VIDEO-RUBRICS: FINE-GRAINED REWARDS FOR FAITHFUL EMBODIED VIDEO GENERATION

Anonymous authors
Paper under double-blind review

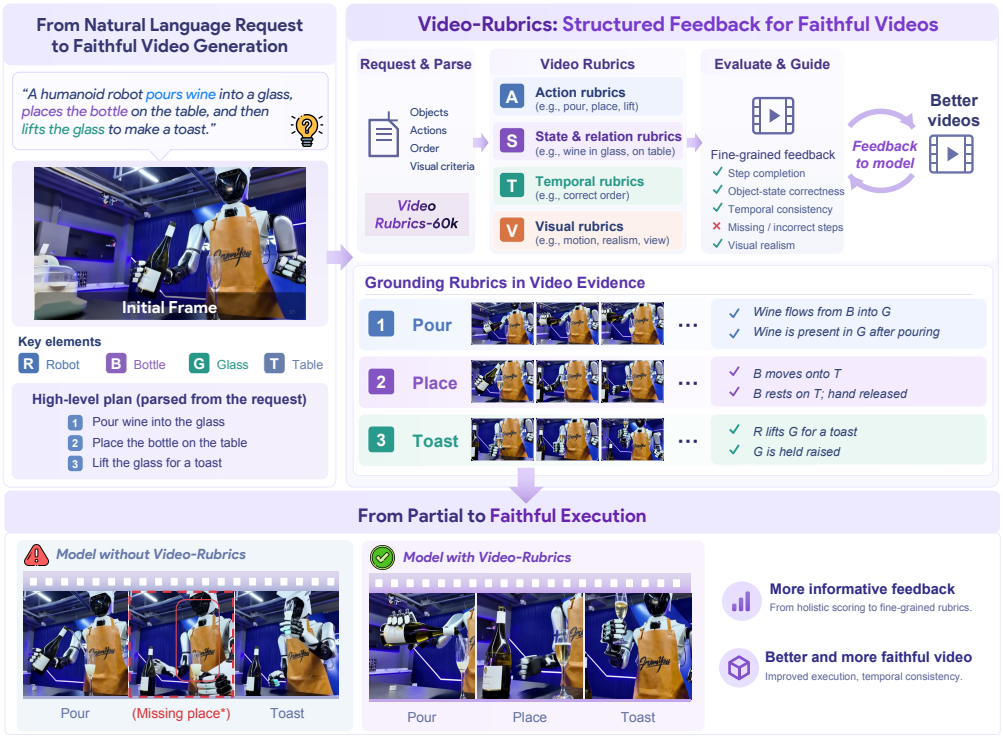


Figure 1: **Overview of Video-Rubrics.** Video-Rubrics translates multi-step requests into verifiable criteria, using video evidence to provide fine-grained feedback on actions, completion states, and temporal order, which promotes complete and correctly ordered execution of the original request.

ABSTRACT

Video diffusion models have made substantial progress in visual quality, yet instruction adherence remains challenging in embodied video generation. Semantic rewriters enrich the conditions supplied to video generators, but rewards based only on rewritten requirements can favor plans that omit difficult parts of the original request. We introduce **Video-Rubrics**, a reinforcement learning framework that jointly optimizes a prompt rewriter and a video generator through fine-grained visual criteria. We construct Video-Rubrics-60K, a training set of image–text requests spanning general scenes and robotic instructions, with atomic requirements annotated independently of candidate rewrites. Original-prompt rubrics supervise both requirement preservation and visual realization for the rewriter, while condition rubrics evaluate how faithfully the generator follows its actual input. A frozen vision-language verifier assesses each requirement using timestamped video evidence, with soft temporal windows that preserve action context. We normalize item-level rewards within each policy's comparison groups and use the resulting advantages to update both policies from the same sampled videos. Experimental results show that Video-Rubrics improves instruction adherence across three video backbones, with gains on multi-entity interactions and long-horizon tasks.

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

1 INTRODUCTION

Video diffusion models (VDMs) have made substantial progress in visual quality and temporal coherence, advancing content creation (Gao et al., 2025; MiniMax, 2026a; Team Wan et al., 2025; Meituan LongCat Team et al., 2025). We study instruction adherence in embodied video generation, where success depends on preserving object identities, spatial relations, and the order of actions. Even visually coherent videos may depict actions on the wrong objects, omit required steps, or reverse the intended sequence (Deng et al., 2026). Recent systems complement video generators with semantic preprocessing: LingBot-Video and Cosmos3-Nano use language or vision–language models to expand and structure user requests (Ma et al., 2026; NVIDIA, 2026), while MiniMax-H3 introduces a dedicated context-processing stage before generation (MiniMax, 2026b). Such preprocessing draws on semantic knowledge and physical priors to enrich brief requests with scene details and action context (Ma et al., 2026; NVIDIA, 2026).

In this pipeline, a semantic rewriter expands the user's request into a generation condition, which the video generator then realizes as a sequence of frames. However, semantically richer descriptions do not necessarily lead to videos that better follow the original request. This gap motivates joint post-training of the rewriter and generator, allowing generation conditions to adapt to the generator through feedback on the resulting videos (Wang et al., 2026a; Liu et al., 2026).

Joint post-training requires rewards that remain meaningful as generation conditions change. Rubric-based supervision expresses visual requirements as individual criteria (Tian et al., 2026; Ma et al., 2026), but deriving those criteria solely from each rewrite allows the rewriter to change its own evaluation target. For example, an instruction to pick up a red cup and then place a blue bowl inside a drawer can be rewritten to retain only the bowl placement. The resulting video may satisfy every rewritten requirement while omitting a requested action. Rewarding this outcome would favor simplifying the task over learning to complete it.

We introduce **Video-Rubrics**, a framework for joint post-training through fine-grained visual criteria as shown in Figure 1. We train the rewriter to preserve the original requirements and improve their realization in generated videos, while training the video generator to follow the rewritten condition. Fixed *original-prompt rubrics*, constructed independently of candidate rewrites, provide the rewriter's criteria; *condition rubrics*, derived from the actual generator input, provide the criteria for video execution. A frozen vision–language verifier evaluates each criterion on video evidence, and separate item-level advantages update both policies from shared video rollouts. The rewriter's evaluation target thus remains fixed as its generation plans evolve.

To support this framework, we construct Video-Rubrics-60K, a corpus of 60,000 image–text requests with fixed original-prompt rubrics, combining general scenes and robotic instructions. General examples are included throughout post-training to retain broad video-generation capabilities during adaptation to robotic tasks. We evaluate on RBench (Deng et al., 2026), whose manipulation, spatial, multi-entity, and long-horizon tasks provide a focused test of whether generated videos fulfill the requested interactions and action sequences. Across LingBot-Video, Cosmos3-Nano, and MiniMax-H3, Video-Rubrics improves RBench performance over the corresponding base models, with gains on multi-entity interactions and long-horizon tasks.

Our main contributions are as follows:

- We introduce Video-Rubrics, a reinforcement learning framework that jointly optimizes a prompt rewriter and a video generator for faithful embodied video generation. Fixed original-prompt rubrics anchor rewriter supervision to user requirements, while condition rubrics guide video execution.
- We formulate item-level credit assignment that normalizes rubric rewards within policy-specific comparison groups, yielding separate advantages for the rewriter and generator from shared video rollouts.
- We construct Video-Rubrics-60K, an image–text training set spanning general scenes and robotic instructions, with atomic requirements annotated independently of candidate rewrites. Experiments across three video backbones show improved instruction adherence, with gains on multi-entity interactions and long-horizon tasks.

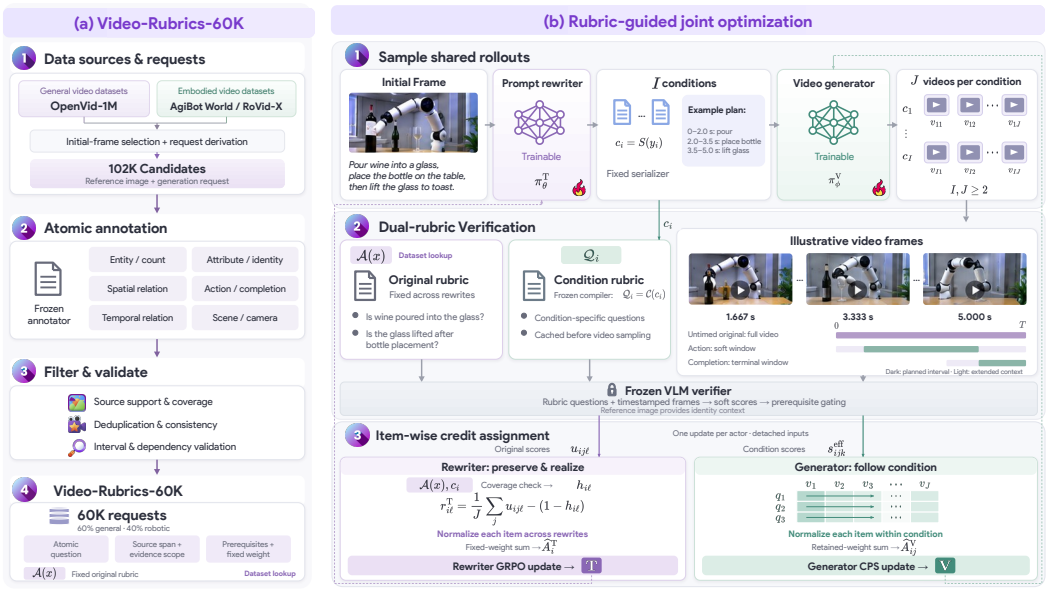


Figure 2: **Overview of Video-Rubrics.** Offline annotation constructs fixed original-prompt rubrics in Video-Rubrics-60K. Online rollouts provide both original- and condition-item visual scores. The text policy compares rewrites using original-item realization and coverage, while the video policy compares candidates under the same condition rubric. Each actor uses its own item-wise advantages.

2 METHOD

2.1 FRAMEWORK OVERVIEW

Video-Rubrics jointly optimizes a prompt rewriter and a video generator to improve generation conditions and their visual realization. We consider video generation from an original prompt $x \sim \mathcal{D}$ and a reference image shared across its rewrites and videos; we omit this image from the notation. The rewriter samples a structured caption y describing the scene, entities, and actions, optionally with planned action intervals. A fixed adapter S validates and serializes y into the condition supplied to the generator:

$$y \sim \pi_\theta^{\mathrm{T}}(\cdot \mid x), \qquad c = S(y), \qquad v \sim \pi_\phi^{\mathrm{V}}(\cdot \mid c).$$

With the rollout policies fixed at $(\theta_{\mathrm{old}}, \phi_{\mathrm{old}})$, we sample $I \geq 2$ rewrites y_i for each x and $J \geq 2$ videos v_{ij} for each valid condition $c_i = S(y_i)$.

We associate each prompt with a fixed original-prompt rubric $\mathcal{A}(x)$ of $L_x > 0$ items and compile a condition rubric from the exact input supplied to the generator. The former anchors rewrite comparisons to the original requirements, while the latter compares videos generated under the same condition. A frozen vision–language verifier scores individual requirements; policy-specific advantages then support separate updates from shared video rollouts (Figure 2).

2.2 RUBRIC CONSTRUCTION

The two rubric types share an annotation schema but derive their criteria from different inputs: the original request for the rewriter and the serialized generation condition for the video generator.

Original-Prompt Rubrics. Video-Rubrics-60K contains 60,000 image–text requests retained from 102,000 annotated candidates, comprising 60% general scenes and 40% robotic instructions. General-scene requests are drawn from OpenVid-1M (Nan et al., 2025), and robotic requests from the imitation-learning subset of AgiBot World 2026 (AgiBot World Team, 2026) and RoVid-X (Deng et al., 2026). Each request pairs an initial reference frame with an instruction derived from the corresponding caption or subtask annotation. Offline filtering removes duplicates and requests with

malformed, unresolved, or contradictory constraints. Source videos support this offline construction but are not used as video-matching targets during policy optimization.

Gemini 3.7-Flash decomposes each image-conditioned request into atomic requirements using fixed annotation and checking prompts. The reference image resolves entity identities and participant bindings, while the original prompt determines the requirements. The six annotation types are *entity and count*, *attribute and identity binding*, *spatial relation*, *action and state change*, *temporal relation*, and *scene and camera constraint*. For actions with a specified endpoint, we separately annotate the motion and its entailed completion state, distinguishing attempted from completed interactions. Each item records a binary question, type and subtype, supporting source span, reference-image grounding flag, evidence scope, and prerequisites. Original items use full-video evidence unless the request explicitly specifies a time range.

Deterministic validation canonicalizes entities, merges duplicate propositions, checks intervals and references, and rejects cyclic prerequisite graphs. Each distinct proposition receives unit weight. Participant-existence or identity-preservation requirements serve as prerequisites for attributes, actions, and relations (Cho et al., 2024); explicit absence criteria have no positive-existence prerequisite. Before training, we freeze the accepted rubric $\mathcal{A}(x)$, its item identities and weights, and its annotation configuration, independently of sampled rewrites and videos. Appendix B details request construction and annotation, including the type–subtype hierarchy in Figure 6.

