CollabVR: Collaborative Video Reasoning with Vision-Language and Video Generation Models

Joowon Kim[1]* **Seungho Shin**[1]* **Eunho Yang**[1,2] **Joonhyung Park**[1,3]†
[1]KAIST [2]AITRICS [3]Kyung Hee University
{kjwispro, ssh9918, eunhoy, deepjoon}@kaist.ac.kr

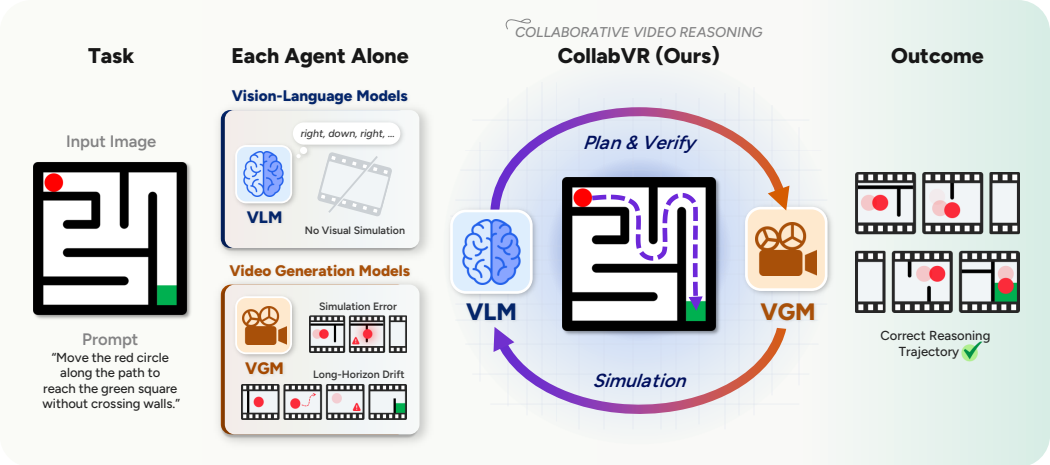


Figure 1: **VLM as planner, VGM as simulator.** A VLM is strong at reasoning but weak at visual simulation, while a VGM simulates short clips but lacks reasoning, causing long-horizon drift and mid-clip simulation errors. **CollabVR** couples them in a closed loop where the VLM plans progressively and diagnoses each generated clip, turning failures into correctable signals.

Abstract

Recent *Thinking with Video* approaches use Video Generation Models (VGMs) for visual reasoning by producing temporally coherent Chain-of-Frames as reasoning artifacts. Even strong VGMs, however, exhibit two recurring failure modes on goal-directed tasks: long-horizon drift on multi-step tasks and mid-clip simulation errors that compound. Both stem from the absence of explicit reasoning built upon the VGM's short-horizon visual prior, a role naturally filled by Vision-Language Models (VLMs), but where to place the VLM is non-trivial: upfront plans commit before any frame is generated and post-hoc critiques over whole videos intervene too late. We propose VLM-VGM Collaborative Video Reasoning (**CollabVR**), a closed-loop framework that couples the VLM with the VGM at step-level granularity: the VLM plans the immediate next action, inspects the clip the VGM generates, and folds the verifier's diagnosis directly into the next action prompt to repair detected failures. On Gen-ViRe and VBVR-Bench, **CollabVR** improves both open-source and closed-source VGMs over single-inference, Pass@k, and prior test-time scaling baselines at matched compute, with the largest gains on the hardest tasks. It also yields further improvements on top of a reasoning-fine-tuned VGM, indicating that step-level VLM supervision is orthogonal to and stackable with reasoning-oriented fine-tuning. We provide video samples and additional qualitative results at our project page: https://joow0n-kim.github.io/collabvr-project-page/.

*Equal contribution. †Corresponding author.

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

1 Introduction

Recent progress in visual reasoning has largely centered on the *Thinking with Images* paradigm, in which Vision-Language Models (VLMs) reason through visual intermediate steps rather than purely textual ones [26, 30, 11, 23, 15]. While promising, static images impose fundamental physical and cognitive limits on reasoning about dynamic processes. A single frame captures one instant in time, and is therefore inherently constrained in representing temporal evolution, causal sequences, or physical interactions that unfold over time.

