation: in practice δ_q depends on the content of the refreshed frame, but the *worst-case* bound we derive only requires a uniform upper bound, which is furnished by the FSQ tokenizer's finite codebook. Finally, the corollaries above are invariant under any affine rescaling of (ε, δ_q): only the ratio δ_q/ε enters W^*, which explains why the optimum window is robust to the specific scale on which token-level error is measured.

D Pseudocode for Reward-Aligned Post-Training

Algorithm 1 summarizes the complete ROBOALIGN-R1 training pipeline, covering all four stages described in §3.2. Color coding marks each stage: **benchmark construction** (blue), **teacher judge training** (green), **student reward distillation** (purple), and **GRPO post-training** (red).

Algorithm 1 Reward-Aligned Post-Training of ROBOALIGN-R1

Require: Pre-trained world model p_θ; robot datasets {RT-1, Bridge, CALVIN, LIBERO}; T2V model; base VLM (Qwen3-VL-8B-Thinking)
Ensure: Post-trained world model p_{θ^*}

— Stage 1: Benchmark Construction —
for each instruction l in robot datasets **do**
  Sample ground-truth video v^+ from dataset
  Generate candidate video $v^- \leftarrow \mathrm{T2V}(l)$
  Annotate (l, v) with raw rubric scores $\mathbf{r} = (r_1, \ldots, r_6)$ using ranges $[3, 2, 1, 1, 1, 2]$ ▷ Six dimensions
end for
$\mathcal{D}_{\text{bench}} \leftarrow \{(l_i, v_i, \mathbf{r}_i)\}_{i=1}^{N}$

— Stage 2: Teacher Judge Training —
Initialize teacher f_ϕ from Qwen3-VL-8B-Thinking
Fine-tune f_ϕ on $\mathcal{D}_{\text{bench}}$: $\mathcal{L}_{\text{teacher}} = -\mathbb{E}\big[\log p_\phi(\mathbf{r} \mid l, v)\big]$ ▷ $f_\phi \rightarrow$ ROBOALIGN-JUDGE

— Stage 3: Student Reward Distillation —
Initialize student g_ψ (compact visual–text encoder + linear head)
Construct $\mathcal{D}_{\text{distill}}$ from benchmark videos and generated rollouts scored by teacher f_ϕ ▷ Teacher-labeled mixed corpus
Train g_ψ with per-dimension weighted Huber regression on normalized teacher scores (Eq. 4)

— Stage 4: GRPO Post-Training with Online Iterative Distillation —
for iteration $n = 1, 2, \ldots$ **do**
  Sample group of G rollouts $\{\hat{x}^{(j)}\}_{j=1}^{G} \sim p_\theta$
  Score each rollout: $R^{(j)} = \sum_{k=1}^{6} w_k [g_\psi(l, \hat{x}^{(j)})]_k$
  Compute advantage: $A^{(j)} = (R^{(j)} - \mathrm{mean}_j R)/\mathrm{std}_j R$
  Update θ via clipped policy gradient $\mathcal{L}_{\text{GRPO}}$ (Eq. 7)
  if $n \bmod K = 0$ **then**
    Score fresh rollouts with teacher f_ϕ and update student g_ψ ▷ Online iterative distillation
  end if
end for
return p_{θ^*}

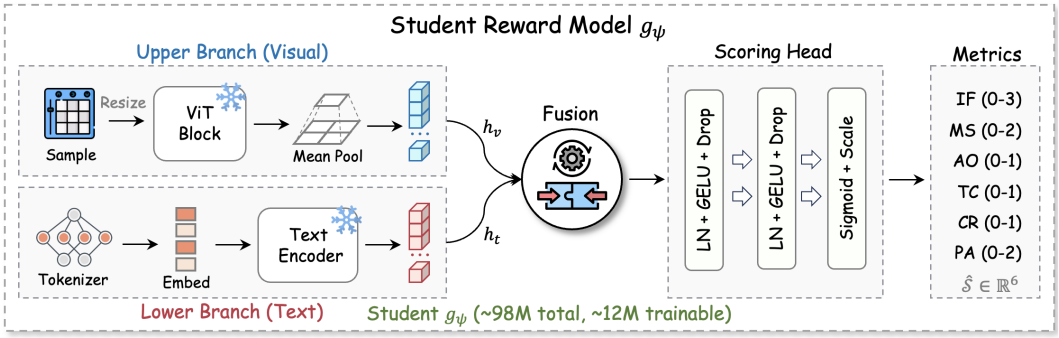


Figure 13: Architecture of the distilled student reward model g_ψ. A visual branch encodes uniformly sampled video frames using a partially unfrozen ViT-Base/16 backbone, while a text branch encodes the instruction using a lightweight Transformer. The fused multimodal representation is then mapped to six normalized reward dimensions by a compact MLP head.

E Student Reward Model Details

Figure 13 presents the architecture of the distilled student reward model g_ψ. The student is designed to provide efficient reward estimation within the inner RL loop by approximating the six-dimensional score vector produced by the teacher judge f_ϕ. In our implementation, the model contains approximately 98M parameters in total, of which about 40.6M are trainable under partial ViT unfreezing. Its inference latency is approximately 20 ms per video (roughly 50 videos/s), making it substantially more efficient than direct teacher inference.

Visual branch. Given a generated video v consisting of T frames, we uniformly sample $N{=}8$ key frames. During training, we introduce temporal jitter by perturbing the uniform frame indices with small random offsets, thereby improving robustness to minor timing variations. The sampled frames are further processed with video-level data augmentation, including a shared random resized crop, a shared horizontal flip, and mild per-frame color jitter. During validation and testing, all frames are deterministically resized to 224×224.

The augmented frames are encoded independently by a pretrained ViT-Base/16 visual encoder. Rather than freezing the backbone entirely, we freeze the first 8 Transformer blocks and unfreeze the final 4 blocks together with the final normalization layer. This design allows high-level visual features to adapt to failure modes specific to robot-manipulation videos while preserving generic low-level image representations. For each frame, we extract the [CLS] feature $\mathbf{h}_i \in \mathbb{R}^{768}$ and perform temporal mean pooling to obtain a video-level representation,

$$\mathbf{v}_{\text{pool}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{h}_i.$$

This representation is then passed through a lightweight projection layer (LayerNorm $\rightarrow$ Linear $\rightarrow$ GELU) to produce the final visual embedding $\mathbf{v} \in \mathbb{R}^{768}$.

Text branch. The instruction l is tokenized using the BERT tokenizer from `bert-base-uncased` and truncated or padded to a maximum length of $L{=}64$. The resulting subword sequence is encoded by a lightweight 4-layer Transformer encoder with hidden size 256, 4 attention heads, and feed-forward dimension 1024. A learnable [CLS] token is prepended to the input sequence and combined with learnable positional embeddings. After passing through the 4 Transformer layers, the output corresponding to the [CLS] token is used as the instruction embedding $\mathbf{t} \in \mathbb{R}^{256}$. Each layer adopts a Pre-Norm Transformer block with GELU activations and dropout at rate 0.2.

Multimodal fusion and scoring head. We fuse the visual and textual modalities by simple concatenation,

$$\mathbf{z} = [\mathbf{v}; \mathbf{t}] \in \mathbb{R}^{1024}.$$

The fused representation is processed by a 3-layer MLP scoring head with the following structure: Linear$(1024 \rightarrow 512) \rightarrow$ LayerNorm $\rightarrow$ GELU $\rightarrow$ Dropout, followed by Linear$(512 \rightarrow 256) \rightarrow$

Table 20: **Backbone and architecture ablation of the student reward model.** Comparison of representative visual encoders and fusion designs under matched training and evaluation settings. Metrics capture both fidelity to teacher judgments and inference efficiency.

Method	Visual encoder	Fusion module	Pearson corr. ↑	MSE ↓	Latency (ms) ↓
ResNet student	ResNet-50	concat + MLP	0.79	0.186	13
Frozen ViT student	ViT-Base/16 (frozen)	concat + MLP	0.84	0.142	18
Text-only student	none	Transformer + MLP	0.52	0.421	4
Ours	ViT-Base/16 (last 4 blocks unfrozen)	concat + MLP	0.88	0.096	20

LayerNorm $\rightarrow$ GELU $\rightarrow$ Dropout, and a final Linear$(256 \rightarrow 6)$ layer. A sigmoid activation maps the output to normalized reward scores in $[0,1]^6$, corresponding to instruction following, manipulation success, action–outcome consistency, temporal consistency, contact realism, and physics adherence. During distillation training and during GRPO post-training (Eq. 5), the student is always consumed in this normalized $[0,1]^6$ form; the raw-range rescaling to $[3,2,1,1,1,2]$ is applied only when student outputs need to be reported alongside teacher scores for diagnostic or visualization purposes, and is never re-applied before the reward aggregation $R(\hat{x}_{1:T})$.

Distillation training. We train the student on teacher-labeled video–instruction pairs using normalized teacher scores, which prevents dimensions with larger numerical ranges from dominating optimization. Instead of plain MSE, we adopt a per-dimension weighted Huber loss:

$$\mathcal{L}_{\text{distill}} = \frac{1}{B}\sum_{b=1}^{B}\sum_{k=1}^{6}\lambda_k \,\text{Huber}_{\delta_h}\big(\hat{s}_{b,k}, \tilde{s}_{b,k}\big), \tag{16}$$

where $\hat{s}_{b,k} = [g_\psi(l_b, v_b)]_k \in [0,1]$ is the student's sigmoid-activated output, $\tilde{s}_{b,k}$ is the teacher score normalized dimension-wise to $[0,1]$, and $\delta_h = 0.5$ is the Huber threshold. Since both sides already share the same $[0,1]$ scale after normalization, the per-dimension weights $\{\lambda_k\}$ are used only to equalize convergence speed across dimensions with unequal label-noise levels (e.g., 3-level rubric vs. binary rubric); in practice we tune them on a validation split rather than tying them to the raw score ranges. Note that $\{\lambda_k\}$ are distinct from the reward aggregation weights $\{w_k\}$ used in Eq. 5.