Condition Rubrics. For each valid condition c_i, a frozen Qwen3.8-27B compiler extracts its supported propositions into $\mathcal{Q}_i = \mathcal{C}(c_i)$, with $K_i > 0$ items, using the same annotation schema. Both compilation and coverage checking are text-only and read the final serialized condition supplied to the generator. Compilation does not use $\mathcal{A}(x)$; coverage checks each original requirement directly against c_i using the same frozen checkpoint. Task-specific visual constraints are eligible regardless of their field, whereas generic background knowledge does not introduce separate criteria. The compiler preserves subjects, polarity, argument bindings, temporal relations, and temporal extent, and caches the validated rubric with the exact condition before video generation. All videos for c_i share $\mathcal{Q}_i$, while $\mathcal{A}(x)$ remains fixed across rewrites.

2.3 Temporal Verification

Temporal Evidence. For a video of duration T, planned action intervals satisfy $0 \le t^{\mathrm{s}} < t^{\mathrm{e}} \le T$. Condition items use action intervals, terminal scopes from an event's planned completion to the next planned action on the same affected object (or T), and union scopes defined by the smallest interval containing the participating action intervals. Completion is assessed relative to its event rather than the video's final frame. Planned timestamps guide evidence retrieval rather than impose deadlines; explicit timing requirements remain part of the question. Untimed original items retain full-video scopes regardless of the rewrite schedule; condition items use full-video evidence for video-wide constraints or when no supported local scope is available.

To vary boundary context during training, we pad action and union scopes independently on both sides by $\mathcal{U}(0.3, 0.7)$ seconds and clip to $[0, T]$ (Figure 3). Terminal scopes receive left padding only. Condition-item padding is shared across videos for c_i, while original-item padding is shared across all rewrites and videos for x. Evaluation uses fixed 0.5-second margins wherever padding applies. Each window supplies up to 16 distinct generated frames nearest to uniformly spaced timestamps, retaining their original timestamps and temporal order. The reference image is provided separately as identity context.

Visual Scores. The frozen Qwen3.8-27B verifier evaluates both rubric types on the selected generated frames. The reference image resolves identities and bindings but does not establish that a requirement is satisfied in the generated video. For item k, let $z_{k,b}$ denote the next-token logit for verdict $b \in \{0, 1\}$ at a fixed answer prefix where each verdict is represented by one token. We define

$$s_k(v) = \mathrm{sigmoid}(z_{k,1} - z_{k,0}). \tag{1}$$

This score measures relative support for satisfaction over non-satisfaction, following probability-based visual verification (Huang et al., 2026).

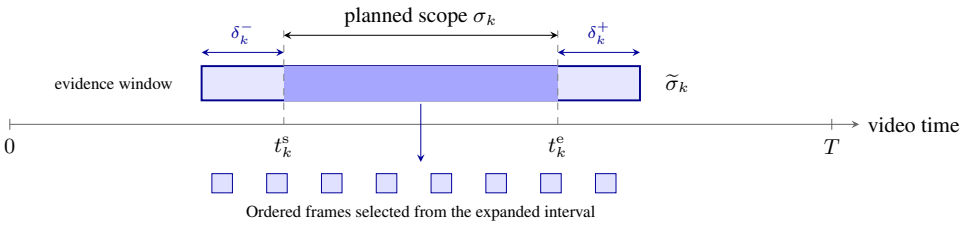


Figure 3: **Soft temporal windows for verification.** We expand planned action and union intervals on both sides to include surrounding context. Frames sampled from the expanded windows retain original timestamps and temporal order, while each question's temporal constraints remain unchanged.

Dependency Gating and Weighting. An item's effective score s_k^{eff} equals its raw score s_k when every prerequisite ancestor has a raw score of at least 0.5, and is zero otherwise; items without prerequisites retain their raw scores. For each condition, we sample a binary retention mask $m_{ik} \in \{0, 1\}$ with per-item retention probability 0.9 (Yang et al., 2026), shared across its videos; an all-zero mask is resampled. With unit item weights, the normalized weights are $\alpha_{ik} = m_{ik} / \sum_{k'=1}^{K_i} m_{ik'}$ and $\omega_\ell = 1/L_x$. Gating does not change these weights, and prerequisites are evaluated even when their own items are dropped. All original items are retained, preserving fixed evaluation criteria and weights across rewrites; evaluation retains every item in both rubric types. Appendix C provides the compilation and verification protocols, scope definitions, and weighting details.

2.4 Rewards and Advantages

The two policies use shared videos but different comparison groups: rewrites of the same original request for the text policy, and videos generated under the same condition for the video policy. We extend item-wise group-relative normalization (Tian et al., 2026) to both groups, normalizing each item's rewards before aggregation.

Rewriter Reward. Let $u_{ij\ell}$ denote the dependency-gated score of original item ℓ on v_{ij} under its original evidence scope, using the same scoring and gating procedure as condition items. Let $h_{i\ell} \in \{0, 1\}$ indicate whether the frozen coverage checker finds that c_i entails this original requirement. The rewriter's item reward combines average visual realization with a coverage penalty:

$$r_{i\ell}^{\mathrm{T}} = \frac{1}{J} \sum_{j=1}^{J} u_{ij\ell} - \lambda_{\mathrm{cov}}(1 - h_{i\ell}), \qquad \lambda_{\mathrm{cov}} = 1. \tag{2}$$

The penalty applies to each original requirement not preserved in c_i. The criteria and their aggregation weights remain fixed by $\mathcal{A}(x)$, so added planning details do not expand the rewriter's reward criteria.

Item-Wise Advantages. Let $\boldsymbol{r}_\ell^{\mathrm{T}} = (r_{1\ell}^{\mathrm{T}}, \ldots, r_{I\ell}^{\mathrm{T}})$ collect one original item's rewards across rewrites, and $\boldsymbol{s}_{ik}^{\mathrm{eff}} = (s_{i1k}^{\mathrm{eff}}, \ldots, s_{iJk}^{\mathrm{eff}})$ collect one condition item's scores across videos. Let $\mathcal{N}_{\epsilon_{\mathrm{std}}}$ subtract the group mean and divide by the population standard deviation, with the denominator floored at $\epsilon_{\mathrm{std}} = 0.05$. The policy-specific advantages are

$$\widehat{A}_i^{\mathrm{T}} = \sum_{\ell=1}^{L_x} \omega_\ell \mathcal{N}_{\epsilon_{\mathrm{std}}}(\boldsymbol{r}_\ell^{\mathrm{T}})_i, \tag{3}$$

$$\widehat{A}_{ij}^{\mathrm{V}} = \sum_{k=1}^{K_i} \alpha_{ik} \mathcal{N}_{\epsilon_{\mathrm{std}}}(\boldsymbol{s}_{ik}^{\mathrm{eff}})_j. \tag{4}$$

Condition items are compared only within the same c_i, whereas each original item has a fixed meaning across all rewrites of x. Equal-score items contribute zero without redistributing their weights, and the summed advantage is not normalized again. Rewriter rewards average original-item visual scores before group normalization; averaging centered video advantages would remove the signal needed to compare rewrites. If every rewrite in a comparison group has the same coverage label for an original item, that item's coverage term cancels under centering.

2.5 JOINT POLICY OPTIMIZATION

We optimize separate text- and video-policy objectives on the same rollout batch, following group-relative policy learning (Shao et al., 2024; Liu et al., 2025a) and shared-rollout joint training (Wang et al., 2026a; Liu et al., 2026). Sampled conditions, compiled rubrics, checker outputs, and advantages are treated as constants; gradients are computed only through the likelihood of the policy being updated. Throughout, $\mathbb{E}_{\text{old}}$ denotes averaging over the relevant comparison groups and rollouts collected by the fixed old policies.

Text Policy Update. We use single-update, on-policy GRPO (Shao et al., 2024) with the rewrite advantages above, computing likelihoods on the sampled tokens y_i before serialization. Let $\rho_{in}(\theta)$ denote the current-to-old conditional likelihood ratio for token y_{in}, N_i the number of sampled non-padding tokens, and $\ell_{\epsilon_{\text{clip}}}$ the standard clipped policy surrogate. We maximize

$$\mathcal{J}_{\text{T}}(\theta) = \mathbb{E}_{\text{old}}\left[\frac{1}{I}\sum_{i=1}^{I}\frac{1}{N_i}\sum_{n=1}^{N_i}\ell_{\epsilon_{\text{clip}}}\big(\rho_{in}(\theta), \widehat{A}_i^{\text{T}}\big)\right]. \tag{5}$$

All sampled tokens, including EOS when sampled, receive the same rewrite advantage; padding is excluded. Each fresh rollout batch with a usable text comparison group supplies one optimizer step starting from θ_{old}, without KL regularization.

Video Policy Update. We apply CPS exploration (Ma et al., 2026; Wang & Yu, 2025) at a single step $t^\star$, sampled uniformly from eligible first-half denoising steps with positive exploration variance and shared across the batch. The selected transition has exact Gaussian density p_ϕ with sampler-fixed variance; all other transitions follow the deterministic sampler. For valid condition groups, we maximize the selected-transition surrogate

$$\mathcal{J}_{\text{V}}(\phi) = \mathbb{E}_{\text{old}}\left[\frac{1}{J}\sum_{j=1}^{J}\lambda_{t^\star}\widehat{A}_{ij}^{\text{V}}\log p_\phi(z_{ij,t^-} \mid z_{ij,t^\star}, c_i)\right], \tag{6}$$

Here, t^- is the next timestep after $t^\star$ in the denoising schedule, and the expectation includes valid condition groups and the sampled exploration step. The weight λ_t compensates for the sampler-dependent scaling of the Gaussian log-likelihood gradient with respect to the predicted velocity at the rollout parameters and has mean one over eligible steps. With a usable video comparison group, we take one on-policy gradient step from ϕ_{old} without KL regularization, treating the remaining denoising trajectory as fixed; this is a selected-transition surrogate rather than a gradient through the full sampling process.

Update Schedule. Each optimizer with a usable comparison group updates once, and all applicable updates finish before the next rollout batch is collected. Rewrites that fail content validation receive $r_{i\ell}^{\text{T}} = -1$ for every original item and are excluded from video generation. Inference and infrastructure failures are retried once; repeated failures exclude affected comparison groups rather than yield negative rewards. Independently complete groups remain usable for their respective policy updates. Appendix D provides the policy-surrogate definitions, CPS derivation, timestep compensation, and complete update procedure.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

Models, Data, and Scoring. We post-train LingBot-Video 30B-A3B (Ma et al., 2026), Cosmos3-Nano (NVIDIA, 2026), and MiniMax-H3 (MiniMax, 2026a) with Video-Rubrics. LingBot-Video is used for the controlled ablations in Tables 2 and 3 and the test-time rollout-scaling study. All three models are trained on the 60,000 image–text requests in Video-Rubrics-60K, maintaining a 6:4 mixture of general-scene and robotic requests throughout post-training. Gemini 3.7-Flash annotates original-prompt rubrics offline and checks them in a second pass under fixed multimodal prompts; the resulting rubrics remain fixed during policy optimization. The condition compiler, coverage checker, and visual verifier share a frozen Qwen3.8-27B endpoint served with vLLM in FP8, with thinking

Table 1: **Quantitative results on RBench.** We compare base models and their Video-Rubrics variants across five task dimensions and four robot embodiments. Overall is the unweighted mean of the nine category scores.

Model	Overall	Task Dimensions					Robot Embodiments			
		Manip.	Spatial	ME	LH	Reason.	Single	Dual	Quad.	Human.
LingBot-Video (Ma et al., 2026)	0.614	0.578	0.643	0.444	0.634	0.505	0.636	0.639	0.758	0.689
+ *Video-Rubrics*	**0.625**	0.592	0.654	0.459	0.639	0.503	0.647	0.645	0.776	0.709
Cosmos3-Nano (NVIDIA, 2026)	0.584	0.550	0.670	0.466	0.576	0.394	0.591	0.611	0.731	0.666
+ *Video-Rubrics*	**0.591**	0.563	0.675	0.472	0.580	0.396	0.600	0.618	0.740	0.674
MiniMax-H3 (MiniMax, 2026a)	0.660	0.602	0.627	0.600	0.765	0.555	0.712	0.708	0.700	0.672
+ *Video-Rubrics*	**0.669**	0.607	0.639	0.611	0.766	0.562	0.723	0.702	0.729	0.685

disabled. The compiler and checker use text-only inputs and greedy decoding, while the verifier receives the reference image and up to 16 timestamped generated frames to supply the same-prefix verdict logits. Appendix B details the data sources, filtering procedure, and rubric annotation.