To overcome these static limitations, a new paradigm has emerged: *Thinking with Video*. Departing from symbolic, text-based Chain-of-Thought, this line of work leverages Video Generation Models (VGMs) to produce a Chain-of-Frames: a form of generative visual reasoning in which the reasoning trajectory is realized as a temporally coherent video rather than a token sequence. dAlongside this shift, a growing suite of benchmarks now evaluates the zero-shot reasoning capabilities of VGMs [9, 20, 4, 21, 33, 40]. Recent studies report that VGMs exhibit promising reasoning behavior within short sequences, covering physical dynamics, spatial consistency, fine-grained visual detail, and geometric tracing, in some cases matching or surpassing VLMs on targeted video reasoning benchmarks [35, 31].

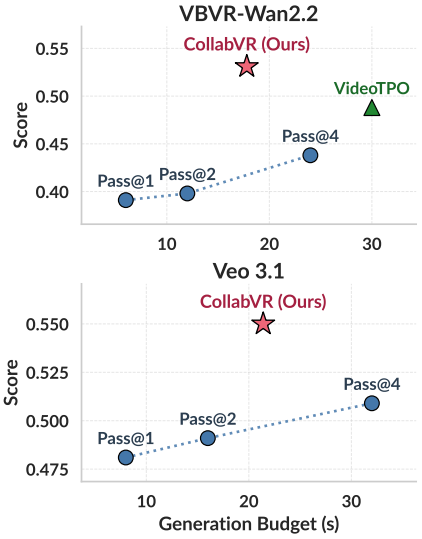


Figure 2: **Performance–Cost trade-off on Gen-ViRe [20].** Pass@k resampling plateaus quickly with cost and VideoTPO [4] trades extra budget for modest improvement, while **CollabVR** reaches markedly higher score at lower budget on both models [33, 7].

VLMs and VGMs nonetheless exhibit complementary strengths and weaknesses (Figure 1). VLMs excel at logical reasoning: they decompose complex problems, formulate multi-step plans, and draw abstract inferences with high fidelity, but their capacity to directly perceive and simulate the visual world remains limited. VGMs, conversely, are strong at perceiving visual detail, preserving physical coherence, and performing short-horizon simulation, yet weak at abstract logical reasoning and long-range causal consistency. This disparity in VGMs aligns with a broader pattern in recent evaluations: modern VGMs are trained to optimize perceptual quality, not task-level reasoning correctness [9, 3].

As a direct consequence, VGMs exhibit two recurring failure modes in goal-directed generative visual reasoning. (*i*) **Overloaded-prompt failure**: when a single prompt specifies a long-horizon task, the VGM collapses it into one short-horizon rollout that deviates from the intended trajectory, since it lacks the planning capacity to decompose the task into coherent sub-goals. (*ii*) **Execution failure**: even within a single short clip, the VGM commits localized errors mid-clip (e.g., an agent crossing a wall, an object losing its identity after an interaction, or a sub-action stopping before completion) that, once introduced, propagate through subsequent frames and contaminate the entire trajectory. Both failures can be traced to a common root cause: the absence of an explicit, corrigible reasoning process, on top of the VGM's strong yet short-horizon visual prior.

In light of this, the VGM needs a reasoning supervisor that can plan beyond a single short clip and verify what was just produced, capabilities at which VLMs are already strong. A straightforward way to leverage VLMs is to select the most perceptually plausible one among k samples, as in existing test-time scaling approaches [19, 10, 5]. On video reasoning tasks, however, valid outputs are task-specific and tightly defined, unlike general text-to-video which admits a wide range of realizations. More importantly, the correct trajectory often lies outside the generator's distribution and thus is difficult to reach by simply drawing more plausible samples (Figure 2). Therefore, in video reasoning, VLM supervision has to serve as a natural complement to the VGM's short-horizon visual prior, most effectively when applied to progressively construct the correct trajectory through the task.

We propose VLM-VGM Collaborative Video Reasoning (**CollabVR**), a closed-loop framework that couples the VLM with the VGM at step-level granularity. The VLM plans only the immediate next action, inspects the clip the VGM produces, and decides on the spot how to proceed, matching the intervention to the diagnosed state rather than applying one fixed recovery across whole videos. This

step-level coupling catches each failure at the moment it occurs, before it contaminates an entire trajectory. Unlike prior work that uses a VLM as a planner or verifier, the VLM in **CollabVR** chooses each action from the frames already generated and repairs a failure at the clip where it occurs, so a fixed generator shows more of its reasoning ability at inference time without any training. **CollabVR** consists of two tightly coupled modules, each targeting one of the failures above.