Training further employs mixup regularization with probability 0.3, AdamW with weight decay 0.05, and differential learning rates: 5×10^{-6} for the unfrozen ViT blocks and 5×10^{-4} for the text encoder and scoring head. The learning-rate schedule uses linear warmup over the first 10% of training steps followed by cosine decay, with a minimum learning rate of 10^{-6}. We additionally apply gradient clipping with maximum norm 1.0, maintain an exponential moving average (EMA) of the trainable parameters with decay 0.999, and select checkpoints based on validation-set Pearson correlation with early stopping (patience 10). Collectively, these design choices improve fidelity to the teacher's ranking behavior and stabilize the student when used as an online reward proxy during RL post-training.

Backbone and architecture comparison. Because the student reward model is intended to serve as a high-throughput reward proxy rather than the primary source of semantic supervision, an important design question is whether a simple CNN-style reward model is sufficient, or whether a partially unfrozen ViT-based architecture is necessary. Table 20 compares representative lightweight baselines with the final partially unfrozen ViT student used in our method. The results indicate that the final student achieves a more favorable trade-off between fidelity to the teacher and online efficiency than simpler reward regressors.

Teacher–student efficiency trade-off. The teacher and student play complementary roles in our framework. The high-capacity teacher remains essential for providing semantically rich, high-fidelity supervision during offline reward distillation. The student is introduced solely to make such supervision practical within the inner RL loop, where thousands of rollout evaluations may be required. Table 21 summarizes the resulting efficiency gap.

This efficiency gap is particularly important for RL post-training. For example, if a single training step requires scoring 16 generated videos, direct teacher evaluation would still take on the order of seconds even with moderate parallelism, whereas the student can score the same batch in a fraction of

Table 21: **Teacher–student efficiency comparison for reward evaluation.** The teacher provides high-fidelity supervisory signals, whereas the distilled student is optimized for fast online reward evaluation during RL post-training. All numbers are measured or estimated on a single A100 40GB GPU.

Statistic	Teacher judge (f_ϕ)	Student reward model (g_ψ)
Architecture	Qwen3-VL-8B-Thinking + LoRA	ViT-Base/16 + 4L text Transformer + MLP
Total parameters	$\sim$8B	$\sim$98M
Trainable parameters	LoRA-only adaptation	$\sim$40.6M
Per-video latency	$\sim$2800 ms	$\sim$20 ms
Inference throughput	$\sim$0.5 videos/s	$\sim$50 videos/s
Inference GPU memory	$\sim$16–20 GB	$\sim$0.5–1 GB
Input representation	video frames + judge prompt	8 sampled frames + instruction
Output representation	autoregressive text + parsed scores	direct 6D score vector

Table 22: **Reward-evaluation cost within the RL loop.** Comparison of teacher-only evaluation and distilled-student evaluation for representative RL workloads.

RL workload	Teacher judge (f_ϕ)	Student reward model (g_ψ)
Reward evaluation for 16 videos / step	$\sim$45 s (serial) / $\sim$12 s (4-way parallel)	$\sim$0.32 s (batched)
Reward cost over 1000 RL steps	$\sim$12,000–45,000 s	$\sim$320 s
Share of total training time	dominant bottleneck	minor overhead
Additional hardware demand	dedicated large-VLM inference budget	can share training GPU

a second. In practice, this shifts reward computation from a dominant bottleneck to a comparatively minor overhead, while preserving the teacher as the source of high-quality supervision.

F Pseudocode and Implementation of Sliding Window Re-encoding

We present the sliding-window re-encoding inference procedure in two complementary formats: Algorithm 2 provides a formal pseudocode overview, and Listing 1 shows the corresponding PyTorch implementation extracted from our codebase. Both use matching color coding for the three key stages: **prompt construction** (blue), **autoregressive generation** (green), and **context refresh** (red).

Algorithm 2 Sliding Window Re-encoding Inference

Require: World model p_θ; tokenizer $\mathcal{T}_{\text{vis}}$; decoder $\mathcal{D}_{\text{vis}}$; conditioning frame x_0; actions $\{a_t\}_{t=1}^{T}$; window size W
Ensure: Generated video $\hat{x}_{1:T}$

$z_1^{\text{ctx}}, z_0^{\text{dyn}} \leftarrow \mathcal{T}_{\text{vis}}(x_0)$ ▷ Encode initial context (x_0 used as both E_c and length-1 E_d input)
frames_out $\leftarrow [\,]$; $m \leftarrow 1$
while $|\text{frames_out}| < T$ **do**
  $W' \leftarrow \min(W,\ T - |\text{frames_out}|)$
  prompt $\leftarrow [\, z_m^{\text{ctx}} \parallel z_0^{\text{dyn}} \parallel \hat{a}_1 \,]$ ▷ Build prompt
  window_tokens $\leftarrow [\,]$
  for $j = 1, \ldots, W'$ **do**
    $\hat{z}_j^{\text{dyn}} \leftarrow p_\theta(\cdot \mid \text{prompt})$ ▷ Generate N_d=80 dynamics tokens
    Append $\hat{z}_j^{\text{dyn}}$ to window_tokens
    prompt $\leftarrow$ prompt $\parallel \hat{z}_j^{\text{dyn}} \parallel \hat{a}_{j+1}$ ▷ Extend prompt
  end for
  $\hat{x}_{\text{win}} \leftarrow \mathcal{D}_{\text{vis}}(z_m^{\text{ctx}},\ \text{window_tokens})$ ▷ Decode window to pixels
  Append $\hat{x}_{\text{win}}$ to frames_out
  if $|\text{frames_out}| < T$ **then**
    $\hat{x}_{\text{last}} \leftarrow \hat{x}_{\text{win}}[-1]$ ▷ Take last decoded frame
    $z_{m+1}^{\text{ctx}}, z_0^{\text{dyn}} \leftarrow \mathcal{T}_{\text{vis}}(\hat{x}_{\text{last}})$ ▷ Re-encode last frame via E_c and length-1 E_d
  end if
  $m \leftarrow m + 1$
end while
return frames_out

Listing 1: **PyTorch implementation** of sliding window re-encoding. Comments are color-coded to match Algorithm 2. @`ctx_tokens`@ corresponds to @z_m^{ctx}@ and @`dyn_tokens`@ to @z_0^{dyn}@ in Algorithm 2.

```python
def generate_sliding_window(
    model, tokenizer, ctx_tokens, dyn_tokens,
    actions, num_frames, window_size=6,
):
    all_frames = []
    frames_done, step = 0, 0

    while frames_done < num_frames:
        W = min(window_size, num_frames - frames_done)

        # [BLUE] Build initial prompt:  [ctx | dyn_0 | action_0]
        prompt = build_prompt(ctx_tokens, dyn_tokens, actions[step])
        window_dyn = []

        # [GREEN] Autoregressive generation within one window
        for j in range(W):
            gen_tokens = model.generate(prompt, max_tokens=80)
            window_dyn.append(gen_tokens)
            prompt = prompt + gen_tokens + discretize(actions[step+j+1])

        # [GREEN] Decode entire window to pixel space
        decoded = tokenizer.decode(ctx_tokens, window_dyn)
        all_frames.extend(decoded)
        frames_done += W
        step += W

        # [RED] Context refresh:  decode-re-encode cycle
        # last_frame fed to both E_c and length-1 E_d
```

```
        if frames_done < num_frames:
            last_frame = decoded[-1]
            ctx_tokens, dyn_tokens = tokenizer.encode(
                last_frame, last_frame
            )

    return all_frames
```

Table 23: SWR efficiency vs. default AR across W: timing, throughput, GPU memory, prompt length (tokens), mean per-frame latency, and re-encoding.

Metric	AR (∞)	W=4	W=6	W=8	W=10	W=15
Total time (s)	$5.646_{\pm 0.01}$	$5.863_{\pm 0.02}$	$5.709_{\pm 0.07}$	$5.858_{\pm 0.28}$	$5.740_{\pm 0.11}$	$5.732_{\pm 0.07}$
FPS	5.31	5.12	5.26	5.13	5.23	5.23
Peak GPU (MB)	34,206	32,066	32,781	33,493	34,214	34,209
Mean prompt len.	2,722	1,506	1,606	1,680	1,792	2,024
Max prompt len.	4,070	1,652	1,838	2,024	2,210	2,675
Mean ms/frame	182.2	174.7	175.7	181.5	180.7	181.7
#Re-encodings	0	7	4	3	2	1
Re-enc. total (ms)	0	99.8	73.8	68.3	48.3	87.0

Table 24: Re-encoding overhead vs. W. % total: re-encoding time / wall time. Per step: mean ms per refresh.

W	#Re	Total (ms)	% total	Per step (ms)
4	7	99.8	1.70%	14.3
6	4	73.8	1.29%	18.5
8	3	68.3	1.17%	22.8
10	2	48.3	0.84%	24.1
15	1	87.0	1.52%	87.0

G SWR window-size ablation

This section complements the main-text SWR panel (Table 3). Unless noted, measurements use Llama-style causal LM (12 layers, hidden 768, 12 heads), vLLM inference, and 300 episodes $\times 30$ frames.

Table 25: Long-horizon prompt scaling: AR max prompt extrapolates linearly with rollout length; SWR (W=6) caps at the 30-frame plateau. Slow-down columns are indicative.