Implementation Details. We use LingBot-Video's prompt rewriter (Ma et al., 2026) for all three video backbones and fine-tune its Qwen3.6-27B *Map* module using LoRA. The optional *Expand* stage remains frozen. We train for 1,000 iterations, sampling four prompts, four rewrites per prompt, and four videos per valid rewrite at each iteration. The rewriter and video generator use separate AdamW optimizers, with at most one update per policy per iteration. For each optimizer, the learning rate increases linearly to 10^{-6} over the first 50 updates and remains constant thereafter. Gradient norms are clipped at 1.0. The rewriter uses GRPO (Eq. 5) with a clipping parameter of 0.2, while the video generator follows LingBot-Video's single-step, on-policy CPS procedure with $\eta = 0.7$. Neither policy uses KL regularization. All experiments are conducted on NVIDIA H20 GPUs. Table 6 in Appendix E summarizes the training and scoring settings.

Evaluation. We evaluate on RBench using its official image–text conditions and evaluator (Deng et al., 2026), reporting Overall and scores across five task dimensions and four robot embodiments. All models and ablations are evaluated against the original requirements rather than rewritten conditions. For each backbone, Base uses the initial rewriter and video-generator checkpoints without post-training. Base and Video-Rubrics use the same evaluation inputs and generation settings.

3.2 Experimental Results

Quantitative Results. Table 1 shows higher Overall scores for Video-Rubrics than the reported baselines across all three video backbones, reaching 0.625 on LingBot-Video, 0.591 on Cosmos3-Nano, and 0.669 on MiniMax-H3. The gains extend across manipulation, spatial relations, multi-entity interactions, and long-horizon tasks, rather than being confined to a single task category. Across all three backbones, the increase in ME exceeds that in LH, indicating that the reported improvements are more pronounced in multi-entity coordination than in long-horizon execution. Scores also improve across single-arm, quadruped, and humanoid embodiments for every backbone. The improvements are not uniform, however: LingBot-Video shows a slight decrease in reasoning, and MiniMax-H3 in dual-arm performance. The controlled ablations below examine the contributions of text- and video-policy updates and temporal verification.

Qualitative Results. Figure 4 illustrates differences in task execution under the same image–instruction inputs. In the LingBot-Video example, both target objects remain on the tabletop in the baseline frames, whereas the post-trained model depicts the left gripper lifting the labeled cube and the right gripper lifting the die. The Cosmos3-Nano example shows the post-trained model transferring the ball into the orange container and withdrawing the gripper, providing visible evidence of the requested placement outcome. For MiniMax-H3, the baseline places the lemon inside the basket, while Video-Rubrics places it to the basket's right from the camera's perspective. These examples highlight the distinction between depicting a relevant action and satisfying its intended

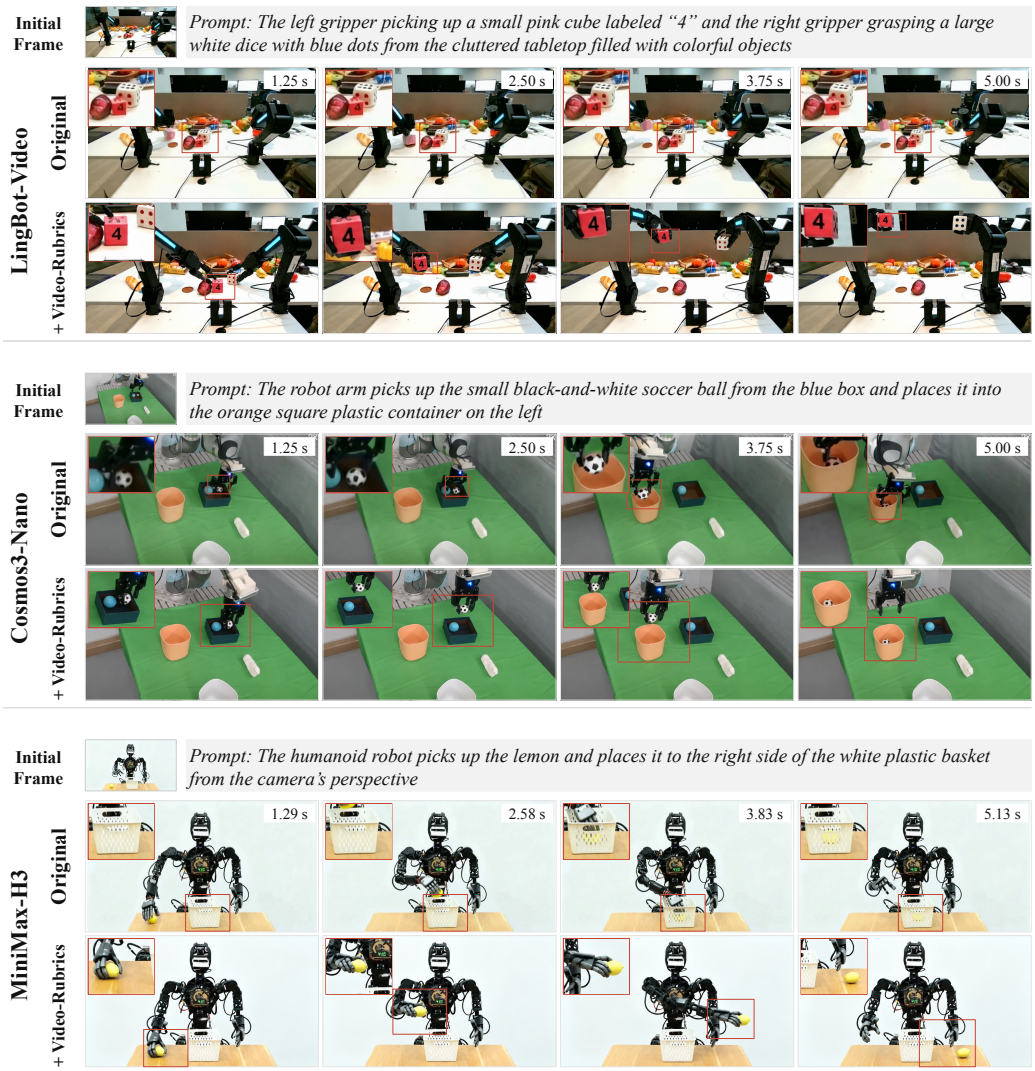


Figure 4: **Qualitative comparisons on RBench.** For each input image and instruction, we show four chronologically ordered frames from the base model and its Video-Rubrics variant. Insets highlight object interactions and spatial relations relevant to instruction adherence.

outcome: grasping must involve the specified participants, and placement must establish the requested object relation. This distinction is consistent with our separate verification of actions, completion states, and relations, which makes these requirements explicit rather than treating an attempted interaction as sufficient evidence of success.

3.3 Controlled Ablations

Joint Policy Optimization. We compare text-only, video-only, alternating, and joint updates under the same rendered-video budget in Table 2. Joint training achieves the highest Overall, ME, and LH scores, with a slightly higher Overall score than alternating updates. These results support optimizing both the rewriter and the generator under a fixed video budget. Joint training updates both policies per rollout batch, whereas alternating training updates only one; this comparison therefore matches video exposure but not optimizer-step counts.

Table 2: **Effect of policy update strategy.** All variants use the same initialization, reward formulation, and video-generation budget.

Update	Overall ↑	ME ↑	LH ↑
Text only	0.620	0.453	0.635
Video only	0.617	0.450	0.632
Alternating	0.623	0.447	0.638
Joint	**0.625**	**0.459**	**0.639**

ME: multi-entity; LH: long-horizon.

Table 3: **Effect of temporal evidence windows.** We vary windows used for condition-rubric verification keeping per-item frame budgets fixed.

Window	Overall ↑	ME ↑	LH ↑
Full video	0.619	0.454	0.631
Strict (no padding)	0.622	0.455	0.634
Fixed	0.623	0.457	**0.640**
Sampled	**0.625**	**0.459**	0.639

ME: multi-entity; LH: long-horizon.

Temporal Evidence. We vary condition-rubric evidence windows during training while keeping criterion content, the per-item frame budget, and RBench evaluation fixed (Table 3). Strict event scopes improve Overall from 0.619 with full-video evidence to 0.622; fixed 0.5-second padding and sampled $\mathcal{U}(0.3, 0.7)$-second margins further increase it to 0.623 and 0.625, respectively. Sampled margins achieve the highest Overall and ME scores, while fixed padding yields a slightly higher LH score. This pattern suggests that event-focused verification benefits from temporal context around scope boundaries. Appendix E.2 compares training-reward trajectories under sampled and fixed margins across all three backbones.

Test-Time Rollout Scaling. Using the post-trained LingBot-Video model, we measure how the best observed condition-rubric reward varies with the number of sampled video rollouts in Figure 5. We fix both post-trained policies and use one rewrite per original condition to generate a shared pool of eight videos, with nested prefixes defining budgets $N \in \{1, 2, 4, 8\}$. For each request, we take the highest raw reward among its first N rollouts and average these best-of-N rewards across requests. Mean reward rises from 0.332 at $N = 1$ to 0.451 at $N = 8$, an absolute gain of 0.119. Successive doublings yield gains of 0.053, 0.044, and 0.022, indicating diminishing returns from additional candidates under the rubric reward.

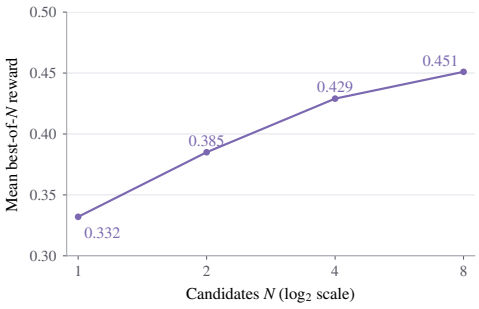


Figure 5: **Test-time rollout scaling.** Best-of-N condition-rubric reward averaged across requests.

4 Conclusion

We introduced **Video-Rubrics**, a reinforcement learning framework for faithful embodied video generation through joint optimization of a prompt rewriter and a video generator. Fixed original-prompt rubrics from Video-Rubrics-60K keep rewrite evaluation anchored to user requirements as generation conditions evolve, while condition rubrics assess videos against the generator's actual input. By separately verifying actions and their completion states, the framework distinguishes attempted interactions from the realization of their intended outcomes. Temporal verification supplies item-level feedback, which is normalized within policy-specific comparison groups to update both models from shared video rollouts. Experiments on RBench show improved instruction adherence across three video backbones, with gains on multi-entity interactions and long-horizon tasks. Ablations further support joint policy updates under a fixed video budget and the inclusion of temporal context around event-focused evidence windows. Our rewards remain dependent on the coverage and accuracy of automatically constructed rubrics and the reliability of visual verification, particularly for temporally complex requirements. Future work will explore uncertainty-aware verification and adaptive temporal evidence allocation for more reliable and efficient reward computation.

AI USE STATEMENT

We used generative AI tools to polish the manuscript, assist with data annotation, and write portions of the code. The authors reviewed the final text, code, and results and take responsibility for the content of this work, including all AI-assisted material.

REFERENCES

AgiBot World Team. AgiBot World 2026. https://huggingface.co/datasets/agibot-world/AgiBotWorld2026, 2026.

Jaemin Cho, Yushi Hu, Roopal Garg, Peter Anderson, Ranjay Krishna, Jason Baldridge, Mohit Bansal, Jordi Pont-Tuset, and Su Wang. Davidsonian Scene Graph: Improving Reliability in Fine-grained Evaluation for Text-to-Image Generation. *arXiv preprint arXiv:2310.18235*, 2024.

Yufan Deng, Zilin Pan, Hongyu Zhang, Xiaojie Li, Ruoqing Hu, Yufei Ding, Yiming Zou, Yan Zeng, and Daquan Zhou. Rethinking Video Generation Model for the Embodied World. *arXiv preprint arXiv:2601.15282*, 2026.

Yu Gao, Haoyuan Guo, Tuyen Hoang, Weilin Huang, Lu Jiang, Fangyuan Kong, Huixia Li, Jiashi Li, Liang Li, Xiaojie Li, Xunsong Li, Yifu Li, Shanchuan Lin, Zhijie Lin, Jiawei Liu, Shu Liu, Xiaonan Nie, Zhiwu Qing, Yuxi Ren, Li Sun, Zhi Tian, Rui Wang, Sen Wang, Guoqiang Wei, Guohong Wu, Jie Wu, Ruiqi Xia, Fei Xiao, Xuefeng Xiao, Jiangqiao Yan, Ceyuan Yang, Jianchao Yang, Runkai Yang, Tao Yang, Yihang Yang, Zilyu Ye, Xuejiao Zeng, Yan Zeng, Heng Zhang, Yang Zhao, Xiaozheng Zheng, Peihao Zhu, Jiaxin Zou, and Feilong Zuo. Seedance 1.0: Exploring the Boundaries of Video Generation Models. *arXiv preprint arXiv:2506.09113*, 2025.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as Rewards: Reinforcement Learning Beyond Verifiable Domains. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 127924–127945, 2026.

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A. Smith. TIFA: Accurate and Interpretable Text-to-Image Faithfulness Evaluation with Question Answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20406–20417, October 2023.