VLM-Driven Progressive Planning addresses *(i)* by letting the VLM adaptively decide the step count and plan only the immediate next action conditioned on previously generated frames, mitigating long-horizon drift without a fixed upfront decomposition; VLM-VGM Collaborative Reasoning addresses *(ii)* by having the VLM verify each clip, diagnose the failure, and revise the next action prompt to repair local errors before they compound. Although the VLM is not infallible, the step-level design contains any single error to one clip, and our human-annotated reliability benchmark (Section 4.4) confirms that VLM-predicted failure localization and step counts are reliable at the granularity the framework requires.

We summarize our contributions as follows.

- **Progressive Planning against long-horizon drift.** An adaptive planning module where the VLM decides the step count on the fly and emits only the immediate next action, conditioned on previously generated frames.
- **Collaborative Reasoning against execution failure.** A failure-aware intervention module where the VLM verifies each VGM clip and folds the diagnosed failure back into the next action prompt for repair.
- **Consistent gains at matched compute. CollabVR** improves both open- and closed-source VGMs over single-inference, Pass@k, and VideoTPO on Gen-ViRe and VBVR-Bench, with further gains on reasoning-fine-tuned VGMs [33]. A human-annotated benchmark confirms that VLM-predicted task complexity and failure localization align with expert judgments.

2 Related Work

2.1 Thinking with Video: Video Reasoning

Using generated or retrieved perceptual artifacts as intermediate reasoning steps originated in the *Thinking with Images* line of work on VLMs [30, 11, 23, 15, 29, 8, 26], where sketches, diagrams, and sub-images scaffold multi-step visual inference, but static images cannot capture dynamic processes or causal unfolding over time. The emergence of high-fidelity Video Generation Models such as Sora [1], Veo [7], and Wan [32], among many others [25, 44, 24, 42], gives rise to the *Thinking with Video* paradigm [9, 35, 31], in which a generated video itself serves as the reasoning artifact: a Chain-of-Frames whose temporal trajectory embodies the solution. A rapidly growing body of benchmarks [9, 31, 21, 40, 20, 4, 33, 17, 3] probes this regime, consistently demonstrating that modern VGMs excel at short-horizon visual simulation but remain weak at long-horizon planning, strict geometric and logical constraints, global state consistency, and process-level faithfulness, the diagnosis that motivates our framework. Recent benchmarks and studies also separate perceptual quality from physical and causal correctness [45, 36, 14, 38] and judge the generation process in addition to the final frame [17]. Reasoning therefore requires intermediate states that are checked and revised during generation, and **CollabVR** adds this structure at inference time.

2.2 Test-Time Scaling for Video Generation

Test-time scaling (TTS) for Large Language Models [28, 2] and diffusion models [22] substantially improves output quality with additional inference compute, and video-specific extensions [19, 10, 5, 16, 13] apply this to the temporal axis through frame-level search, evolutionary sampling, and self-refinement. These methods optimize *visual quality* rather than task correctness, but reasoning failures are systematic (wrong solution paths, skipped sub-goals, incorrect physical outcomes) and cannot be averaged out by sampling more, requiring instead diagnosis and repair by a VLM. Among reasoning-targeting approaches, VideoTPO [4] uses LLM critique to iteratively re-prompt the generator, but its single mechanism of whole-video prompt refinement leaves task-decomposition failures unaddressed. We instead operate at sub-action granularity, with the VLM verifying each clip and folding its diagnosis directly into the next action prompt for repair.