Frames	AR max prompt	SWR W=6 max	Prompt drop	AR slow-down	SWR slow-down
30	4,070	1,838	54.8%	∼7%	∼0%
100	∼12,673	1,838	85.5%	∼25–30%	∼0%
300	∼37,273	1,838	95.1%	OOM risk	∼0%

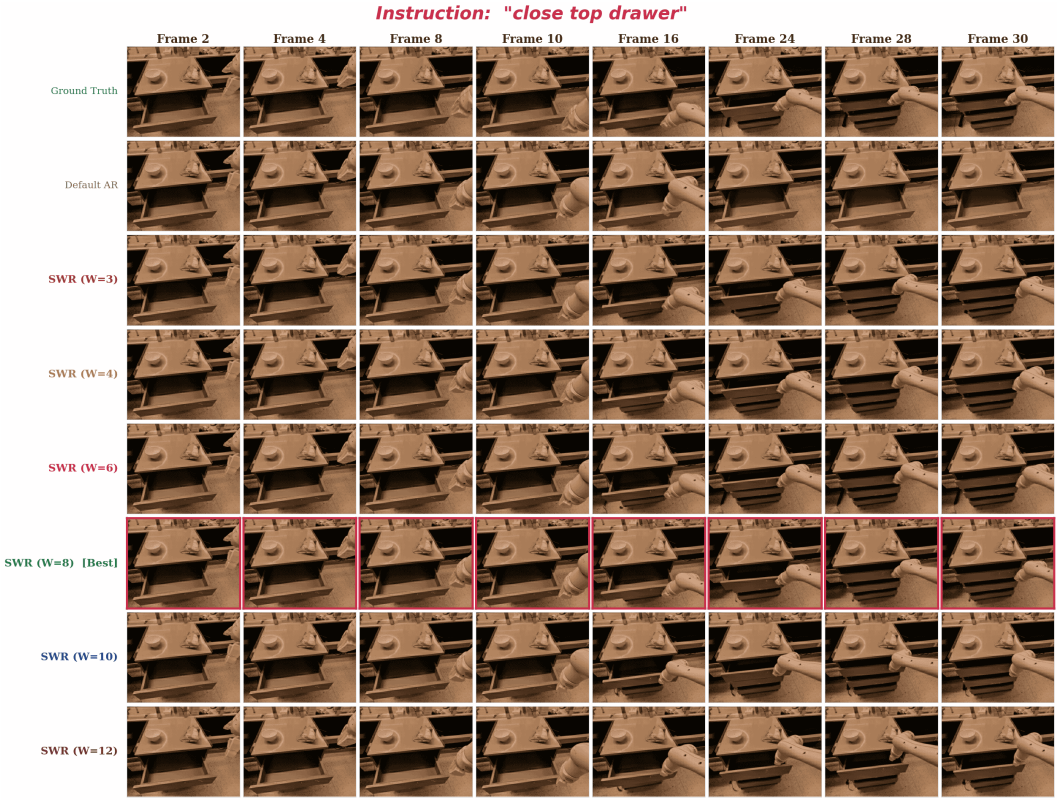


Figure 14: Qualitative comparison for the instruction *"close top drawer"*. The highlighted row denotes the visually best SWR result for this episode.

H SWR qualitative visualizations

This section provides qualitative case studies comparing default autoregressive decoding with SWR under different refresh windows. In each figure, columns show sampled frames along the rollout, while rows compare Ground Truth, Default AR, and SWR with different window sizes W. The highlighted row marks the visually best SWR variant for that episode. Consistent with the quantitative results, periodic refresh generally reduces long-horizon drift and better preserves task progression and object configuration, although the best per-episode W can vary with the manipulation trajectory.

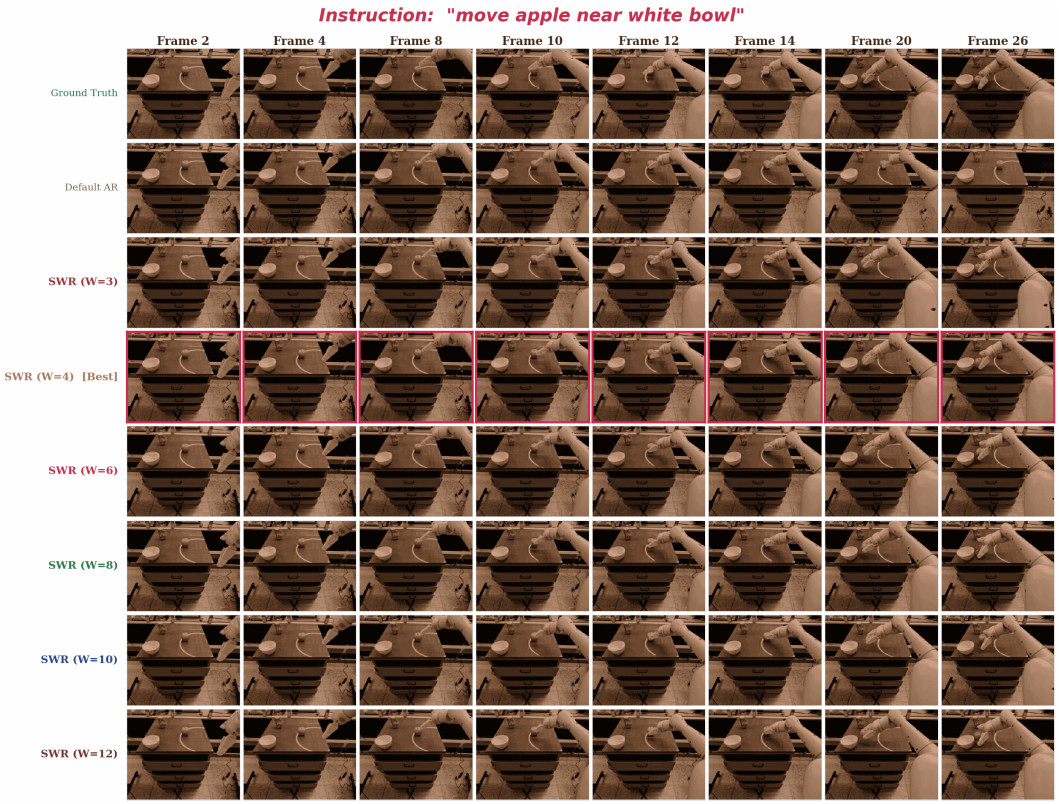


Figure 15: Qualitative comparison for the instruction *"move apple near white bowl"*. The highlighted row denotes the visually best SWR result for this episode.

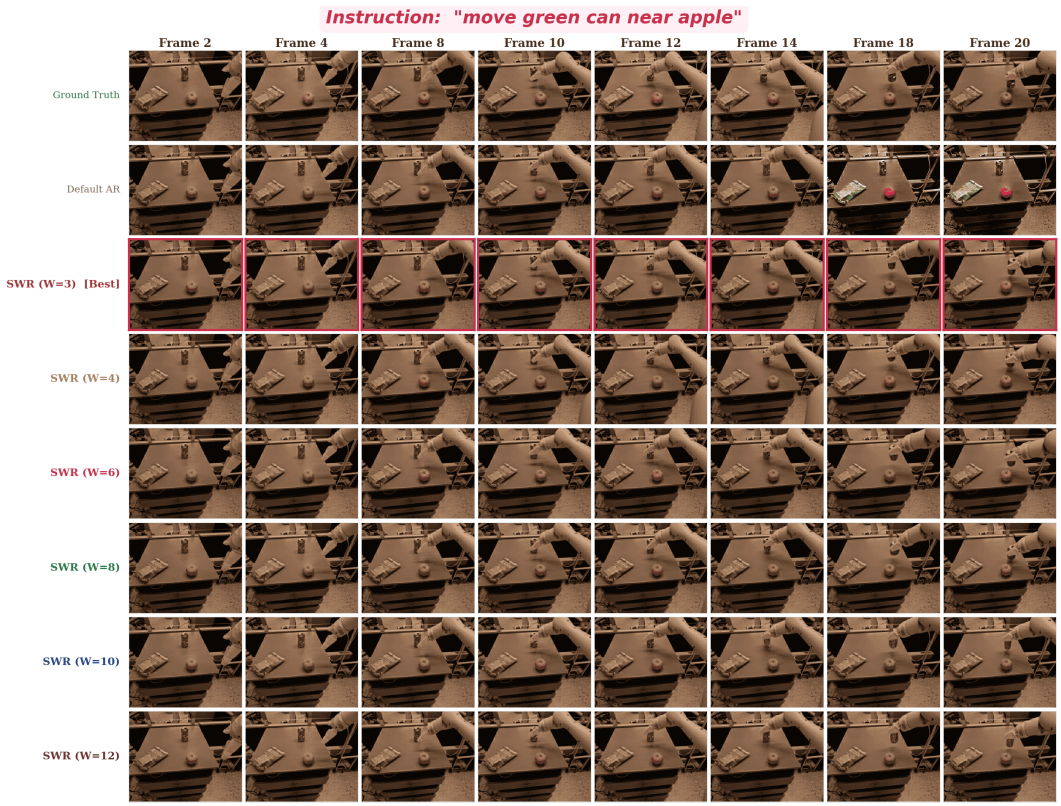


Figure 16: Qualitative comparison for the instruction *"move green can near apple"*. The highlighted row denotes the visually best SWR result for this episode.