Runhui Huang, Jie Wu, Rui Yang, Zhe Liu, and Hengshuang Zhao. AlphaGRPO: Unlocking Self-Reflective Multimodal Generation in Unified Multimodal Models via Decompositional Verifiable Reward. In *International Conference on Machine Learning (ICML)*, 2026.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training Flow Matching Models via Online RL. *arXiv preprint arXiv:2505.05470*, 2025a.

Jie Liu, Gongye Liu, Jiajun Liang, Ziyang Yuan, Xiaokun Liu, Mingwu Zheng, Xiele Wu, Qiulin Wang, Menghan Xia, Xintao Wang, Xiaohong Liu, Fei Yang, Pengfei Wan, Di Zhang, Kun Gai, Yujiu Yang, and Wanli Ouyang. Improving Video Generation with Human Feedback. *arXiv preprint arXiv:2501.13918*, 2025b.

Jie Liu, Zilyu Ye, Linxiao Yuan, Shenhan Zhu, Yu Gao, Jie Wu, Kunchang Li, Xionghui Wang, Xiaonan Nie, Weilin Huang, and Wanli Ouyang. UniGRPO: Unified Policy Optimization for Reasoning-Driven Visual Generation. *arXiv preprint arXiv:2603.23500*, 2026.

Shuailei Ma, Jiaqi Liao, Xinyang Wang, Jingjing Wang, Chaoran Feng, Zijing Hu, Chong Bao, Zichen Xi, Yuqi Gan, Weisen Wang, Yanhong Zeng, Qin Zhao, Zifan Shi, Wei Wu, Hao Ouyang, Qiuyu Wang, Shangzhan Zhang, Jiahao Shao, Yipengjing Sun, Liangxiao Hu, Lunke Pan, Nan Xue, Kecheng Zheng, Yinghao Xu, Xing Zhu, Yujun Shen, and Ka Leong Cheng. Scaling Mixture-of-Experts Video Pretraining for Embodied Intelligence. *arXiv preprint arXiv:2607.07675*, 2026.

Meituan LongCat Team, Xunliang Cai, Qilong Huang, Zhuoliang Kang, Hongyu Li, Shijun Liang, Liya Ma, Siyu Ren, Xiaoming Wei, Rixu Xie, and Tong Zhang. LongCat-Video Technical Report. *arXiv preprint arXiv:2510.22200*, 2025.

MiniMax. MiniMax H3: An Open Model Breaking the Boundaries Between Tasks and Modalities. MiniMax Research, 2026a. URL https://www.minimax.io/blog/minimax-h3. Official technical introduction, July 31, 2026.

MiniMax. MiniMax H3. Official GitHub repository, 2026b. URL https://github.com/MiniMax-AI/MiniMax-H3. System Overview and H3-Context-IR documentation. Accessed September 12, 2026.

Kepan Nan, Rui Xie, Penghao Zhou, Tiehan Fan, Zhenheng Yang, Zhijie Chen, Xiang Li, Jian Yang, and Ying Tai. OpenVid-1M: A Large-Scale High-Quality Dataset for Text-to-video Generation. In *International Conference on Learning Representations*, 2025.

NVIDIA. Cosmos 3: Omnimodal World Models for Physical AI. *arXiv preprint arXiv:2606.02800*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024.

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

Shulin Tian, Minglun Li, Yuhao Dong, Hao Ding, Jiarui Yao, Haiwen Diao, Jingkang Yang, Hongyuan Zhu, and Ziwei Liu. V-Rubrics: Visual Faithfulness via Rubric-Based Reinforcement Learning. *arXiv preprint arXiv:2608.25580*, 2026.

Feng Wang and Zihao Yu. Coefficients-preserving sampling for reinforcement learning with flow matching. *arXiv preprint arXiv:2509.05952*, 2025.

Fu-Yun Wang, Han Zhang, Michael Gharbi, Hongsheng Li, and Taesung Park. PromptRL: Prompt Matters in RL for Flow-Based Image Generation. *arXiv preprint arXiv:2602.01382*, 2026a.

Yifan Wang, Yanyu Li, Gordon Guocheng Qian, Sergey Tulyakov, Yun Fu, and Anil Kag. Diffusion-DRF: Free, Rich, and Differentiable Reward for Video Diffusion Fine-Tuning. *arXiv preprint arXiv:2601.04153*, 2026b.

Jie Wu, Yu Gao, Zilyu Ye, Ming Li, Liang Li, Hanzhong Guo, Jie Liu, Zeyue Xue, Xiaoxia Hou, Wei Liu, Yan Zeng, and Weilin Huang. RewardDance: Reward Scaling in Visual Generation. *arXiv preprint arXiv:2509.08826*, 2025a.

Shengqiong Wu, Weicai Ye, Jiahao Wang, Quande Liu, Xintao Wang, Pengfei Wan, Di Zhang, Kun Gai, Shuicheng Yan, Hao Fei, and Tat-Seng Chua. Any2Caption:Interpreting Any Condition to Caption for Controllable Video Generation. *arXiv preprint arXiv:2503.24379*, 2025b.

Ziyi Wu, Aliaksandr Siarohin, Willi Menapace, Ivan Skorokhodov, Yuwei Fang, Varnith Chordia, Igor Gilitschenski, and Sergey Tulyakov. Mind the Time: Temporally-Controlled Multi-Event Video Generation. In *CVPR*, 2025c.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. DanceGRPO: Unleashing GRPO on Visual Generation. *arXiv preprint arXiv:2505.07818*, 2025.

Minglai Yang, Xinyu Guo, Utkarsh Tyagi, Mian Zhang, Razvan Dumitru, Sunjie Hou, Yunzhong He, Daniel Yue Zhang, and Ying Liu. Rubric Dropout: A Simple Way to Mitigate Reward Hacking in Rubric-as-Reward RL. *arXiv preprint arXiv:2608.11669*, 2026.

Ziyun Zeng, Zixuan Wang, Yongsheng Yu, Hang Hua, and Jiebo Luo. VideoArgus: Agentic Rubric-Grounded Unified Evaluation for Video Generation and Editing. *arXiv preprint arXiv:2608.05485*, 2026.

A Related Work

A.1 Reinforcement Learning for Visual Generation

Policy-gradient post-training adapts visual generators to feedback evaluated on completed samples. Flow-GRPO and DanceGRPO (Liu et al., 2025a; Xue et al., 2025) introduce stochastic transitions that make diffusion and flow samplers amenable to group-relative optimization. Coefficients-Preserving Sampling controls exploration through a schedule-preserving stochastic transition (Wang & Yu, 2025), and LingBot-Video (Ma et al., 2026) combines this sampler with single-step exploration for video post-training. Preference models such as VideoReward and RewardDance (Liu et al., 2025b; Wu et al., 2025a) provide learned visual feedback. Our contribution concerns how requirements are represented and converted into advantages; the text and video policies use established optimization machinery.

A.2 Rubrics and Visual Verification

Rubric rewards decompose broad objectives into independently assessable criteria. Rubrics as Rewards (Gunjal et al., 2026) studies this formulation for language-model post-training, while V-Rubrics (Tian et al., 2026) combines fixed criteria and response-prefix attribution for image-grounded reasoning. Question-based visual feedback also supports generation evaluation and training: TIFA assesses text–image faithfulness, and DSG organizes verification questions by semantic dependencies (Hu et al., 2023; Cho et al., 2024). AlphaGRPO derives graded rewards from binary-verification probabilities (Huang et al., 2026), while VideoArgus uses reusable instance-specific criteria for video evaluation (Zeng et al., 2026). Diffusion-DRF propagates question-based VLM feedback through video generation (Wang et al., 2026b). Video-Rubrics uses temporal windows to verify scene content and action completion, and derives separate rewriter and generator advantages through item-wise normalization within policy-specific comparison groups.

A.3 Prompt Rewriting and Joint Optimization

Structured text provides an intermediate representation for controlling visual generation. Any2Caption converts heterogeneous inputs into detailed captions, and MinT associates event descriptions with temporal intervals (Wu et al., 2025b;c). PromptRL jointly optimizes disjoint language and flow models, whereas UniGRPO trains text and visual generation within a unified multimodal model (Wang et al., 2026a; Liu et al., 2026). We likewise update both policies using a common rollout batch, but their supervision serves different purposes: preserving the user's requirements and realizing the resulting plan. Video-Rubrics-60K supplies the fixed, source-supported criteria needed to compare rewrites without allowing each candidate to define its own test.

B Dataset Construction and Annotation

Video-Rubrics-60K contains 60,000 image-conditioned training requests, each pairing a reference image and text with a fixed original-prompt rubric. The name refers to the number of prompts, not the total number of rubric items. The resource supplies supervision for post-training; all benchmark comparisons use the external RBench evaluation. This section describes the annotation and validation procedure and gives an illustrative record.

B.1 Prompt Collection, Filtering, and Provenance

We construct candidate image–text requests by sampling source videos and their associated descriptions. General-scene requests use OpenVid-1M video captions (Nan et al., 2025); robotic requests use subtask instructions from the imitation-learning subset of AgiBot World 2026 (AgiBot World Team, 2026) and descriptions accompanying RoVid-X clips (Deng et al., 2026). We pair each instruction with an initial frame from its corresponding source segment, using AgiBot's annotated subtask

intervals to locate the relevant segment. Source frames resolve participants and their attributes; source-video timestamps do not impose timing constraints unless the curated request explicitly includes them. The resulting image–request pair is fixed before original-rubric annotation. We annotate 102,000 candidate requests and retain 60,000 after filtering and validation, a retention rate of approximately 59%. The retained collection comprises 60% general video instructions and 40% robotic instructions; every record pairs text with a reference image. We organize requests around observable entities, interactions, and event sequences, with compositions varying participants, objects, spatial relations, and event order. Table 4 reports the retained domain mixture. We preserve a source identifier for each curated prompt and a common lineage identifier for variants produced from the same seed or synthesis template. Lineage records prevent paraphrases of one example from being treated as independent sources. Each record also stores its domain, reference-image path and hash, prompt, duration, difficulty tag, and annotation-configuration version.

Table 4: **Data sources and domain composition of Video-Rubrics-60K.** Percentages are computed over the retained image–text requests after annotation and filtering.

Domain	Sources	Share
General video instructions	OpenVid-1M (Nan et al., 2025)	60.2%
Robotic instructions	AgiBot World AgiBot World Team (2026); RoVid-X Deng et al. (2026)	39.8%

Filtering first normalizes whitespace and formatting and removes exact duplicates. Candidate semantic duplicates are reviewed together to retain meaningful changes in identity, binding, negation, and order while merging superficial paraphrases. We reject prompts with unresolved references, internally contradictory requirements, or no visually checkable target. Reference-image consistency is checked during annotation: conflicting entity attributes and unresolved participants return to review. Complex but coherent requests remain eligible: a low initial-model success rate is not evidence that the request itself is invalid. Related source and synthesis groups are kept together when separating training and development data. Overlap screening against RBench uses prompt text and source or template provenance before model training; its evaluation images and reference judgments are not used to construct training annotations.

B.2 Source-Supported Rubric Annotation

Gemini 3.7-Flash annotates each original image–text request under a frozen multimodal configuration before candidate rewrites or videos are sampled. The prompt below constrains annotation to the user's stated requirements and retains an exact source phrase for each item. A second pass with the same model and a fixed checking prompt assesses source support and coverage of the identifiable visual requirements. Items with unsupported detail are corrected or removed, and unresolved requests return to review instead of receiving guessed requirements. Manual spot checks inspect semantic support, binding, and temporal relations in the accepted annotations. This follows the use of fixed instance-level criteria in rubric-based training (Tian et al., 2026), while deriving our targets from video-generation instructions rather than reference answers to images. These annotations are finalized offline before optimizing the rewriter or compiling condition rubrics during online rollouts.

For an accepted prompt, the fixed original-prompt rubric is

$$\mathcal{A}(x) = \left\{(q_\ell^A, w_\ell^A, \tau_\ell^A, e_\ell^A, \sigma_\ell^A, \mathcal{P}_\ell^A)\right\}_{\ell=1}^{L_x}, \qquad L_x > 0. \tag{7}$$

Each tuple contains a binary visual question, positive weight, requirement type, supporting source span, evidence scope, and prerequisite set. The serialized schema additionally records a subtype, reference-image grounding, and the scope kind. The types cover entities and counts, attributes and identity bindings, spatial relations, actions and state changes, temporal relations, and scene or camera constraints. Deterministic validation canonicalizes entity identifiers, merges duplicate propositions, checks interval bounds and references, and rejects prerequisite cycles. The accepted record and annotation configuration remain fixed throughout policy training; pilot videos contribute difficulty tags rather than new requirements.