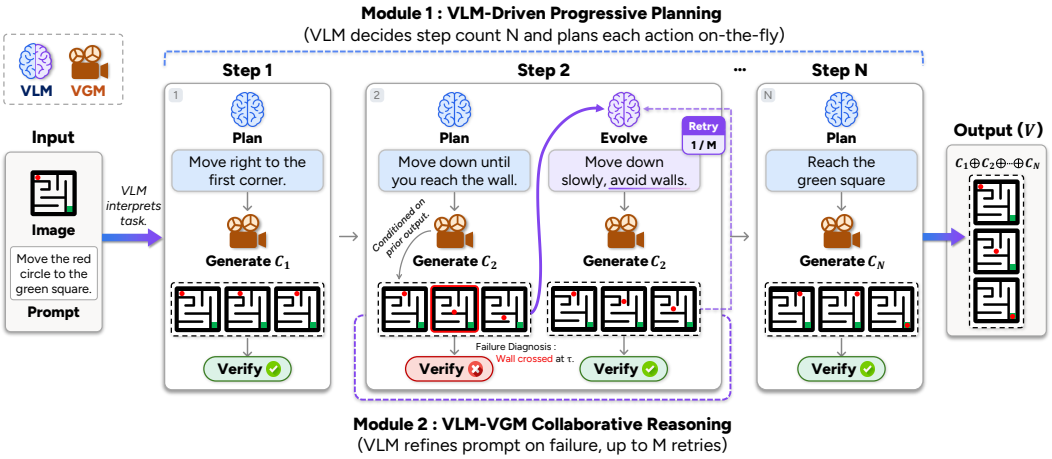


Figure 3: **Overall pipeline of CollabVR.** A persistent VLM plans one action at a time and, after observing each generated clip, decides whether to accept, re-generate, or re-plan. **Module 1** adaptively determines the step count, and **Module 2** verifies each clip and folds the verifier's diagnosis into the next action prompt to repair the failure.

2.3 Iterative Refinement and VLM-Guided Generation

Iterative refinement and LLM/VLM-guided generation cast an LLM as verifier or planner in a closed loop with the generator, originating in the image domain [37, 41] and extending to video [18, 43, 39, 34, 12], but these systems optimize visual or physical quality, treat the video as an indivisible unit, and lack a mechanism to diagnose specific failures or localize corrections. **CollabVR** closes these gaps in a single training-free loop that plans progressively, verifies and recovers at each step based on an explicit failure diagnosis, and works with any off-the-shelf VGM.

3 CollabVR: Closed-Loop Step-Level Video Reasoning

We present **CollabVR**, a closed-loop test-time framework that treats video reasoning as a construction problem: the correct trajectory is assembled stepwise through VLM planning and VGM execution, rather than sampled from the generator's output distribution. Figure 3 illustrates the overall pipeline, whose two core modules target the two failure modes identified in Section 1. VLM-Driven Progressive Planning (Section 3.2) addresses overloaded-prompt failure by letting the VLM plan adaptively, one step at a time. VLM-VGM Collaborative Reasoning (Section 3.3) addresses execution failure by letting the VLM verify each clip and revise the action prompt with the diagnosis.

The reliability of this VLM-as-supervisor design (whether plan-depth, verification, and prompt evolution genuinely align with human judgment rather than amplifying VLM hallucinations) is empirically validated in Section 4.4 on a dedicated human-annotated benchmark.

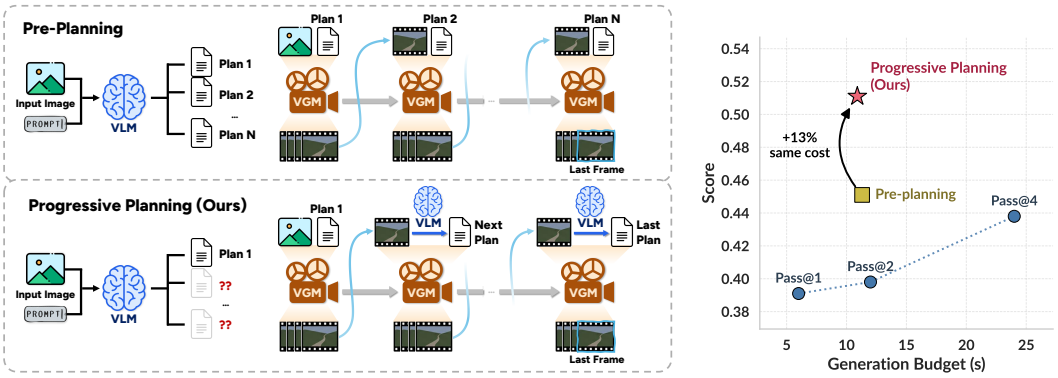
Algorithm 1 `CollabVR` $(I_0, q, N_{\max}, M)$.

Require: input image I_0, task prompt q; max planning steps $N_{\max}$; per-step attempt budget M
Ensure: generated video V

$\mathcal{H} \leftarrow \emptyset$, $f \leftarrow I_0$
for $t = 1, \ldots, N_{\max}$ **do**
  $a_t \leftarrow \pi_{\text{plan}}(I_0, q, \mathcal{H})$ {(Section 3.2)}
  for $j = 1, \ldots, M$ **do**
    $c_t \leftarrow g(f, a_t)$
    $(v, d) \leftarrow \pi_{\text{verify}}(I_0, q, \mathcal{H}, c_t)$ {(Section 3.3)}
    if $v =$ `accept` **then**
      $\mathcal{H} \leftarrow \mathcal{H} \cup \{c_t\}$
      $f \leftarrow$ last frame of c_t
      if task complete **then**
        return $V = c_1 \oplus \cdots \oplus c_t$
      end if
      break
    else
      $a_t \leftarrow \text{evolve}(a_t, d)$ {prompt evolution}
    end if
  end for
end for
return $V = c_1 \oplus \cdots \oplus c_{|\mathcal{H}|}$

3.1 Problem Formulation

A video reasoning task is specified by an input image I_0 and a task prompt q that describes the desired visual process or transformation, and the goal is to produce a video V whose trajectory realizes the



(a) Pre-planning vs. Progressive planning pipeline.