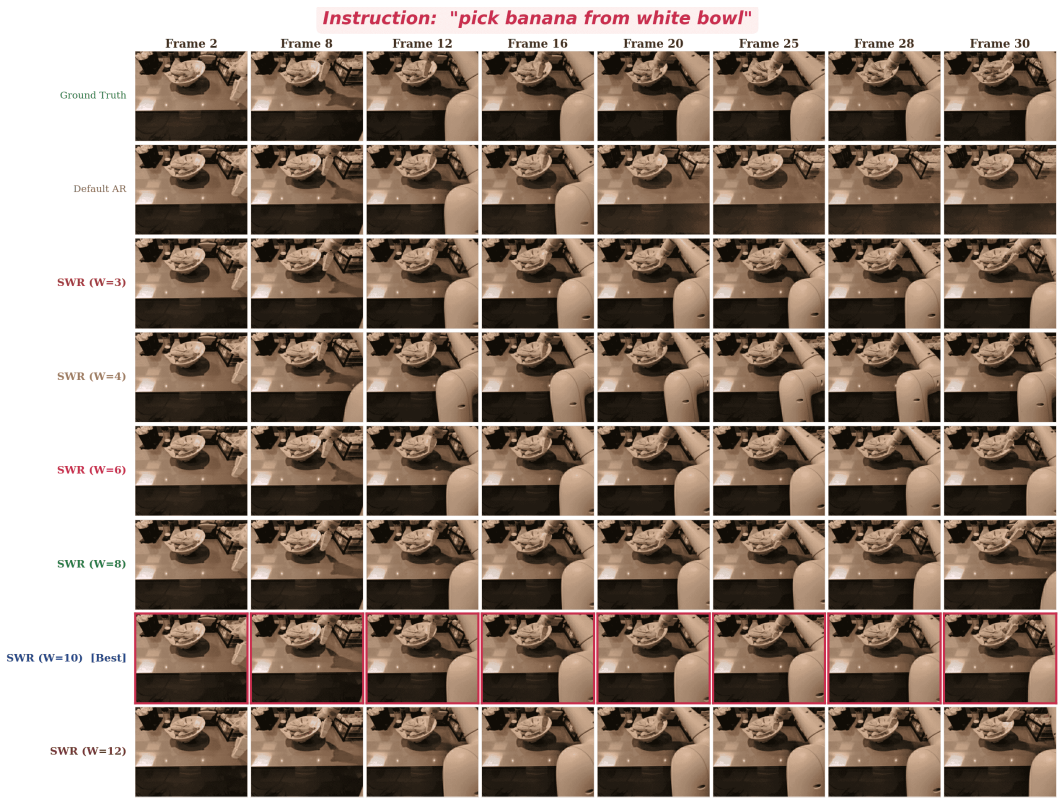


Figure 17: Qualitative comparison for the instruction *"pick banana from white bowl"*. The highlighted row denotes the visually best SWR result for this episode.

I Evaluation Metrics

This section provides a detailed description of all evaluation metrics used in our experiments.

I.1 Semantic and Physical Alignment Metrics

The ROBOALIGN-JUDGE evaluates generated videos along six fine-grained dimensions, each scored on a bounded scale:

- **Instruction Following (IF)** $\in [0, 3]$: Whether the generated video faithfully executes the language instruction.
- **Manipulation Success (MS)** $\in [0, 2]$: Whether the robot successfully completes the target manipulation task.
- **Action–Outcome Consistency (AO)** $\in [0, 1]$: Whether the observed outcome is consistent with the executed action.
- **Temporal Consistency (TC)** $\in [0, 1]$: Whether the video maintains smooth and temporally consistent transitions.
- **Contact Realism (CR)** $\in [0, 1]$: Whether contact events between the robot and objects appear physically plausible.
- **Physics Adherence (PA)** $\in [0, 2]$: Whether the video obeys basic physical laws (gravity, rigidity, collision response).

The aggregate score is the sum of the raw six-dimensional scores, yielding a maximum of 10.0. When these scores are used for student distillation, each dimension is normalized independently to $[0, 1]$ before regression.

I.2 Pixel-Level Reconstruction Metrics

We report four standard metrics computed over the full image:

- **MSE**: Mean Squared Error between generated and ground-truth frames (pixel values normalized to $[0, 1]$).
- **PSNR**: Peak Signal-to-Noise Ratio, defined as $\mathrm{PSNR} = 10 \log_{10}(1/\mathrm{MSE})$.
- **SSIM** [22]: Structural Similarity Index, measuring luminance, contrast, and structural similarity.
- **LPIPS** [21]: Learned Perceptual Image Patch Similarity, measuring perceptual distance using deep features.

I.3 Motion-Mask-Based ROI Metrics

Motivation. In robot manipulation videos, the background typically remains static while only the robot arm and manipulated objects undergo significant motion. As a result, global pixel-level metrics (e.g., PSNR, SSIM) are dominated by the large static background, which is trivially easy to reconstruct. This dilutes the evaluation signal from the critical dynamic interaction regions. To address this, we introduce **Region-of-Interest (ROI) metrics** that focus exclusively on the dynamic foreground.

Motion mask extraction. We derive the ROI from ground-truth video sequences using temporal frame differencing. Specifically, for each frame x_t, we compute the average absolute pixel difference against neighboring frames within a temporal window of size τ:

$$D_t(i, j) = \frac{1}{|\mathcal{N}_t|} \sum_{t' \in \mathcal{N}_t} \frac{1}{3} \sum_{c=1}^{3} |x_t^c(i, j) - x_{t'}^c(i, j)|, \tag{17}$$

where $\mathcal{N}_t = \{t' : 0 < |t' - t| \leq \tau,\ 1 \leq t' \leq T\}$ and c indexes the RGB channels. The binary motion mask is obtained by thresholding:

$$M_t(i, j) = \mathbf{1}[D_t(i, j) > \theta], \tag{18}$$

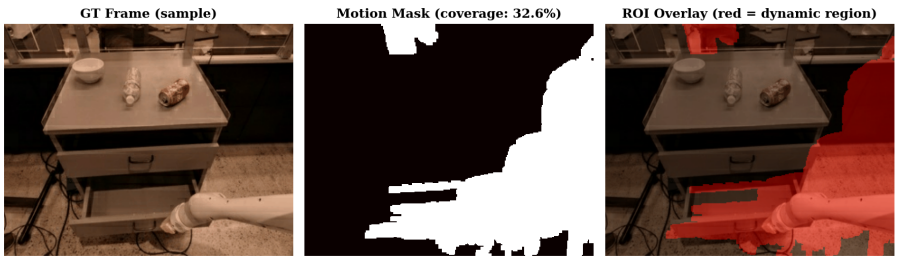


Figure 18: **Motion mask visualization.** The extracted motion mask (highlighted) isolates the dynamic interaction region—the robot arm and manipulated object—from the static background. ROI metrics are computed exclusively within this masked region, providing a more faithful assessment of generation quality for task-critical areas.

followed by morphological closing and dilation (elliptical kernel, size k) to fill small gaps and include surrounding context. In our experiments, we use τ=3, θ=15, and k=15. The **union mask** $M_{\cup} = \max_t M_t$ aggregates all per-frame masks into a single ROI that covers the entire dynamic region across the episode.

Illustration. Figure 18 visualizes the motion mask extraction process on a representative RT-1 episode. The left panel shows the original ground-truth frame; the right panel overlays the extracted motion mask (highlighted region) on the frame. The mask accurately captures the robot arm and the manipulated object while excluding the static background, confirming that ROI metrics focus evaluation on the most task-relevant regions.

ROI metric definitions. Given a ground-truth frame x_t, a generated frame $\hat{x}_t$, and the binary mask M_t (or $M_{\cup}$), the ROI metrics are defined as:

- **ROI-MSE**: $\text{ROI-MSE} = \dfrac{\sum_{i,j} M(i,j) \cdot \|x_t(i,j) - \hat{x}_t(i,j)\|^2}{3 \cdot \sum_{i,j} M(i,j)}$
- **ROI-PSNR**: $\text{ROI-PSNR} = 10 \log_{10}\left(\dfrac{1}{\text{ROI-MSE}}\right)$
- **ROI-SSIM**: Local SSIM map computed over the full image, then averaged only within the masked region.
- **ROI-LPIPS**: LPIPS computed by masking out the static background (setting it to zero in both ground-truth and generated frames) before passing through the perceptual network.

ROI coverage. We additionally report **ROI coverage**, defined as the fraction of pixels in the union mask: $|M_{\cup}|/(H \times W)$. This provides context for interpreting ROI metrics—a typical RT-1 episode has ROI coverage of approximately 30–35%, confirming that the majority of the image is static background.

J ROBOALIGN-JUDGE: Multimodal Teacher Judge Training Details

This appendix provides a comprehensive account of the training pipeline, data construction, and evaluation of ROBOALIGN-JUDGE—the multimodal teacher judge at the core of ROBOALIGN-R1's reward-aligned post-training framework (§3.2). ROBOALIGN-JUDGE is instantiated by fine-tuning Qwen3-VL-8B-Thinking [36] with LoRA [42], producing structured six-dimensional quality scores together with chain-of-thought reasoning for robot manipulation videos.

J.1 Motivation and Design Rationale

As discussed in §3.2, standard low-level metrics (MSE, LPIPS, SSIM) are poorly aligned with the properties that matter for robot world models: instruction correctness, physical plausibility, and action–outcome consistency. A multimodal judge that can assess these high-level properties provides much richer supervision for RL post-training. We choose Qwen3-VL-8B-Thinking as the teacher backbone for three reasons:

(i) **Native video understanding.** Qwen3-VL supports multi-frame visual inputs with dynamic resolution, enabling direct processing of robot manipulation video sequences without frame-level feature extraction.

(ii) **Built-in reasoning.** The "Thinking" variant provides structured chain-of-thought reasoning via `<think>` tokens, which naturally aligns with our requirement for interpretable, dimension-wise scoring with explicit justification.

(iii) **Parameter efficiency.** At 8B parameters, the model is large enough to capture nuanced physical and semantic judgments, yet small enough for efficient LoRA fine-tuning and batch inference during teacher labeling.

J.2 Training Data Construction

The training data for ROBOALIGN-JUDGE forms the annotation component of ROBOTWORLDBENCH (Eq. 2), which spans four robot datasets and 10,000 annotated video–instruction pairs in total. In this appendix, we summarize two representative construction pipelines used in the corpus and report the cross-dataset composition used in the current study.

Part A: Synthetic degradation with rule-based annotation. As one representative source within the broader corpus, we extract 500 ground-truth (GT) episodes from the RT-1 dataset [38], each containing a robot manipulation video and its corresponding language instruction. For each GT video, we generate two degraded variants by randomly sampling 1–3 degradation types from a pool of 10 operations spanning four categories:

- **Temporal** (3 types): frame shuffling, frame dropping, segment reversal—simulating temporal incoherence and causal violations.
- **Visual** (4 types): Gaussian blur, Gaussian noise, color jitter, resolution degradation—simulating perceptual quality loss and compression artifacts.
- **Spatial** (1 type): random per-frame spatial shifts—simulating camera instability and spatial inconsistency.
- **Semantic** (2 types): video truncation, end-frame freezing—simulating task incompleteness and action termination failures.