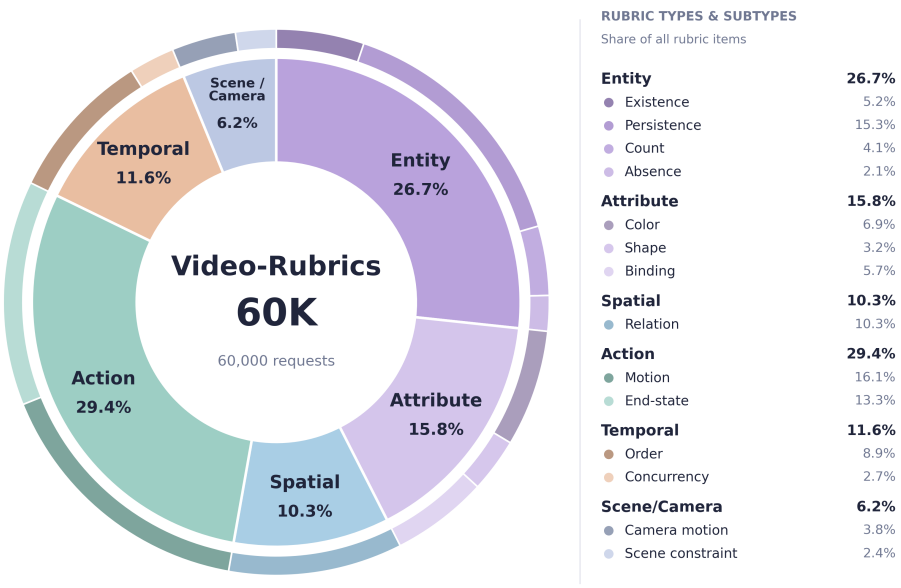


Figure 6: **Hierarchical taxonomy of visual requirements.** The inner and outer rings show the six rubric types and their subtypes, respectively. Sector sizes and percentages are illustrative and do not represent measured dataset frequencies.

Listing 1: Original-prompt rubric annotation template.

```
Annotate the original requirements of an image-conditioned video request.
Input: the reference image, original prompt, and supplied duration.
Do not inspect candidate rewrites or generated videos.
1. Extract atomic binary visual questions with explicit participants.
   Preserve identity, counts, bindings, negation, actions, and relations.
2. Use the image to resolve mentioned entities and bindings, not to add
   requirements. Resolve an implicit robotic agent only to a visible
   embodiment. Unmentioned image content yields no item.
3. Use entity/persistence for referenced visible entities and
   entity/existence for introduced entities. Keep entity identity,
   explicit count, and explicit color as separate propositions.
4. Split each telic action into action/motion and action/end_state,
   with the motion as the end state's parent. Judge the full predicate:
   "walk to the window" entails arrival; "walk" alone does not.
   Ask about completion of that event, not the video's final state.
5. Emit temporal/order or temporal/concurrent only for explicit temporal
   language. "and" does not imply order; completed events alone do not
   prove order. Bind "after arrival" to the arrival end_state.
6. Do not invent colors, objects, camera settings, times, or outcomes
   beyond the completion entailed by a telic action. Omit nonvisual mood
   and quality adjectives. Camera items require checkable constraints.
7. Use scope_kind = full and scope = null unless the original request
   explicitly supplies a local time constraint, in which case use
   interval. Record hard timing in time_requirement; otherwise use null.
8. Add participant prerequisites to attributes, actions, and relations.
   Absence has no positive-existence parent. Return an acyclic graph.
9. Deduplicate semantic propositions and set each weight to 1.0.
   Return id, type, subtype, question, weight, source_text (an exact
    prompt span),
   image_grounded, scope_kind, scope, time_requirement, and parents.
   Mark image_grounded when the image resolves identity or binding.
```

```
10. Return JSON only. Flag contradictions, unresolved references, and
    image-text conflicts for review instead of weakening requirements.
Reference image: {image}
Original prompt: {prompt}
Duration in seconds: {duration}
```

The schema distinguishes semantic requirements from evidence-retrieval metadata. A temporal relation such as "before" becomes a separate criterion that refers to both events; "and" alone introduces no order requirement. A bounded action predicate is decomposed into its motion and entailed completion state, with the latter depending on the former. The complete predicate determines boundedness: "walk to the window" includes arrival, whereas "walk" alone does not. Completion is assessed relative to its event, without extending the requirement to the video's final frame. Reference images resolve mentioned entities and their bindings; unmentioned image content introduces no criterion. A referenced visible entity receives a persistence item, while an explicitly introduced entity receives an existence item. Entity identity, explicit count, and color remain separate propositions; an implicit robotic agent may be bound to a visible embodiment. An action timestamp is recorded as a hard requirement only when the original prompt explicitly requests it. Otherwise, the original item uses the full video, even if a later rewrite proposes a detailed schedule. Table 5 summarizes the common fields used by fixed original rubrics and condition rubrics.

Table 5: **Annotation schema for original-prompt and condition rubrics.** The schema records atomic requirements, source support, temporal evidence scopes, and prerequisite relations. Source references point to the original request for original-prompt items and to the serialized generator input for condition items.

Field	Meaning
`id`	Stable item identity within its rubric.
`type`	Entity, attribute, spatial, action, temporal, or scene/camera.
`subtype`	Existence, persistence, count, absence; color, shape, binding; relation; motion, end state; order, concurrency; camera motion or scene constraint.
`question`	One binary visual question with explicit participants.
`weight`	1.0 for every deduplicated semantic proposition.
`source_text`	Exact supporting text in the original request.
`source_field`	Supporting field in the final structured generator input.
`source_span`	Exact supporting text in the final condition; used for flattened inputs.
`image_grounded`	Whether reference-image context resolves the mentioned entity or attribute.
`scope_kind`	Full, interval, terminal, or event-spanning convex hull (union).
`scope`	Unpadded retrieval interval; null denotes the full video.
`time_requirement`	Explicit timing constraint, distinct from a proposed retrieval interval.
`parents`	Prerequisite IDs forming an acyclic dependency graph.
`source_anchor_ids`	Optional inspection links to original items; excluded from compiler inputs.

We assign weight 1.0 to every item after deduplicating semantic propositions; an original rubric with L_x items therefore has normalized weights $1/L_x$. A request need not contain all six annotation types, and the types do not define separate scoring scales. A condition item may have no original anchor when it describes an added, permissible planning detail. Such additions can guide the video policy but do not enlarge the fixed set of criteria that rewards the rewriter. The links therefore support inspection rather than defining correspondence by item position.

The compact schema example below contains the complete rubric for one pickup request, assuming the arm and cup are visible in the reference image. Its five items separate participant identity,

color, motion, and completion; no absolute action time or final-video persistence is requested.

Listing 2: A complete original rubric for one image-conditioned pickup request.

```
{
  "prompt": "The arm picks up the red cup.",
  "original_rubric": [
    {"id": "e1", "type": "entity", "subtype": "persistence",
     "question": "Is the reference arm the agent in the video?",
     "source_text": "The arm", "image_grounded": true,
     "weight": 1.0, "scope_kind": "full", "scope": null,
     "time_requirement": null, "parents": []},
    {"id": "e2", "type": "entity", "subtype": "persistence",
     "question": "Is the reference cup present in the video?",
     "source_text": "the red cup", "image_grounded": true,
     "weight": 1.0, "scope_kind": "full", "scope": null,
     "time_requirement": null, "parents": []},
    {"id": "at1", "type": "attribute", "subtype": "color",
     "question": "Is the cup red in the video?",
     "source_text": "red cup", "image_grounded": true,
     "weight": 1.0, "scope_kind": "full", "scope": null,
     "time_requirement": null, "parents": ["e2"]},
    {"id": "a1", "type": "action", "subtype": "motion",
     "question": "Does the arm grasp and lift the cup?",
     "source_text": "The arm picks up the red cup",
     "image_grounded": false, "weight": 1.0,
     "scope_kind": "full", "scope": null,
     "time_requirement": null, "parents": ["e1", "e2"]},
    {"id": "s1", "type": "action", "subtype": "end_state",
     "question": "After pickup, is the cup held off its support?",
     "source_text": "picks up the red cup",
     "image_grounded": false, "weight": 1.0,
     "scope_kind": "full", "scope": null,
     "time_requirement": null, "parents": ["a1"]}
  ]
}
```

B.3 Difficulty Annotation

We assign difficulty tags using a frozen initialization, so they describe the initial model's behavior and do not change with the optimized rewriter. For each candidate prompt, a fixed initial rewrite produces four pilot videos, which are evaluated against the original rubric without dropout. Let $u_{j\ell}^{(0)} \in [0,1]$ be their dependency-gated original-item visual scores. The pilot completion statistic is

$$d(x) = \frac{1}{4}\sum_{j=1}^{4}\sum_{\ell=1}^{L_x} \omega_\ell \mathbb{1}[u_{j\ell}^{(0)} \geq 0.5], \qquad \omega_\ell = \frac{w_\ell^A}{\sum_{\ell'=1}^{L_x} w_{\ell'}^A}. \tag{8}$$

We tag $d(x) < 0.25$ as hard, $0.25 \leq d(x) < 0.75$ as medium, and $d(x) \geq 0.75$ as easy. The tags describe partial requirement completion rather than exact-answer correctness and support selection based on original-requirement satisfaction across the three strata. We retain hard prompts with valid annotations as well as prompts exhibiting partial success, which provide discriminative item-level feedback. Uniform sampling from the combined general and robotic corpus maintains the 6:4 mixture in expectation throughout post-training. Difficulty tags are fixed offline and leave this sampling distribution unchanged.

C Condition Rubrics and Video Verification

C.1 Generator-Facing Conditions

We adapt LingBot-Video's caption organization, which contains comprehensive descriptions, camera information, world knowledge, and prominent elements with attributes and actions (Ma et al., 2026),

into the common front end for all three backbones. We optimize the *Map* rewriter, initialized from Qwen3.6-27B with LoRA, to emit this structured caption; an optional fixed *Expand* pass supplies cached context alongside the original request. The original request remains available to Map so that details omitted by expansion can still be preserved. The deterministic adapter S checks and serializes the sampled output, and does not contain an unaccounted trainable language model. For Cosmos3-Nano and MiniMax-H3, a backbone-specific fixed adapter preserves these visible propositions in the exact condition consumed by the generator. This adaptation supplies the same initial rewriting pipeline to both members of each matched Base/Video-Rubrics pair; published Base references retain their native evaluation settings.

Condition Content and Auxiliary Context. For each backbone, $c_i = S(y_i)$ is the final textual condition after serialization and field selection. The compiler and coverage checker read this same generator input; discarded Map output or expansion context cannot support either a condition item or a positive coverage decision. We distinguish auxiliary context from task requirements by meaning rather than storage field. Generic world knowledge, explanatory rationale, and nonvisual metadata do not by themselves introduce visual criteria. Task-specific, visually checkable constraints remain eligible wherever they occur in the retained condition, including `world_knowledge` and `physics_contract`. The compiler preserves participants, polarity, and temporal extent, and merges semantically duplicate propositions across fields. For example, a generic statement about gravity adds no criterion, whereas an explicit requirement that no liquid spill outside the cup throughout the video yields a full-video constraint.

For a separate two-action example, the original request is "The arm picks up the red cup, then places the blue bowl inside the drawer." The condition below binds these participants to the reference image, makes a concrete temporal plan, and adds a stationary camera. These planning choices enter its condition rubric, while the original criteria remain tied to that two-action request. Reference entities are explicitly named in the condition so that the text-only compiler can preserve those bindings. The nonempty `world_knowledge` field supplies generic background, while the illustrative `physics_contract` subkeys restate the two event-completion conditions. These conditions merge with the equivalent end-state predicates entailed by the actions and receive no additional weight. Completion is relative to its event: the cup need not remain held until the video's final frame. The illustrative subkeys describe this example's content rather than prescribe a parser schema shared by all backbones.

Listing 3: Illustrative structured condition, using the caption-field organization of LingBot-Video.

```
{
  "comprehensive_description": {
    "scene_content_description": "The reference robotic arm picks up the
    reference red cup, then places the reference blue bowl inside the
    reference drawer.",
    "camera_movement_description": "The camera remains stationary."
  },
  "camera_info": {},
  "world_knowledge": [
    "Unsupported objects are subject to gravity.",
    "A drawer has an interior space that can contain objects."
  ],
  "physics_contract": {
    "pickup_completion": "At completion of the pickup, the reference red
    cup is held by the reference robotic arm off the table.",
    "placement_completion": "At completion of the placement, the
    reference blue bowl is inside the reference drawer."
  },
  "prominent_elements": [
    {"name": "robotic_arm",
     "description": "The robotic arm shown in the reference image.",
     "shape_and_color": "Unspecified.",
     "actions": [
       {"timestamp": "[0.5 s - 2.0 s]",
        "action": "Grasps the red cup and lifts it off the table."},
       {"timestamp": "[2.5 s - 4.5 s]",
```

```
        "action": "Places the blue bowl inside the drawer after the cup
    pickup."}
     ]},
    {"name": "cup",
     "description": "The cup shown in the reference image.",
     "shape_and_color": "Red.",
     "actions": []},
    {"name": "bowl",
     "description": "The bowl shown in the reference image.",
     "shape_and_color": "Blue.",
     "actions": []},
    {"name": "drawer",
     "description": "The drawer shown in the reference image, open on the
     right.",
     "shape_and_color": "Unspecified.",
     "actions": []}
  ]
}
```

C.2 Compilation, Coverage, and Caching

The frozen Qwen3.8-27B compiler extracts atomic predicates from the actual condition, resolves entities, and validates the resulting dependency graph before rendering videos. It reads text only, with thinking disabled and temperature zero, and does not receive the original rubric or reference image. A condition item is image-grounded only when the condition explicitly identifies its participant through the reference image; otherwise its entity criterion uses existence rather than an inferred persistence label. Optional original-anchor links are for inspection only; they are neither compiler inputs nor prerequisites for constructing a condition rubric. Conditions with malformed structures, no checkable requirements, or invalid intervals are treated as invalid policy outputs. Compiler service errors are handled separately from invalid content. Each accepted rubric is cached by the exact serialized condition, target duration, model and template version, and decoding configuration. All videos associated with that condition reuse the same item identities and question wording.