(b) Performance–cost on Gen-ViRe.

Figure 4: **Pre-planning vs. Progressive planning** on Gen-ViRe with VBVR-Wan2.2 (Module 1 only). Progressive planning achieves a $+13\%$ relative gain over pre-planning at matched cost.

reasoning demanded by (I_0, q). Our framework has two actors: a VLM-based planner/verifier π (queried in two roles, π_{plan} and π_{verify}) and an image-to-video generator g that maps a conditioning frame f and an action prompt a_t to a short clip c_t. Throughout the loop we maintain f as the latest conditioning frame (initially I_0) and $\mathcal{H}$ as the history of accepted clips. The full closed loop is given in Algorithm 1, and the output is the concatenation of accepted clips $V = c_1 \oplus \cdots \oplus c_N$; the rest of this section motivates its two design choices, progressive planning (Section 3.2) and collaborative reasoning (Section 3.3).

3.2 VLM-Driven Progressive Planning

A naive extension of Chain-of-Thought to video is *pre-planning* [12] (Figure 4a, top): the VLM decomposes q into N milestone prompts upfront, and the VGM sequentially generates a clip per milestone. This reduces drift on long-horizon tasks but commits the plan before any VGM output exists, so it cannot adapt to the realized generation, and N itself is difficult to fix from the prompt alone. We therefore adopt *progressive planning* (Figure 4a, bottom; line 4 of Algorithm 1): the VLM plans only the immediate next action and inspects the realized clip before deciding whether to continue, so both subsequent steps and N adapt to what the generator actually produces, with N capped at a hyperparameter $N_{\max}$. At matched generation budgets, this yields a substantially better performance–cost trade-off than pre-planning (Figure 4b).

3.3 VLM-VGM Collaborative Reasoning

For each generated clip c_t, the VLM verifier π_{verify} produces a structured judgment (v, d), where $v \in \{\text{accept}, \text{reject}\}$ and d packages a textual reason and an actionable suggestion for repair (line 6 of Algorithm 1). The verifier judges whether the planned action was executed, flagging wrong direction, wrong target, or scene collapse but not partial-but-correct progress, which is the planner's concern. This single-clip judgment stays within VLMs' strong perception regime, so d is specific enough to be actionable. The simplest recovery folds d's suggestion (e.g., *"circle the red square, not the yellow"*) back into the action prompt, $a_t \leftarrow \text{evolve}(a_t, d)$, and re-samples the VGM with the plan fixed, reusing the verifier's output without an extra VLM call.

4 Experiments

We evaluate **CollabVR** on Gen-ViRe and VBVR-Bench, two complementary video reasoning benchmarks. The section presents the experimental setup (Section 4.1), main accuracy comparisons under matched compute (Section 4.2), per-module ablations (Section 4.3), and an analysis covering per-category module effectiveness and the human-annotated reliability of the VLM-as-supervisor design (Section 4.4). Complementary evidence in the appendix includes a blind user study (Appendix B.1) and a cost decomposition validating the matched-compute framing (Appendix D.3).

Table 1: Benchmarking results on Gen-ViRe. All scores are 3-run VLM evaluation averages (Gemini 2.5 Pro) for stability. **VGM Cost** is the average total VGM generation seconds per sample (all steps and re-generations included). Higher is better. **Bold**: best within each model; Underlined: second best.

Method	VGM Cost (s)	Avg.	Category					
			Abst.	Algo.	Analog.	Perc.	Plan.	Spat.
Open-source Video Models								
VBVR-Wan2.2	6.0	0.391	0.479	0.415	0.250	0.261	0.554	0.387
VBVR-Wan2.2 + Pass@2	12.0	0.398	0.576	0.437	0.278	0.257	0.481	0.357
VBVR-Wan2.2 + Pass@4	24.0	0.438	**0.622**	0.418	0.250	0.275	0.604	0.462
VBVR-Wan2.2 + VideoTPO [4]	30.0	0.488	0.535	0.443	**0.417**	0.313	0.671	**0.552**
VBVR-Wan2.2 + **CollabVR**	17.8	**0.531**	0.569	**0.606**	0.333	**0.367**	**0.821**	0.488
Closed-source Video Models								
Veo 3.1	8.0	0.481	0.420	0.512	0.361	0.274	0.744	0.573
Veo 3.1 + Pass@2	16.0	0.491	**0.458**	0.587	0.389	0.242	0.721	0.571
Veo 3.1 + Pass@4	32.0	0.509	0.425	0.573	0.417	0.296	0.726	0.646
Veo 3.1 + **CollabVR**	21.4	**0.550**	0.434	**0.641**	**0.472**	**0.325**	**0.768**	**0.657**