Each degradation is applied at a severity level sampled from $\{\text{mild}, \text{moderate}, \text{severe}\}$ with probabilities $\{0.2, 0.4, 0.4\}$, biasing toward harder examples. Scores are computed deterministically: each degradation type has predefined per-dimension deduction fractions that scale with severity, ensuring consistent and reproducible annotations. This yields $\sim$994 augmented videos. An additional 100 GT videos are retained as perfect-score references (total score $= 10$), providing the upper anchor for the score distribution. Figure 19 shows representative examples from this synthetic-degradation pipeline, illustrating how the rule-based transformations create controlled failures across multiple error types and severity levels.

Part B: Real I2V model outputs with human annotation. Synthetic degradation cannot fully capture the failure modes of real generative models—*e.g.*, hallucinated objects, semantic drift, physically implausible contact dynamics, or mode collapse. To address this gap, we collect generated

videos from 10 diverse image-to-video (I2V) models evaluated on RT-1 prompts, spanning both general-purpose and robotics-specific architectures (Table 26). Each video is scored by trained human annotators following the identical six-dimension rubric used in Part A.

Table 26: I2V models used for collecting human-annotated training data in ROBOTWORLDBENCH. Models span diffusion-based, transformer-based, and autoregressive architectures across general and robotics domains.

Model	**Architecture**	**Domain**	**Key Characteristic**
CogVideoX	Diffusion-based I2V	General	3D causal VAE + DiT
HunyuanVideo	Diffusion-based I2V	General	Dual-stream DiT
I2VGen-XL	Cascaded diffusion	General	Two-stage generation
LTX-Video	Latent Transformer	General	Efficient latent space
SVD	Diffusion-based I2V	General	Temporal layer fine-tuning
OpenSora	DiT-based I2V	General	Open-source Sora replica
DynamiCrafter	Diffusion-based I2V	General	Image-conditioned dynamics
iVideoGPT	Autoregressive GPT	Robotics	Token-based world model
RoboDreamer	Goal-conditioned diffusion	Robotics	Compositional planning
Vid2World	Diffusion world model	Robotics	Interactive simulation

Data complementarity and balancing. The two sources serve complementary roles: Part A provides controllable coverage across score ranges with deterministic labels, while Part B introduces authentic generative artifacts that improve robustness to real-world evaluation scenarios. Beyond these RT-1-centered examples, the full ROBOTWORLDBENCH training corpus used in the current study spans four robot datasets and contains 10,000 annotated video–instruction pairs for judge training and reward distillation. After score-bin balancing and corpus aggregation, the resulting training set maintains broad coverage over the $[0, 10]$ score range, which is important for preventing the judge from developing score-range biases that would propagate through reward distillation into RL post-training.

Table 27: ROBOTWORLDBENCH data statistics.

Dataset	GT	Synthetic	Generated	Total Pairs
RT-1	150	450	600	1,200
BridgeData V2	450	1,050	1,500	3,000
CALVIN	300	700	1,000	2,000
LIBERO	550	1,350	1,900	3,800
Total	**1,450**	**3,550**	**5,000**	**10,000**



Figure 19: Representative examples from Part A synthetic degradation with rule-based annotation. Starting from a ground-truth robot manipulation video, we apply controlled temporal, visual, spatial, and semantic perturbations at different severity levels to generate training examples with reproducible score deductions.

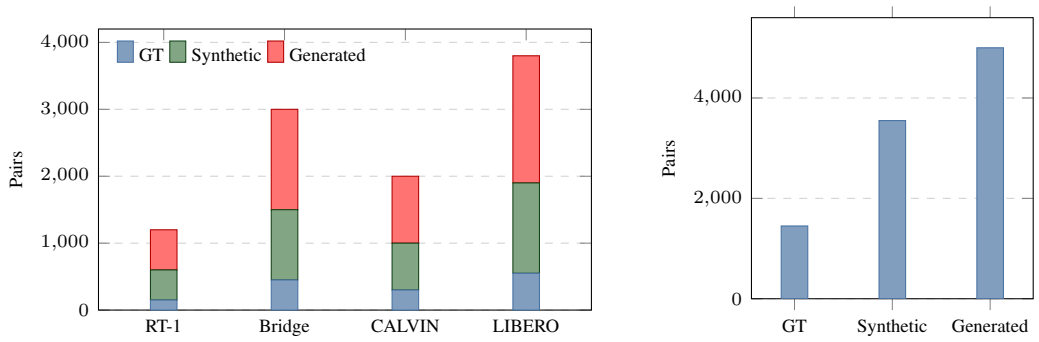


Figure 20: **Corpus composition of RoboTWorldBench. Left:** per-dataset breakdown by source type. **Right:** aggregate source composition across the full 10,000-pair annotation corpus. Generated samples form the largest component, complemented by rule-based synthetic degradations and a smaller set of ground-truth reference videos.

J.3 Six-Dimension Scoring Rubric

ROBOALIGN-JUDGE evaluates each video along six complementary dimensions, grouped into *task alignment* and *physical realism* categories (Table 28). The rubric is designed to capture the properties most relevant to robot world-model utility: whether the predicted future correctly reflects the intended manipulation, and whether the physical dynamics are plausible enough to support downstream planning.

Table 28: Six-dimension scoring rubric for ROBOALIGN-JUDGE. Dimensions are grouped by task alignment (semantic correctness) and physical realism (dynamics plausibility).

Category	Dimension	Range	Scoring Criteria
	Instruction Following (IF)	0–3	0: completely unrelated action; 1: vaguely related but wrong; 2: correct action but incomplete/imprecise; 3: perfectly follows instruction
	Manipulation Success (MS)	0–2	0: task completely failed; 1: partial success (object moved but not to target); 2: full success
	Action-Outcome Consist. (AO)	0–1	0: actions and outcomes are inconsistent; 1: logically consistent
	Temporal Consistency (TC)	0–1	0: severe temporal artifacts (flickering, jumps); 1: smooth and temporally coherent
	Contact Realism (CR)	0–1	0: unrealistic contacts (penetration, floating); 1: natural and physically realistic
	Physics Adherence (PA)	0–2	0: severe physics violations; 1: minor issues but mostly plausible; 2: fully physically plausible
	Total	**0–10**	Sum of all six dimensions

J.4 Model Architecture and LoRA Fine-Tuning

Architecture. ROBOALIGN-JUDGE is built on Qwen3-VL-8B-Thinking, a multimodal large language model with native multi-frame video understanding. We apply Low-Rank Adaptation (LoRA) to all linear projection layers in both the attention blocks ($\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o$) and the MLP blocks ($\mathbf{W}_{\text{gate}}, \mathbf{W}_{\text{up}}, \mathbf{W}_{\text{down}}$), totaling 7 target modules per Transformer layer. This broad LoRA coverage ensures that both the visual–language alignment and the reasoning capacity are adapted to the robot manipulation evaluation domain. Table 29 summarizes the complete training configuration.

Table 29: Training configuration for ROBOALIGN-JUDGE LoRA fine-tuning.

Hyperparameter	Value
Base Model	Qwen3-VL-8B-Thinking
Trainable Parameters	∼160M (LoRA only) / 8.3B total
LoRA Rank (r)	64
LoRA Scaling (α)	128
LoRA Dropout	0.05
Target Modules	`q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj`
Training Epochs	5
Optimizer	AdamW (β_1=0.9, β_2=0.999, ϵ=10^{-8})
Learning Rate	2×10^{-4}
LR Schedule	Cosine decay with linear warmup
Warmup Ratio	0.1
Effective Batch Size	32 (1×4 gradient accumulation $\times$ 8 GPUs)
Max Sequence Length	4,096 tokens
Precision	BFloat16 (mixed precision)
Gradient Checkpointing	✓
Hardware	8× NVIDIA A100 40GB
Training Time	∼31 minutes

Input format. Each training sample is structured as a three-turn conversation following the Qwen3-VL chat template:

(1) **System prompt**: defines the evaluator role, the six scoring dimensions with detailed rubrics, and the required JSON output schema (see Appendix J.11 for the full prompt).
(2) **User message**: contains the language instruction l, the initial frame x_0 (as an image), and $N{=}8$ frames uniformly sampled from the generated video v.
(3) **Assistant response**: begins with `<think>...</think>` chain-of-thought reasoning that analyzes each dimension step by step, followed by a structured JSON object containing the six dimension scores and total.

The training loss $\mathcal{L}_{\text{teacher}}$ (Eq. 3) is computed only on the assistant response tokens, ensuring that the model learns to produce both the reasoning trace and the structured scores while treating the system and user turns as context.

Training dynamics. The training loss drops rapidly in the first epoch (from 0.762 to 0.066), indicating fast adaptation of the LoRA parameters to the evaluation task. Convergence is reached by epoch 3–4, with the final loss stabilizing at $\sim$0.016 by epoch 5. The smooth convergence without loss spikes or oscillation confirms that the LoRA rank ($r{=}64$) provides sufficient capacity for this task, and that the learning rate schedule is well-calibrated. Table 30 reports the loss trajectory at key checkpoints, and Figure 21 visualizes the full training curve.

Table 30: Training loss trajectory for RoboAlign-Judge. The model converges within 3 epochs and stabilizes by epoch 5.