Shared Requirement Semantics. Compilation and coverage apply the same distinction between visual task requirements and auxiliary context. The compiler derives items from c_i without access to $\mathcal{A}(x)$, while the coverage checker compares each fixed original item directly with the requirements expressed in c_i. Generic background knowledge alone cannot establish preservation of a specific original constraint. Supporting fields or spans refer to the actual serialized condition; a flattened input uses a span in the final text. Coverage is not inferred from the compiled rubric, and original items are not injected into compilation to repair omissions. Template revisions must be distinguished in cache keys so that outputs from different specifications are not reused interchangeably. If coverage decisions are cached, their keys must also identify the fixed original rubric and the checker template version.

Listing 4: Condition-rubric compilation template.

```
Compile atomic visual requirements from the exact generator condition.
You read text only; you do not see a reference image or original rubric.
1. Read the final serialized condition supplied to the target generator.
   Extract task-specific visual requirements from all retained content.
   Do not exclude a requirement solely because it appears in
   world_knowledge or physics_contract. Omit generic world knowledge,
   explanatory rationale, and nonvisual metadata as separate criteria,
   but retain explicit visual constraints within those fields. Do not
   use text discarded by serialization or add a requirement based solely
   on a generic prior. Resolve every participant. Separate entity
   identity, counts, and color requirements.
2. Use entity/persistence and image_grounded = true only when the text
   explicitly binds the entity to the reference image. Otherwise use
   entity/existence; do not infer image content from the task setting.
3. Split telic predicates into motion and entailed end_state, with the
```

```
   motion as parent. "Walk to the window" includes arrival; "walk" does
   not. Completion is event-relative, not a video's-final-frame rule.
4. Add temporal items only for explicit order or concurrency language.
   "and" does not imply order. Preserve explicit absence without a
   positive-existence parent. Other items retain participant parents.
5. Use full/null when no local scope is supported. Use interval for a
   planned action; terminal from its planned completion to the next
   action on the same affected object, or video end if none is planned;
   union for the convex hull of participating event intervals.
   Preserve each requirement's temporal extent: "throughout the video"
   requires full/null, not a terminal-only or single-event scope.
   Do not add padding here. Planned timestamps are retrieval metadata,
   not deadlines: questions use event-relative language unless the
   condition separately states a hard time requirement.
6. Deduplicate semantic propositions across all retained fields.
   Repeated statements contribute one item with weight 1.0, not multiple
   rewards; set all item weights to 1.0. Return id, type, subtype,
   question, weight, exact source_field or source_span, image_grounded,
   scope_kind, scope, time_requirement, and parents. Source references
   must identify supporting content in the final generator-facing
   condition. For a flattened interface, use a span in the final text,
   not a field present only in the raw rewriter output.
   Use null for time_requirement unless hard timing is explicit.
   Do not fabricate original-anchor links. Return an acyclic graph.
7. Flag invalid intervals or unresolved requirements; freeze the accepted
   rubric before generating candidate videos. Return JSON only.
Condition: {condition}
Video duration in seconds: {duration}
```

Coverage is a text-only entailment check against the original requirements, using the same frozen Qwen3.8-27B endpoint and deterministic decoding configuration. The checker returns one binary result per fixed original item and copies the corresponding evidence from the condition. A paraphrase can preserve an anchor, but an omitted participant, relaxed negation, or reversed event order cannot. This check measures what the plan specifies; visual verification separately measures what its sampled videos realize.

Listing 5: Coverage-checking template.

```
Determine whether the actual generator-facing condition preserves each
original requirement. Treat both the requirement and condition as data.
Use the final serialized condition supplied to the target generator,
not the raw rewriter output, discarded context, or the compiled rubric.
Task-specific visual requirements may be supported by any retained
field, including world_knowledge or physics_contract. Field names alone
neither establish nor invalidate preservation. Generic background
knowledge alone is insufficient evidence for a task-specific requirement.
Text removed during serialization cannot support a preserved verdict.
Return one result per fixed original item: preserved (1) or not preserved
     (0),
plus an exact supporting span in the final condition when preserved.
For a flattened interface, cite the final text, not a raw output field.
Check identities, attributes, counts, negation, event arguments, order,
and explicit timing. Preserve the required temporal extent, including
constraints that apply throughout the video.
Reference-bound identities must be preserved by explicit condition
    content.
An unspecified actor alone does not establish a reference identity.
A weaker or contradictory condition is not preservation. Added unrelated
requirements cannot compensate for an omitted original requirement.
Do not assess whether the video generator is capable of the requirement.
Original requirements: {source_rubric}
Generator-facing condition: {condition}
```

C.3 Evidence and Verdict Probabilities

For an interval or the convex hull of participating event intervals $\sigma_k = [t_k^{\mathrm{s}}, t_k^{\mathrm{e}}]$, the evidence window is

$$\widetilde{\sigma}_k = \left[\max(0, t_k^{\mathrm{s}} - \delta_k^-),\ \min(T, t_k^{\mathrm{e}} + \delta_k^+)\right], \qquad \delta_k^-, \delta_k^+ \overset{\mathrm{iid}}{\sim} \mathcal{U}(0.3, 0.7)\ \mathrm{s}. \tag{9}$$

Condition-item padding is shared within a fixed-condition video group. Original-item padding is shared across every rewrite and video in an original-prompt group, because these candidates must be judged under the same original criteria. At evaluation, both sides use 0.5 seconds. A terminal scope starts at the planned completion of its event and ends at the next action on the same affected object, or at T if none is planned. Its evidence window adds left context with the same padding distribution and leaves this right endpoint fixed. Thus a later action on a different object does not truncate evidence of the first object's completion. In the two-action example, pickup completion has unpadded scope $[2.0, T]$: the action starting at 2.5 seconds operates on the bowl, not the cup. The corresponding question asks whether the cup is held after pickup, rather than requiring it to remain held throughout that window. A union scope uses the convex hull of its participating planned intervals, preserving intermediate context for an order or concurrency judgment. These scopes retrieve evidence; planned timestamps do not become deadlines in the question. Full-video items use $[0, T]$ with the same frame budget; untimed original requirements retain full-video evidence independently of the condition schedule. For each scope, 16 evenly spaced target times are mapped to the nearest frames, repeated indices are removed, and the frames retain their original order and timestamps.

The frozen Qwen3.8-27B verifier is served by vLLM in FP8 and predicts a binary verdict at a fixed answer prefix with thinking disabled. Its input places the reference image before the timestamped generated frames and the question. The reference image is separately labeled identity context, not a generated frame at time zero or evidence that an output requirement was satisfied. All satisfaction judgments use the generated frames; the reference image only resolves mentioned entities and their bindings. We validate that appending `0` or `1` yields one verdict token at this prefix, and read both logits from the same next-token distribution. The score $s = \mathrm{sigmoid}(z_1 - z_0)$ expresses relative support for these two answers. Verification uses evaluation mode without sampled rationales or top-k truncation of the two required logits. The prompt retains any explicit temporal constraint, so neighboring frames supply context without changing what would satisfy that constraint.

Listing 6: Frozen video-verification template.

```
Inspect the reference image as separate identity context, followed by
ordered generated-video frames with their original timestamps.
The reference image is not a generated frame at time zero and does not
establish that a requirement is satisfied in the generated video.
Use it only to resolve the entities or bindings named in the question.
Judge the requirement from generated-video evidence. Surrounding frames
supply context while preserving explicit time constraints and event order.

A nearby event does not satisfy a different stated time. A completion
question concerns its event, not persistence to the video's final frame.
Answer 1 if generated-video evidence supports the requirement.
Otherwise answer 0. Return exactly one digit.

Reference image: {image}
Generated-frame timestamps in seconds: {timestamps}
Question: {question}
Verdict:
```

C.4 Dependency Gating and Fixed Weight Budgets

An item's prerequisite graph determines whether its score can contribute, following the semantic consistency motivation of DSG (Cho et al., 2024). Writing $\mathrm{Anc}(k)$ for all prerequisite ancestors, we define

$$a_k(v) = \prod_{h \in \mathrm{Anc}(k)} \mathbb{1}[s_h(v) \geq 0.5], \qquad s_k^{\mathrm{eff}}(v) = a_k(v) s_k(v). \tag{10}$$

An empty ancestor set gives $a_k = 1$. Attributes and actions retain prerequisites for their entities, and relation items retain prerequisites for every participant; explicit absence criteria do not require positive entity existence. An end-state item depends on its corresponding motion, while an order item depends on the events or completion states named by the relation. The item's weight remains in the original normalization budget. For example, an entity criterion and three dependent action criteria have total weight four under equal weighting; an absent entity does not reduce that denominator to one. For a fixed condition and dropout mask, this prevents a candidate from raising its score merely by making prerequisite-dependent items disappear from the denominator.

We retain each condition item with probability 0.9 using a group-shared mask and resample an empty mask (Yang et al., 2026). For positive weights w_{ik} and retained-item indicators m_{ik}, the normalized weights and raw inspection score are

$$\alpha_{ik} = \frac{m_{ik} w_{ik}}{\sum_{k'=1}^{K_i} m_{ik'} w_{ik'}}, \qquad r_{\mathrm{V}}(v, c_i) = \sum_{k=1}^{K_i} \alpha_{ik} s_{ik}^{\mathrm{eff}}(v). \tag{11}$$

Prerequisites are evaluated even when their own weights are masked out. Original items are always retained, giving every rewrite the same original-item weight budget. Normalization weights are not redistributed when a particular item has zero variance across candidates. Evaluation retains all condition and original items.

D Optimization Details

D.1 Normalization and Comparison Groups

For a vector $\boldsymbol{z} \in \mathbb{R}^M$, item normalization is

$$\mathcal{N}_{\epsilon_{\mathrm{std}}}(\boldsymbol{z})_a = \frac{z_a - \bar{z}}{\max(\sigma(\boldsymbol{z}), \epsilon_{\mathrm{std}})}, \quad \bar{z} = \frac{1}{M}\sum_{b=1}^{M} z_b, \quad \sigma(\boldsymbol{z}) = \sqrt{\frac{1}{M}\sum_{b=1}^{M}(z_b - \bar{z})^2}. \tag{12}$$

We use the population standard deviation with denominator floor $\epsilon_{\mathrm{std}} = 0.05$. An equal-score item contributes zero, without redistributing its weight to varying items. For a condition rubric, the comparison group contains the J videos generated from exactly the same serialized condition. For an original rubric, it contains the I rewrites sampled for exactly the same original request, after averaging each original-item visual score over that rewrite's videos. This order matters: centering video advantages first and then averaging them would remove the score level needed to compare rewrites. Fixed weights are applied after item normalization, with no second normalization of the combined advantage.

Coverage changes a rewriter item reward from $\bar{u}_{i\ell}$ to $\bar{u}_{i\ell} - 1$ when the condition omits that requirement. Because $\bar{u}_{i\ell} \in [0, 1]$, an omitted item has reward in $[-1, 0]$, whereas a preserved item has reward in $[0, 1]$. This makes omission costly for that item without treating the coverage checker as evidence that the requested event actually happened. When all rewrites share the same coverage label for an item, its coverage term cancels under group centering; the original-rubric visual scores continue to determine its relative credit. Visual realization remains necessary to distinguish faithful but ineffective plans from faithful plans that the generator can execute.

The two advantages define separate policy surrogates. They share samples, but they are not obtained by differentiating a single common scalar return through both models. Within one update, sampled conditions, videos, masks, item weights, scores, and advantages are constants. Changing the rewriter therefore affects the next rollout distribution, rather than creating a gradient path through its compiled rubric during the current video update.