Table 2: Benchmarking results on VBVR-Bench. Overall In-Domain (ID) and Out-of-Domain (OOD) scores are reported alongside category-wise performance. Higher is better. **Bold**: best in group; Underlined: second best.

Models	VGM Cost (s)	Overall	In-Domain by Category						Out-of-Domain by Category					
			Avg.	Abst.	Know.	Perc.	Spat.	Trans.	Avg.	Abst.	Know.	Perc.	Spat.	Trans.
VBVR-Wan2.2	3.70	0.671	0.762	0.701	0.746	0.802	0.793	0.803	0.577	0.674	**0.674**	0.503	0.528	0.633
VBVR-Wan2.2 + Pass@2	7.40	0.694	0.783	0.791	0.742	0.795	0.774	0.812	0.602	0.728	0.617	0.494	0.532	0.701
VBVR-Wan2.2 + Pass@4	14.80	0.707	0.789	0.751	0.734	**0.826**	0.805	0.841	0.622	0.785	0.660	0.535	0.577	0.683
VBVR-Wan2.2 + VideoTPO [4]	11.10	0.650	0.717	0.723	0.698	0.641	0.744	0.816	0.582	0.767	0.619	0.513	0.540	0.572
VBVR-Wan2.2 + **CollabVR**	10.91	**0.757**	**0.819**	**0.828**	**0.784**	0.805	**0.828**	**0.852**	**0.696**	**0.884**	0.634	**0.641**	**0.608**	**0.720**
Cosmos-Predict2.5	3.70	0.308	0.312	0.272	0.327	0.355	0.227	0.390	0.304	0.368	0.169	0.309	0.377	0.274
Cosmos-Predict2.5 + **CollabVR**	10.91	**0.403**	**0.406**	**0.404**	**0.431**	**0.411**	**0.301**	**0.482**	**0.400**	**0.481**	**0.286**	**0.400**	**0.471**	**0.346**

4.1 Implementation Details

Benchmarks and evaluation. We evaluate **CollabVR** on two complementary video-reasoning benchmarks. **Gen-ViRe** [20] contains 72 samples across 6 categories and uses a rubric-based VLM judge (Gemini 2.5 Pro) that scores each generated video against per-task criteria targeting task correctness rather than visual quality.[2] **VBVR-Bench** [33] focuses on synthetic visual reasoning tasks with controlled ground-truth references. It consists of a 500-sample test set spanning 5 reasoning categories with In-Domain and Out-of-Domain splits, and adopts a rule-based, judge-free protocol for comparing generated videos against ground-truth references.

Video generation models. We apply **CollabVR** to two main VGMs: the closed-source API model[3] **Veo 3.1** [7], and **VBVR-Wan2.2** [32, 33], a 14B open-source image-to-video model that is the strongest open baseline on VBVR-Bench. We additionally report on **Cosmos-Predict-2.5** [25] as a second open-source VGM.

Baselines. We compare **CollabVR** against three baselines: (i) *Single Inference*, one-shot generation from (I_0, q); (ii) *Pass@k*, $k \in \{2, 4\}$ independent generations from which a VLM selects the one that best achieves the task; and (iii) *VideoTPO* [4], a TTS-for-video-reasoning baseline that iteratively rewrites prompts based on full-video critiques. Both (ii) and (iii) use Gemini 2.5 Pro, the same model used as our step-level verifier; full configuration is in Appendix A.1. Since VLM compute is negligible relative to VGM compute, we report Cost throughout as the total seconds of video generated by the VGM per sample.

Our framework. We use Gemini 2.5 Pro as the default VLM for both planning and verification, and evaluate alternative VLMs in Section 4.3. Unless stated otherwise, we set $N_{\max}=3$ and the per-step attempt budget to $M=3$; the resulting total generation budget is comparable to Pass@2–Pass@4 for a single VGM.