Epoch	0.27	0.53	0.80	1.05	2.11	3.16	4.21	5.00
Loss	0.762	0.209	0.066	0.036	0.025	0.021	0.018	0.016

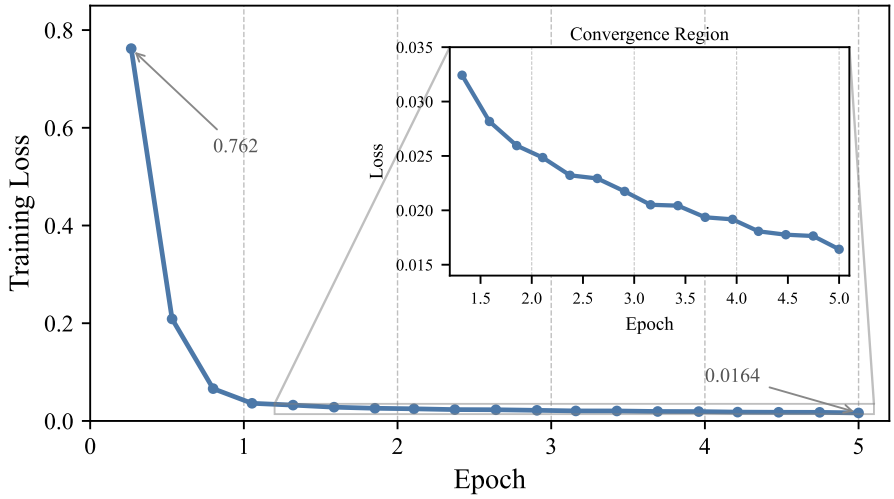


Figure 21: Training loss curve for RoboAlign-Judge LoRA fine-tuning over 5 epochs. The inset shows the convergence region (epochs 1–5). Loss drops rapidly in the first epoch and stabilizes around 0.016.

J.5 Chain-of-Thought Reasoning

A distinctive feature of RoboAlign-Judge is its use of explicit chain-of-thought (CoT) reasoning before producing scores. By leveraging the "Thinking" capability of Qwen3-VL, the judge first generates a detailed analysis within `<think>...</think>` tags, examining each evaluation dimension in sequence:

- Whether the robot's motion trajectory matches the instructed action;
- Whether the manipulation target is correctly identified and reached;

- Whether object contacts appear physically natural (no penetration or floating);
- Whether the video maintains temporal coherence without flickering or jumps;
- Whether the final state is consistent with the expected action outcome.

This reasoning trace serves two purposes: (1) it improves scoring accuracy by forcing the model to attend to each dimension before committing to a score, and (2) it provides interpretable justifications that can be inspected during reward debugging and used to diagnose failure modes in the world model's predictions.

J.6 From Teacher Judge to Reward Signal

ROBOALIGN-JUDGE serves as the teacher f_ϕ in the reward distillation pipeline (§3.2). Given a language instruction l and a generated video v, the judge produces a structured raw score vector $\hat{\mathbf{r}} = f_\phi(l, v)$ across the six dimensions, using the original rubric ranges defined in Table 28. For student distillation, these raw teacher scores are normalized dimension-wise to $[0, 1]$ before regression. These teacher scores are used in two ways:

(i) **Offline labeling for student distillation.** The teacher scores a mixed corpus consisting of benchmark videos together with generated rollouts from baseline or current world models to create the distillation dataset $\{(l_i, v_i, f_\phi(l_i, v_i))\}$, which trains the lightweight student reward model g_ψ via regression (Eq. 4).

(ii) **Online iterative calibration.** Every K policy updates during GRPO post-training, fresh rollouts from the current world model are scored by the teacher and used to update the student, preventing reward hacking from distributional shift (Algorithm 1, Stage 4).

The teacher's autoregressive decoding cost ($\sim$2.8 seconds per video on a single A100) makes it impractical as a direct online reward. The distilled student reduces this to a single forward pass ($\sim$20 ms), achieving $>10\times$ speedup while maintaining high correlation with teacher judgments.

J.7 Comparison with Alternative VLM Judges

A central question is whether the main findings remain stable under independent model-based evaluation, rather than depending on a single in-domain judge. To answer this question, we conduct an external VLM-based validation by comparing ROBOALIGN-JUDGE against 8 alternative VLM judges—4 proprietary and 4 open-source—on a held-out test set of 50 human-annotated samples from ROBOTWORLDBENCH.

Evaluation protocol. The 50 test samples are drawn from both data sources (25 from synthetic degradation, 25 from real I2V model outputs) and span the full $[0, 10]$ score range. Each sample has human annotations across all six dimensions and is used solely for independent cross-checking rather than for training the main automatic evaluator. All VLM judges receive the identical system prompt (Appendix J.11), the same input format (instruction + initial frame + 8 sampled frames), and are asked to produce the same structured JSON output. For proprietary models, we use the official API with default parameters; for open-source models, we use greedy decoding (T=0) to minimize variance; for ROBOALIGN-JUDGE, we report results with both greedy (T=0) and sampling (T=0.6, top-p=0.95) decoding.

Metrics. We evaluate judge quality along five complementary axes:

(i) **Pearson** ρ: linear correlation between predicted and human total scores, measuring absolute scoring accuracy.

(ii) **Spearman** ρ_s: rank correlation between predicted and human total scores, measuring ordinal consistency—critical for reward-based ranking in GRPO.

(iii) **Per-dimension MAE**: mean absolute error averaged across all six dimensions, measuring fine-grained scoring precision.

(iv) **Pairwise Accuracy**: given all $\binom{50}{2} = 1{,}225$ video pairs, the fraction where the judge's relative ordering agrees with the human ranking—directly measuring the quality of the reward signal for preference-based optimization.

(v) **Inference Cost**: wall-clock time per video on a single NVIDIA A100 40GB GPU (or API latency for proprietary models), measuring practical feasibility as a teacher labeler.

Results. Table 31 reports the full comparison. We view this table as an independent external cross-check of judge consistency on a held-out human-annotated subset, rather than as a replacement for human evaluation itself. Under this protocol, ROBOALIGN-JUDGE remains the most aligned with the held-out annotations while maintaining the same inference cost as its base model.

Table 31: **Comparison of VLM judges on 50 human-annotated test samples from ROBOWORLDBENCH.** All zero-shot judges use the same prompt template. Best results per metric are **bolded**; second-best are underlined. †: API latency includes network overhead.

Judge	Type	Pearson $\rho \uparrow$	Spearman $\rho_s \uparrow$	MAE $\downarrow$	Pair Acc. $\uparrow$	Cost (s) $\downarrow$
Proprietary models (zero-shot)						
GPT-4o	API†	0.72	0.68	0.82	71.3%	∼8.5
GPT-4.1	API†	0.75	0.71	0.76	73.8%	∼7.2
Gemini 2.5 Pro	API†	0.70	0.66	0.85	69.5%	∼6.8
Claude 3.7 Sonnet	API†	0.68	0.64	0.89	67.2%	∼9.1
Open-source models (zero-shot)						
Qwen3-VL-8B-Thinking	Local	0.58	0.54	1.12	62.4%	∼2.8
Qwen2.5-VL-7B	Local	0.52	0.48	1.28	58.1%	∼2.5
InternVL2.5-8B	Local	0.49	0.45	1.35	56.7%	∼2.6
LLaVA-OneVision-7B	Local	0.44	0.40	1.48	53.2%	∼2.4
Fine-tuned (ours)						
ROBOALIGN-JUDGE	Local	**0.89**	**0.86**	**0.41**	**87.6%**	∼2.8

Analysis. Several key observations emerge from Table 31 and Figure 22:

(1) **Fine-tuning dramatically outperforms zero-shot.** ROBOALIGN-JUDGE achieves Pearson $\rho = 0.89$, surpassing the best proprietary model GPT-4.1 ($\rho = 0.75$) by $+0.14$ and its own base model Qwen3-VL-8B-Thinking ($\rho = 0.58$) by $+0.31$. This $+53\%$ relative improvement over the base model demonstrates that domain-specific LoRA fine-tuning on robot manipulation data is essential—general VLMs lack the calibration needed for fine-grained physical and task-level assessment.

(2) **This serves as an independent model-based cross-check.** The Pairwise Accuracy of 87.6% means that in ∼88% of video pairs, ROBOALIGN-JUDGE correctly identifies which video is better—a prerequisite for effective GRPO optimization. In contrast, GPT-4.1 achieves only 73.8%, meaning ∼26% of pairwise comparisons would provide incorrect gradient signals during RL training.

(3) **Open-source zero-shot models are insufficient.** All open-source models score below $\rho = 0.60$ and Pairwise Accuracy below 63%, with particularly poor performance on physics-related dimensions (TC, CR, PA). This is expected: these models are trained on general visual understanding tasks and lack exposure to the specific failure modes of robot manipulation videos.

(4) **Proprietary models provide complementary evidence but remain impractical at scale.** While GPT-4.1 and GPT-4o achieve reasonable correlation ($\rho \geq 0.72$), their API latency (∼7–9 seconds) and per-query cost make them infeasible for large-scale teacher labeling. Labeling 10,000 rollouts would cost ∼\$500–\$1,000 and take ∼20 hours sequentially, compared to ∼\$0 and ∼8 hours for ROBOALIGN-JUDGE on local GPUs with parallelism.

(5) **Per-dimension analysis reveals domain gaps.** Figure 23 shows that the largest improvements from fine-tuning occur in *Physics Adherence* (MAE: 1.42 → 0.38) and *Temporal Consistency* (MAE: 1.18 → 0.29), precisely the dimensions that require understanding of robot-specific physical dynamics. General VLMs perform relatively better on *Instruction Following* (a more semantic/linguistic dimension), but still lag behind the fine-tuned judge.

Evaluation stability. Because ROBOALIGN-JUDGE uses sampling-based decoding at inference time, its output can vary slightly across runs. To measure this effect directly, we evaluate the same 50-sample test set 5 times with T=0.6 and top-p=0.95. Figure 25 shows that this variability is small: Pearson ρ stays within 0.86–0.91 (mean 0.89, std 0.02), and Pairwise Accuracy stays within 85.2%–89.4% (mean 87.6%, std 1.5%). Importantly, the relative ranking of all judges is unchanged across the 5 runs. This indicates that stochastic decoding does not materially affect the judge's

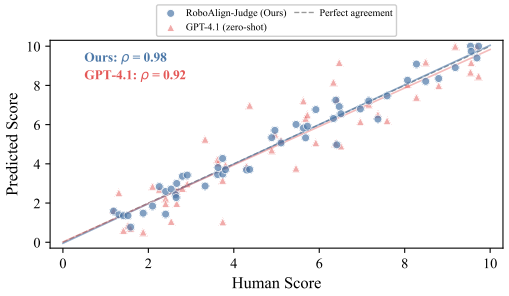


Figure 22: Predicted vs. human total scores for RoboAlign-Judge (blue) and GPT-4.1 (orange) on 50 test samples. The dashed line indicates perfect agreement. RoboAlign-Judge shows tighter clustering around the diagonal.