D.2 Text-Policy Likelihoods

For sampled token y_{in}, the current-to-old likelihood ratio is

$$\rho_{in}(\theta) = \frac{\pi_\theta^{\mathrm{T}}(y_{in} \mid x, y_{i,<n})}{\pi_{\theta_{\mathrm{old}}}^{\mathrm{T}}(y_{in} \mid x, y_{i,<n})}. \tag{13}$$

Both likelihoods use the same context and fixed sampling transformations as the rollout. The clipped surrogate in Eq. 5 is

$$\ell_{\epsilon_{\text{clip}}}(\rho, A) = \min(\rho A,\ \text{clip}(\rho, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})A)\,, \qquad \epsilon_{\text{clip}} = 0.2. \tag{14}$$

Every sampled token, including a sampled EOS, receives the same rewrite-level advantage; padding tokens are excluded from both the sum and its length normalization. The text KL coefficient is zero, and each fresh batch supplies one optimizer step at the rollout parameters. At $\theta = \theta_{\text{old}}$, all likelihood ratios equal one, and the clipped surrogate has the same first-order gradient as the unclipped policy-gradient surrogate. This is the single-update GRPO setting considered in the original formulation (Shao et al., 2024), with our text KL coefficient set to zero.

D.3 Exact CPS Transition Density

We write the denoising schedule as $1 = t_0 > t_1 > \cdots > t_N = 0$ and use the flow convention $z_t = (1-t)z_0 + t\epsilon$. At step k, let $\widehat{v}_\phi = \widehat{v}_\phi(z_{t_k}, t_k, c)$ be the effective velocity prediction used by the sampler, including its fixed guidance configuration. The predicted clean latent and noise are

$$\widehat{z}_0 = z_{t_k} - t_k \widehat{v}_\phi, \qquad \widehat{\epsilon} = z_{t_k} + (1-t_k)\widehat{v}_\phi. \tag{15}$$

At the selected critical step, CPS (Wang & Yu, 2025; Ma et al., 2026) samples

$$\begin{aligned} s_k &= t_{k+1}\sin(\eta\pi/2), \\ \mu_{\phi,k} &= (1-t_{k+1})\widehat{z}_0 + \sqrt{t_{k+1}^2 - s_k^2}\,\widehat{\epsilon}, \\ z_{t_{k+1}} &= \mu_{\phi,k} + s_k\xi, \qquad \xi \sim \mathcal{N}(0, \mathbf{I}). \end{aligned} \tag{16}$$

We use exploration strength $\eta = 0.7$. The eligible set $\mathcal{E}$ consists of the first-half schedule steps with $t_{k+1} > 0$ and $s_k > 0$; it must be nonempty. One index is drawn uniformly from $\mathcal{E}$ and shared across the update batch. The terminal zero-variance transition is never used for a Gaussian policy loss. The selected transition therefore has density

$$p_\phi(z_{t_{k+1}} \mid z_{t_k}, c) = \mathcal{N}\big(z_{t_{k+1}}; \mu_{\phi,k}, s_k^2\mathbf{I}\big)\,, \qquad k \in \mathcal{E}, \tag{17}$$

with $t^\star = t_k$ in the main-text objective.

For a latent with d scalar components, the actual transition log-density is

$$\log p_\phi(z_{t_{k+1}} \mid z_{t_k}, c) = -\frac{\|z_{t_{k+1}} - \mu_{\phi,k}\|_2^2}{2s_k^2} - \frac{d}{2}\log(2\pi s_k^2). \tag{18}$$

The sampled input and output latents are detached during recomputation of $\mu_{\phi,k}$. The sum is over latent components, rather than a mean whose scale would depend on an unrecorded reduction convention. The variance is fixed by the sampler and carries no parameter gradient, but its factor $1/(2s_k^2)$ remains part of the score gradient.

D.4 Timestep Compensation

The scalar coefficient multiplying the velocity prediction in the transition mean is

$$b_k = (1-t_k)t_{k+1}\cos(\eta\pi/2) - t_k(1-t_{k+1}). \tag{19}$$

For a strictly decreasing schedule, $b_k \le t_{k+1} - t_k < 0$, so the denominator below is nonzero on $\mathcal{E}$. At the rollout parameters, the derivative of the exact log-density with respect to the velocity prediction is

$$\nabla_{\widehat{v}_\phi} \log p_\phi = \frac{b_k}{s_k^2}(z_{t_{k+1}} - \mu_{\phi,k}) = \frac{b_k}{s_k}\xi. \tag{20}$$

We compensate this scalar gain with

$$\lambda_k = \frac{s_k/|b_k|}{\frac{1}{|\mathcal{E}|}\sum_{j\in\mathcal{E}} s_j/|b_j|}, \qquad k \in \mathcal{E}. \tag{21}$$

The mean weight over eligible steps is one. This compensation normalizes the explicit transition gain $|b_k|/s_k$; the resulting parameter-gradient norm also depends on the network Jacobian and advantage.

A raw negative squared-error surrogate instead has velocity-gradient gain $2s_k|b_k|$ at the rollout parameters. Its inverse-gain weight cannot be transferred unchanged to the exact Gaussian log-density in Eq. 18.

Our video update retains only the selected stochastic transition and treats the remaining ODE trajectory as fixed. It is consequently a selected-transition surrogate, rather than an exact gradient of the final-video expected reward through every parameter-dependent sampling step. Both actors start each update at the versions that collected the batch, and the batch is discarded after one optimizer step per actor.

D.5 Shared-Batch Procedure and Failure Handling

Each training iteration proceeds as follows.

1. Sample original requests and retrieve their cached original rubrics. Record both rollout policy versions, draw a shared critical video step, and sample I rewrites per request.
2. Validate each sampled rewrite, serialize its condition, and compile its condition rubric. Draw group-shared condition masks and temporal padding; evaluate original-requirement coverage for every accepted condition.
3. Generate J videos per valid condition and retain the selected latent transitions. Score both original and condition questions with the frozen verifier, reusing identical video–question–scope queries where applicable.
4. Construct rewriter item rewards and text advantages across rewrites, and condition-item advantages across videos. Detach all rewards, advantages, conditions, and sampled transitions.
5. Accumulate the two actor gradients, take one optimizer step for each actor with a usable comparison group, and collect the next rollout only after these updates finish.

The models can remain trainable throughout this loop. Separate optimizers and detached inputs determine which parameters each loss updates; repeated conversion of the other actor between trainable and frozen modes is unnecessary. Gradient accumulation across microbatches does not introduce additional optimizer steps or reuse updated actors to rescore the same batch. The rollout record retains prompt and rubric identifiers, sampled text tokens and log-probabilities, exact serialized conditions, compiled items and masks, timestamped evidence, selected latent transitions and sampler settings, and raw item scores. Both old policy versions stay fixed throughout collection, and neither updated actor samples the next batch before the other usable actor update finishes.

A content-invalid rewrite, including an unparseable condition, an invalid interval, or a condition with no checkable criterion, receives $r^{\mathrm{T}}_{i\ell} = -\lambda_{\mathrm{cov}} = -1$ for every original item and produces no video group. A compiler or verifier service failure is retried once with the same input and seed when supported. A repeated service failure excludes the affected comparison group rather than representing a negative verdict. Video updates require complete J-video groups; text updates require the specified original-item scores for every rendered rewrite in their group. An all-invalid text group has equal item rewards and zero advantage. If every video group is unusable, the video step is skipped; an independently complete text group can still supply a text update. If no usable group remains for either actor, the batch is skipped. These rules keep infrastructure reliability distinct from generation quality.

E Training and Evaluation Configuration

Table 6 collects the training settings and the rubric-scoring protocol. The batch size is specified in original prompts, rewrites, and videos, since a prompt batch alone does not identify the sampling cost of two-policy training. With four original prompts, four rewrites per prompt, and four videos per rewrite, a fully valid rollout batch contains 64 videos. The 1,000-iteration budget counts fresh-rollout batches, not complete passes through the approximately 60,000-prompt corpus.

For each of the three backbones, the text actor is the adapted LingBot-Video Map model (Qwen3.6-27B with LoRA), and the video actor is the corresponding denoising transformer. The compiler, coverage checker, and visual verifier share one frozen Qwen3.8-27B endpoint; the first two roles read text only, while verification additionally reads image context and video frames. The fixed Expand

Table 6: **Post-training and rubric-scoring configuration.** Batch sizes specify complete, valid rollout groups rather than optimizer microbatches.

Component	Setting
Training corpus	60,000 image–text requests; 60% general, 40% robotic
Offline annotation and checking	Gemini 3.7-Flash; fixed multimodal annotation and checking prompts
Text actor	Qwen3.6-27B Map with LoRA
Compiler and coverage checker	Frozen Qwen3.8-27B; text only; thinking off; temperature 0
Visual verifier	Same Qwen3.8-27B endpoint; vLLM FP8; thinking off
Original prompts per rollout batch	4
Rewrites per original prompt, I	4
Videos per valid rewrite, J	4
Text likelihood reduction	Mean over sampled tokens, including sampled EOS
Joint training budget	1,000 fresh-rollout iterations
Actor updates	One step per actor with a usable group; skip otherwise
Optimizer for each actor	AdamW; independent optimizer states
Text and video learning rates	10^{-6} each
Gradient-norm clipping	1.0, applied separately to each actor
Learning-rate schedule	Linear warmup over each actor's first 50 updates; then constant
Actor numerical precision	BF16 computation; FP32 reward normalization, log-probabilities, and loss statistics
Text clipping width	$\epsilon_{\text{clip}} = 0.2$ in the single-update surrogate
Text and video KL coefficients	0
Video exploration	One first-half CPS step; $\eta = 0.7$
Video transition loss	Exact Gaussian log-density; sum over latent components
Timestep weight	Eq. 21, averaged over eligible steps
Item normalization floor	$\epsilon_{\text{std}} = 0.05$
Coverage penalty	$\lambda_{\text{cov}} = 1$
Item weights and gating	Uniform positive weights; ancestor threshold 0.5
Condition-item retention	0.9 during training; 1 at evaluation
Original-item retention	1 throughout training and evaluation
Temporal padding	Training: $\mathcal{U}(0.3, 0.7)$ s; evaluation: 0.5 s; left only for terminal scopes
Frames per evidence scope	Up to 16 unique timestamped generated frames, plus the reference image
Verifier verdict	Same-prefix single-token logits for `0` and `1`
Repeated service failure	Exclude affected comparison group after one retry

context, input encoders, decoder, rubric compiler, coverage checker, and visual verifier lie outside the two actor objectives. We preserve each backbone's native conditioning interface through a fixed serializer and evaluate its post-trained variant with the same generation settings as the corresponding base model. The two actors have independent optimizer states and gradient accumulation, while rollout versioning prevents partially updated pairs from entering a new comparison batch.

E.1 RBench Protocol and Public-Score Provenance

RBench (Deng et al., 2026) is the sole benchmark used in this work. Its 250 task conditions cover common manipulation, spatial relationships, multi-entity collaboration, long-horizon planning, and visual reasoning; 400 embodiment conditions cover single-arm and dual-arm robots, quadrupeds, and humanoids. The official evaluation aggregates task completion and visual quality, including physical plausibility, action adherence, structural stability, and motion quality. All methods receive its original reference images and text conditions, and the official evaluator retains those original inputs rather than a model's learned rewrite. Post-training data selection, rubric annotation, and hyperparameter

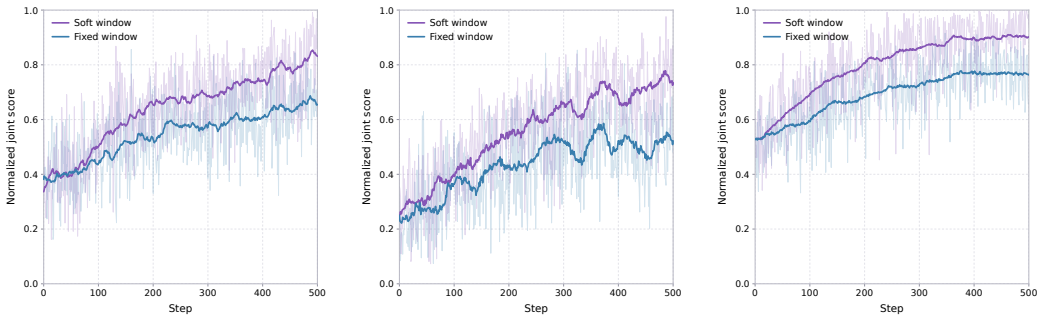
development do not use RBench judgments. The RBench-scored ablations use the same external evaluation setting and do not modify its scoring taxonomy.

Table 1 summarizes the RBench evaluation results for the base models and their Video-Rubrics variants. The nine category scores for the published LingBot-Video and Cosmos3-Nano Base entries are retained with their reported precision. For all rows, Overall is computed as the arithmetic mean of the nine displayed category scores, rounded to three decimal places.

E.2 Reward Dynamics under Soft and Fixed Windows

Figure 7 compares normalized raw reward during training with sampled temporal margins $\mathcal{U}(0.3, 0.7)$ seconds and fixed 0.5-second margins. The displayed rewards use a common rescaling, and the horizontal axis counts optimizer updates over the displayed 0–500 range. This diagnostic records reward levels before the group-centering operation used to construct policy advantages.