[2] We report Gen-ViRe scores averaged over three runs to account for judge stochasticity.

[3] We exclude Sora 2 from the main results due to insufficient first-frame fidelity for step-by-step clip concatenation, and defer its analysis to Appendix A.4.

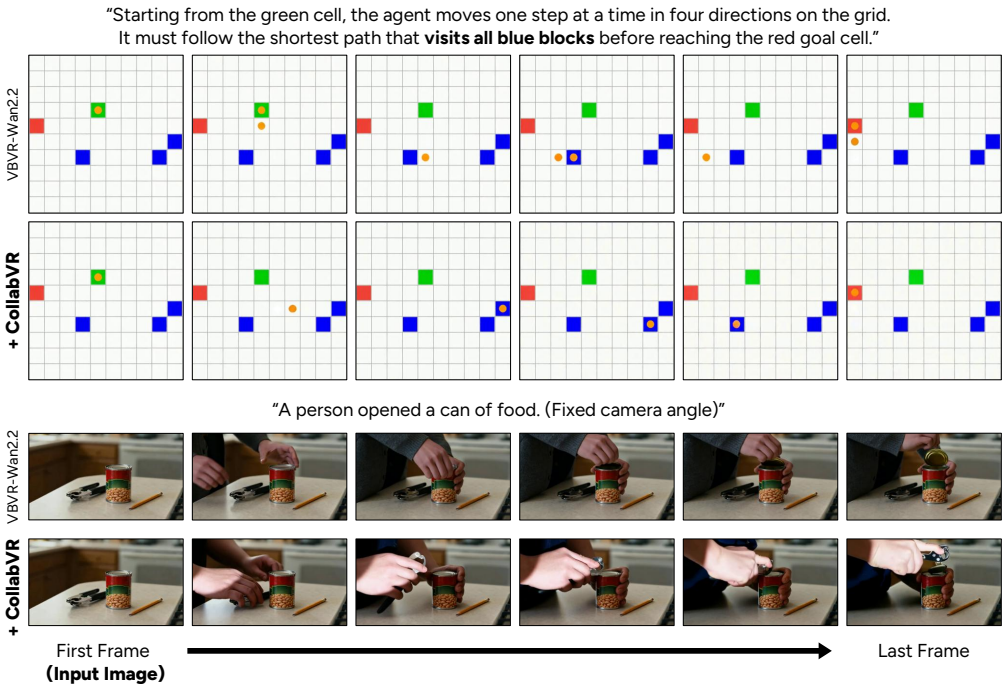


Figure 5: Qualitative comparison on various visual reasoning tasks from Gen-ViRe and VBVR-Bench. We compare VBVR-Wan2.2 with **CollabVR** on diverse tasks that require step-by-step reasoning.

4.2 Main Results

Tables 1 and 2 report our main results on Gen-ViRe and VBVR-Bench. On Gen-ViRe, **CollabVR** delivers consistent improvements over single-inference on both the open-source VBVR-Wan2.2 (Pass@1 0.391 $\rightarrow$ **0.531**) and the closed-source Veo 3.1 (Pass@1 0.481 $\rightarrow$ **0.550**) (Table 1), with the largest margins over baselines on Planning and Algorithmic categories where complex long-horizon reasoning is required.

On VBVR-Bench, **CollabVR** consistently surpasses baselines on both open-source models, VBVR-Wan2.2 and Cosmos-Predict-2.5 (Table 2), with gains most pronounced on categories requiring multi-step spatial or transformation reasoning. Compared to VideoTPO and Pass@k, **CollabVR** achieves higher accuracy at lower per-sample generation cost, supporting our claim that adaptive progressive planning coupled with failure-aware recovery is a more effective test-time scaling axis than full-video resampling. Notably, **CollabVR** also yields further gains on top of VBVR-Wan2.2, a VGM already fine-tuned on reasoning data, demonstrating that test-time reasoning supervision is orthogonal to and stacks on reasoning-oriented fine-tuning.

Qualitative results. Figure 5 shows representative examples where **CollabVR** contributes over the baseline. On a long-horizon task (top), VBVR-Wan2.2 alone cannot solve in a single shot, whereas **CollabVR** succeeds by decomposing the plan to visit the goal cells starting from the right. The framework also extends to real-world scenarios (bottom): for *"opened a can of food"*, VBVR-Wan2.2 alone bypasses the can opener and pries the lid by hand, whereas **CollabVR**'s planner emits explicit tool-use

Table 3: Per-module ablation on Gen-ViRe and VBVR-Bench. Δ: gain over the baseline (no module).