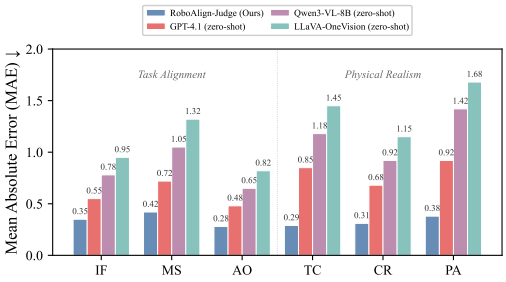


Figure 23: Per-dimension MAE comparison across judge types. Fine-tuning yields the largest improvements on physics-related dimensions (TC, CR, PA), where general VLMs lack domain knowledge.

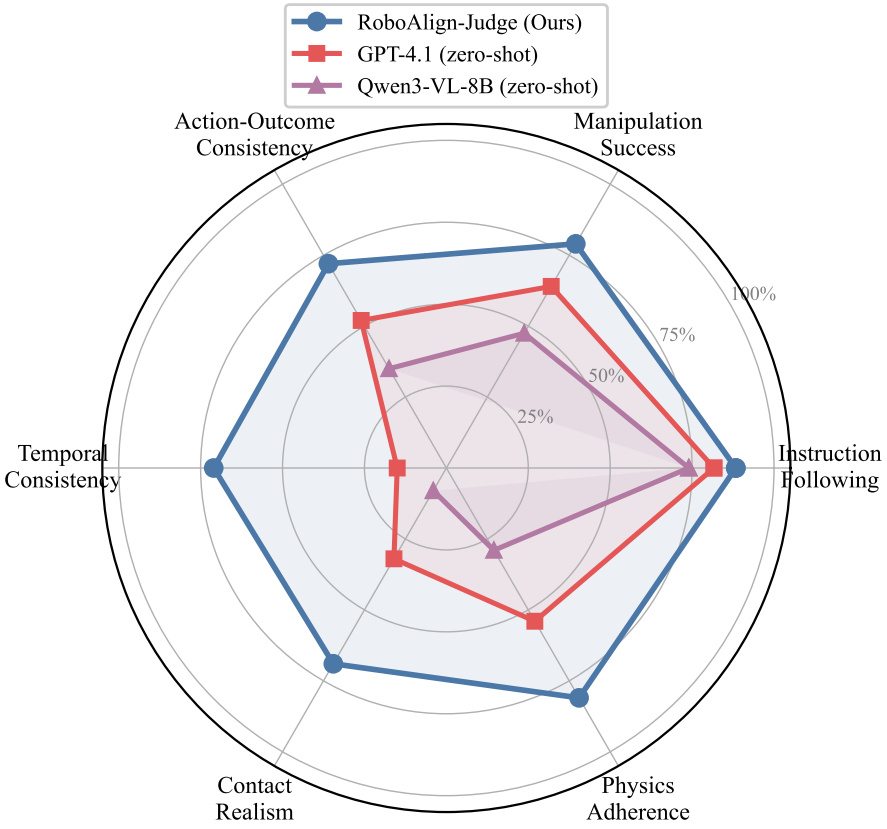


Figure 24: Normalized scoring accuracy (1 − MAE / max_score) across six dimensions for representative judges. RoboAlign-Judge achieves near-uniform high accuracy across all dimensions, while zero-shot models show pronounced weaknesses in physical realism dimensions.

usefulness as a teacher model, since any residual per-sample noise is averaged out during large-batch distillation and RL training. Figure 25 visualizes the stability across runs.

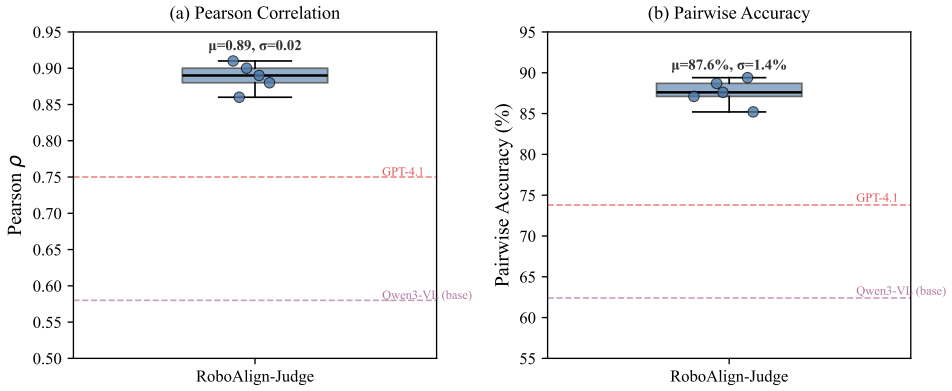


Figure 25: Evaluation stability of ROBOALIGN-JUDGE over 5 independent runs (sampling decoding, T=0.6). Box plots show the distribution of Pearson ρ and Pairwise Accuracy. The narrow interquartile ranges confirm reliable scoring despite stochastic decoding.

J.8 Small-scale Blinded Human Evaluation

To complement the automatic judge-based evaluation, we conduct a small-scale blinded human study on a held-out subset. We sample instruction–initial-frame pairs that are disjoint from judge training and reward-distillation data and compare videos from ROBOALIGN-R1 against strong baselines under anonymized ordering. Each comparison is evaluated along four axes: *overall preference*, *task success*, *physical plausibility*, and *temporal coherence*. Each sample is labeled by multiple raters, and tie decisions are allowed when differences are not visually distinguishable.

Table 32 summarizes the results. Across all pairwise comparisons, ROBOALIGN-R1 is preferred over strong baselines on overall quality and consistently receives higher ratings on task success and physical plausibility. The human ranking is broadly consistent with the automatic ROBOALIGN-JUDGE results, providing complementary evidence that the improvements are not unique to a single in-domain evaluator.

Table 32: **Small-scale blinded human evaluation.** Pairwise human preference on a held-out subset comparing ROBOALIGN-R1 against strong baselines. Higher is better for win rates and mean preference scores.

Comparison	Overall win rate ↑	Task success ↑	Physical plausibility ↑	Temporal consistency ↑	Tie rate	Mean pref. score ↑
ROBOALIGN-R1 vs. iVideoGPT	66.7%	69.3%	71.3%	63.3%	10.0%	3.96 / 5
ROBOALIGN-R1 vs. Wan2.2-TI2V-5B (LoRA)	62.0%	64.7%	66.7%	59.3%	12.7%	3.82 / 5
ROBOALIGN-R1 vs. RLVR-World	59.3%	61.3%	63.3%	57.3%	14.0%	3.74 / 5

J.9 Comparison with Alternative Judge Backbones

Beyond the zero-shot comparison above, we also evaluate the effect of the *backbone choice* for fine-tuning. We select Qwen3-VL-8B-Thinking over alternative backbones based on preliminary experiments on a 50-sample validation set. Table 33 summarizes the comparison.

Table 33: Comparison of candidate judge backbones. Qwen3-VL-8B-Thinking achieves the best balance of scoring accuracy, reasoning quality, and inference efficiency.

Backbone	Params	CoT Reasoning	Multi-frame	Selected
GPT-4o (proprietary)	—	✓	✓	✗ (cost, API latency)
Gemini 2.5 (proprietary)	—	✓	✓	✗ (cost, reproducibility)
Qwen2.5-VL-7B	7B	✗	✓	✗ (no native CoT)
InternVL2.5-8B	8B	✗	✓	✗ (no native CoT)
LLaVA-OneVision-7B	7B	✗	✓	✗ (limited video)
Qwen3-VL-8B-Thinking	**8B**	✓	✓	✓

Key advantages of Qwen3-VL-8B-Thinking: (1) native `<think>` token support enables structured CoT without prompt engineering; (2) dynamic-resolution multi-frame input avoids the need for fixed-size frame preprocessing; (3) open-weight availability ensures full reproducibility and enables LoRA fine-tuning on custom data; (4) 8B scale provides sufficient capacity for nuanced physical reasoning while remaining efficient for batch labeling.

J.10 Limitations and Future Directions

We acknowledge several limitations of the current ROBOALIGN-JUDGE:

- **Dataset coverage.** Although the current training corpus contains 10,000 samples spanning four robot datasets, coverage remains uneven across embodiments, environments, and failure modes. Expanding annotation density and generative-model diversity would likely improve generalization further.
- **Sampling variance.** The CoT decoding introduces non-trivial variance (std ≈ 0.02 on Pearson ρ). Ensemble averaging or deterministic decoding could reduce this, at the cost of inference speed or diversity.
- **Domain transfer.** The current corpus is still dominated by tabletop manipulation and related interaction patterns. Extending to other embodiments (mobile manipulation, dexterous hands) requires additional domain-specific annotation and evaluation.
- **Temporal granularity.** The judge processes 8 uniformly sampled frames, which may miss brief but critical events (e.g., momentary contact). Adaptive frame sampling could improve sensitivity to such events.

J.11 Judge Prompt Templates

For completeness, we provide the prompt templates used for the robot manipulation judge. The prompting interface has two parts: a system prompt that specifies the evaluator role and scoring rubric, and a user message template that injects the task instruction, the initial frame, and uniformly sampled generated frames.

Prompt structure.

Component	Role
System prompt	Defines the judge identity, the six evaluation dimensions, and the required JSON output schema.
User message	Instantiates the instruction, initial frame, and N sampled video frames for a concrete evaluation example.

Scoring rubric.