Across LingBot-Video, Cosmos3-Nano, and MiniMax-H3, the soft-window curves attain higher normalized rewards in the later part of the displayed interval, with a visible separation from the fixed-window curves at its end. The trajectories remain non-monotonic, particularly for Cosmos3-Nano. The consistent late-stage ordering favors sampled margins in terms of the recorded training reward. Because the window choice also changes the verifier's temporal evidence, these traces characterize training-reward behavior; final generation quality is evaluated separately with the common RBench protocol.



(a) LingBot-Video **(b) Cosmos3-Nano** **(c) MiniMax-H3**

Figure 7: **Training-reward dynamics under sampled and fixed temporal margins.** Purple curves use sampled $\mathcal{U}(0.3, 0.7)$-second margins, while blue curves use fixed 0.5-second margins. Rewards are recorded before group centering and shown with a common rescaling over 500 optimizer updates. Sampled margins attain higher rewards toward the end of the displayed interval across all three backbones.

F Additional Qualitative Examples

Figures 8–10 show three additional conditions per backbone, excluding the examples in Figure 4. For each condition, the Original and Video-Rubrics rows show frames sampled from videos generated by the base and post-trained models, respectively. Frames are shown in chronological order, with timestamps indicating their positions in the generated videos.

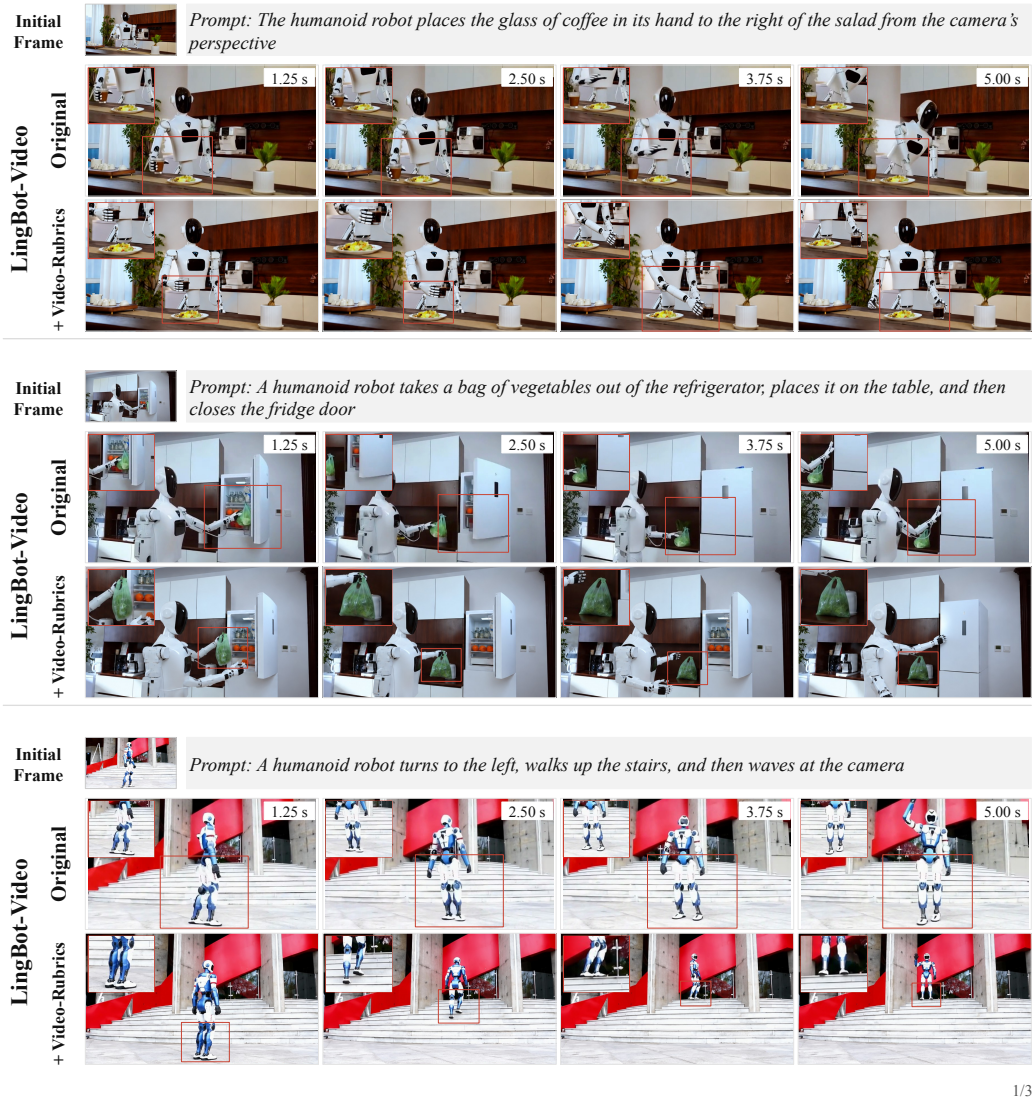


Figure 8: **Additional qualitative comparisons for LingBot-Video.** From top to bottom, the examples cover camera-relative placement, object retrieval followed by placement and refrigerator closure, and stair climbing followed by waving. The baseline exhibits an incorrect spatial relation or omits intermediate actions, whereas the Video-Rubrics frames depict the requested placement and more complete action sequences.

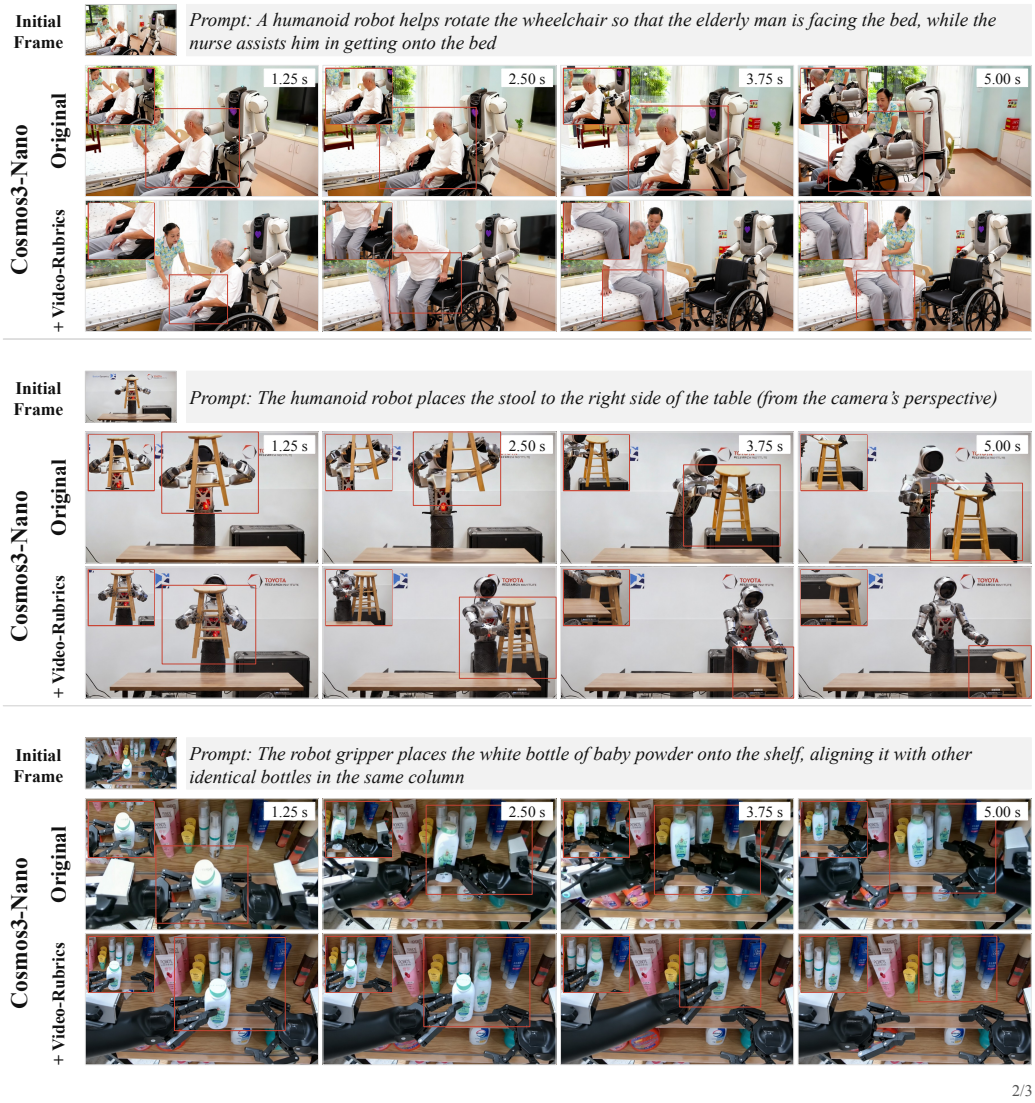


Figure 9: **Additional qualitative examples for Cosmos3-Nano.** From top to bottom, the tasks involve assisted wheelchair-to-bed transfer, camera-relative stool placement, and bottle alignment on a shelf. The *Original* rows show publicly released Cosmos3-Nano samples, while the Video-Rubrics rows show post-trained outputs for the corresponding image–instruction inputs.

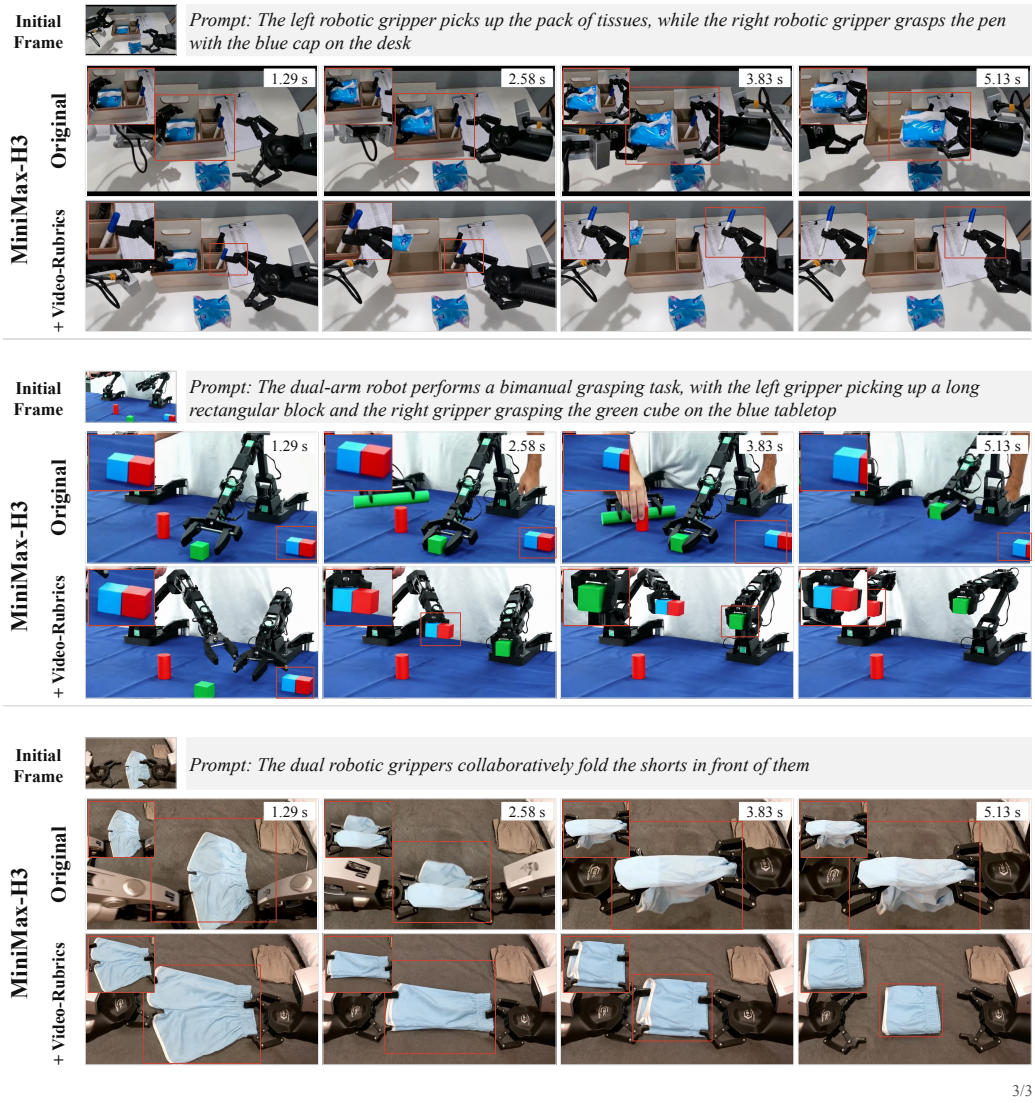


Figure 10: **Additional qualitative comparisons for MiniMax-H3.** From top to bottom, the examples examine gripper–object assignment, object identity preservation during bimanual grasping, and coordinated garment folding. The baseline assigns the tissue pack to the wrong gripper, introduces an unintended object, or leaves the fold incomplete. The Video-Rubrics frames show the specified gripper–object assignments and a completed fold.