Dimension	Range	Criterion
Instruction Following	0–3	Whether the robot attempts and correctly executes the instructed action.
Manipulation Success	0–2	Whether the manipulation objective is fully completed by the end of the video.
Action–Outcome Consistency	0–1	Whether the observed outcomes are logically consistent with the robot's actions.
Temporal Consistency	0–1	Whether the video remains smooth and free of major temporal artifacts.
Contact Realism	0–1	Whether object contacts appear physically natural.
Physics Adherence	0–2	Whether the generated motion obeys basic physical plausibility.
Total	0–10	Sum of the six dimension scores.

System Prompt.

```
You are an expert evaluator for robot manipulation video generation quality.
You will be given:
1. An instruction describing what the robot should do
2. An initial frame (the starting state)
3. A sequence of frames from a generated video

Your task is to evaluate the generated video across 6 dimensions. Score each dimension
carefully.

Scoring Dimensions:
1. Instruction Following (0-3 points): Does the video show the robot attempting and
executing the action described in the instruction?
0: Completely unrelated action
1: Vaguely related but wrong action
2: Correct action but incomplete or imprecise
3: Perfectly follows the instruction

2. Manipulation Success (0-2 points): Is the manipulation task successfully completed by
the end of the video?
0: Task completely failed
1: Partial success (object moved but not to target)
2: Full success (task completed as instructed)

3. Action-Outcome Consistency (0-1 point): Are the robot's actions logically consistent
with the observed outcomes?
0: Actions and outcomes are inconsistent
1: Actions and outcomes are consistent

4. Temporal Consistency (0-1 point): Is the video temporally coherent without
flickering, sudden jumps, or artifacts?
0: Severe temporal artifacts
1: Smooth and temporally consistent

5. Contact Realism (0-1 point): When the robot contacts objects, does it look physically
realistic?
0: Unrealistic contacts
1: Contacts look natural and realistic

6. Physics Adherence (0-2 points): Does the video obey basic physics?
0: Severe physics violations
1: Minor physics issues but mostly plausible
2: Fully physically plausible
```

```
Output Format:
First, think step by step about what you observe in the video.  Then output your
evaluation as a JSON object:
{
"reasoning":  "Your detailed analysis",
"instruction_following":  <0-3>,
"manipulation_success":  <0-2>,
"action_outcome_consistency":  <0-1>,
"temporal_consistency":  <0-1>,
"contact_realism":  <0-1>,
"physics_adherence":  <0-2>,
"total":  <0-10>
}
```

User Message Template.

```
Instruction:  {instruction}

Initial Frame (starting state):
[Initial frame image]

Generated Video Frames ({N} frames sampled uniformly):
[Frame 1] [Frame 2] ...  [Frame N]

Please evaluate this generated video based on the 6 dimensions described.
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and the Introduction state three concrete contributions—(i) a reward-aligned RL post-training framework with a distilled multimodal reward (§3.2), (ii) the RoboTWorldBench benchmark and teacher–student reward distillation pipeline (§3.2, Appendix J), and (iii) sliding window re-encoding (SWR) for long-horizon inference (§3.3). All headline numbers ($+10.1\%$ aggregate judge score, $+2.8\%$ SSIM / -9.8% LPIPS from SWR, $\sim 10\times$ reward-cost reduction) are supported by the experiments (see Experiments section) and the appendices; scope limitations (single-arm tabletop manipulation, no downstream policy validation) are stated explicitly in the Conclusion.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Conclusion explicitly states that our current evaluation is limited to single-arm tabletop and countertop manipulation, that downstream policy improvement is not yet validated, and that extending to diverse embodiments and control-level gains remains future work. Appendix J.10 discusses failure modes and limitations of the teacher judge, and Appendix C clearly marks the SWR stability result as a stylized local analysis under explicit assumptions (**A1**–**A2**) rather than a global guarantee.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.

- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Our theoretical results are Proposition 1 and Corollaries 1, 2 (SWR error bound, asymptotic optimum W^*, and SWR-vs-AR gap). Their assumptions **A1** (bounded per-step error with local contraction) and **A2** (bounded decode–re-encode distortion) are stated in Appendix C, and the stylized derivations are given there in full detail with a proof sketch also included in the main text. The limitations of the stylized assumptions are discussed in the "Discussion of assumptions" paragraph of Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The full world-model training pipeline (dual-branch FSQ tokenizer, 12-layer LLaMA-style Transformer, token/action vocabulary, optimization infrastructure, and data splits) is documented in Appendix B with a dedicated Reproduction Protocol (Appendix B.5); the reward-aligned RL post-training procedure is formalized in Algorithm 1 and the student reward model architecture and hyperparameters in Appendix E; SWR inference is given as both pseudocode (Algorithm 2) and a PyTorch implementation (Listing 1). The teacher judge training data, rubric, prompts, and LoRA configuration are in Appendix J. We also release code at the project repository linked in the abstract.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same

dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The project repository and project page are linked in the abstract, and will host the training/inference code for the world model, reward distillation, GRPO post-training, and SWR, together with the RoboticWorldBench annotations and evaluation scripts. All upstream datasets (RT-1, BridgeData V2, CALVIN, LIBERO) are publicly available from their respective creators and are used under their released terms. Anonymized versions of code and data will be provided with the submission where required.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The Experiments section summarizes the main experimental protocol, including datasets (RT-1, BridgeData V2), baselines, and evaluation metrics. Full training details for the world model (optimizer, learning-rate schedule, batch size, number of iterations, hardware) are provided in Appendix B; the student reward model's optimizer, differential learning rates, mixup, EMA, and early-stopping criterion are in Appendix E; the teacher judge's LoRA hyperparameters and training data composition are in Appendix J; SWR evaluation uses the protocol in Appendix G (12-layer LLaMA, hidden 768, 12 heads, vLLM inference, 300 episodes $\times$ 30 frames).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Tables 1, 2, and 3 report mean $\pm$ standard deviation across repeated evaluations over the full generated rollouts. The reported error bars are 1σ standard deviations computed from the per-episode scores, as stated in the corresponding table captions. For the human study (Appendix J.8) we report agreement statistics on the held-out subset.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: All experiments use NVIDIA A100 GPUs; per-stage configurations (e.g., $8\times$ A100-40GB for world-model training, a single A100 for SWR inference measurements and for student/teacher latency profiling) are documented in Appendix B.6 and in the captions of Tables 21, 22, and 23. Per-video inference latency ($\sim$2800 ms teacher vs. $\sim$20 ms student),

RL reward-cost comparisons, and SWR wall-clock / peak-memory numbers are reported in the corresponding tables. We additionally disclose that exploratory runs (tokenizer and RL ablations that did not make it into the paper) consumed additional compute beyond the reported runs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses publicly available robot manipulation datasets under their original licenses, does not involve personally identifiable information, and relies only on simulated and tabletop manipulation videos. The human evaluation (Appendix J.8) was conducted on a small internal panel reviewing de-identified video clips of tabletop manipulation with fairly compensated volunteer evaluators and no sensitive attributes. We have reviewed the NeurIPS Code of Ethics and confirm that our work conforms to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Positive impacts include more sample-efficient and safer robot learning—better world models can act as learnable simulators for planning and policy evaluation, reducing the need for real-robot trials and improving physical plausibility of predicted rollouts. Negative impacts are limited but nontrivial: improved robot video generation could be misused to synthesize misleading robot demonstrations, and reward-model distillation could inherit biases from the teacher VLM. We do not anticipate immediate deployment risks because the models are restricted to tabletop manipulation scenarios, and we recommend gated release and clear provenance metadata when distributing trained generators, in line with common practice for generative models.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our released artefacts consist of a benchmark of annotated tabletop manipulation videos (sourced from existing, already-released robot datasets) and a world model and reward model narrowly trained on single-arm tabletop manipulation. The model cannot generate human faces, persons, or open-domain imagery, so we do not consider it to pose a high misuse risk. Should the project be extended to broader embodiments or open-domain video generation, we will revisit the need for gated release and additional safety filters.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All upstream datasets and models are cited at first use in the Experiments section and in the appendices: RT-1 [38], BridgeData V2 [39], CALVIN [40], LIBERO [41], Qwen3-VL-Thinking [36], LoRA [42], and the baseline world models and video generators compared in Table 1. These assets are used in accordance with their respective licenses (mostly Apache-2.0 / MIT / research-use licenses as posted by their maintainers). The aggregated annotation files released with RoboticWorldBench will be distributed under a CC-BY-4.0 license together with per-source license notices.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release three new assets: (i) RoboWorldBench, a 10,000-pair annotated video–instruction benchmark, documented in §3.2 and Appendix J.2 (data sources, degradation pipeline, rubric, annotation protocol); (ii) RoboAlign-Judge, a LoRA-fine-tuned Qwen3-VL teacher judge, documented in Appendix J together with its prompt templates (Appendix J.11); and (iii) the distilled student reward model (§3.2, Appendix E). Each asset will be accompanied by a model/dataset card describing training data, intended use, limitations, and license.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: The small-scale blinded human evaluation (Appendix J.8) uses internal volunteer raters, not crowd workers; Appendix J.8 provides the full instructions given to raters, the rubric and interface template, and the number of pairwise comparisons used. Raters were fairly compensated at or above the local minimum wage and were not exposed to personally identifiable or sensitive content. No identifying information about raters is collected or released.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: Our human evaluation consists only of blinded rating of short tabletop manipulation video clips with no personally identifiable or sensitive content; under the relevant institutional guidelines such minimal-risk evaluation does not require formal IRB review. The task, expected effort, and voluntary nature of participation were disclosed to all raters before they consented to participate, as described in Appendix J.8.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: A multimodal LLM (Qwen3-VL-8B-Thinking) is used as the backbone of the RoboAlign-Judge teacher model after LoRA fine-tuning on RobotWorldBench (§3.2, Appendix J). Its role, training data, and how its scores are distilled into the student reward model used for RL post-training are described in detail. Any LLM use for writing or editing is consistent with NeurIPS policy and is not part of the core methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.