M1	M2	Cost (s)	Overall	Δ
Gen-ViRe				
✗	✗	6.0	0.391	–
✓	✗	10.9	0.511	+0.120
✗	✓	9.9	0.436	+0.045
✓	✓	17.8	**0.531**	**+0.140**
VBVR-Bench				
✗	✗	3.70	0.671	–
✓	✗	6.19	0.706	+0.035
✗	✓	6.03	0.734	+0.063
✓	✓	10.91	**0.757**	**+0.086**

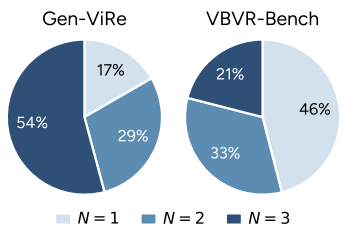


Figure 6: Human-annotated distribution of step counts N for the benchmarks.

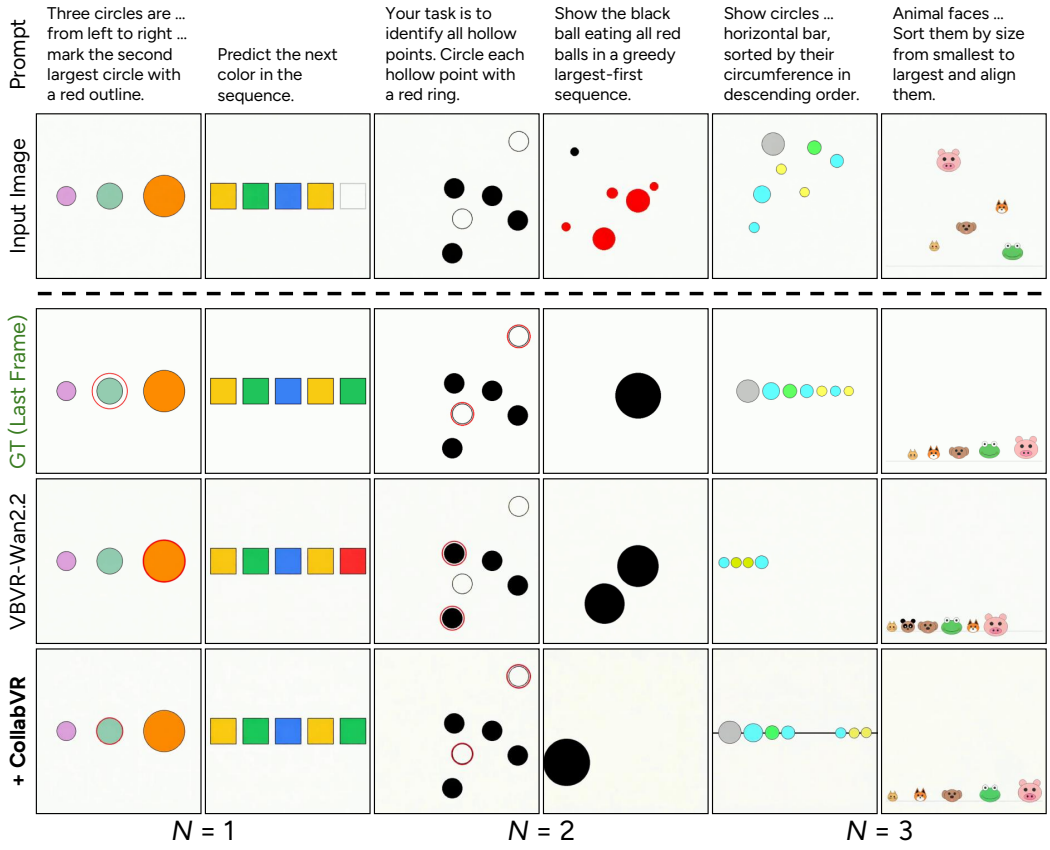


Figure 7: **Qualitative gains scale with planner-predicted step count N on VBVR-Bench.** For each task we show the input image, GT last frame, single-shot VBVR-Wan2.2 output, and VBVR-Wan2.2+**CollabVR** output, with two representative tasks grouped under each of N=1, 2, and 3.

sub-actions and the VGM produces a faithful execution. Beyond automated metrics, we further confirm via a user study that human annotators prefer **CollabVR**'s outputs (73.8%) over Pass@4 (19.7%) and Pass@1 (6.5%) on a blind side-by-side comparison (Appendix B.1).

Figure 7 groups VBVR-Bench examples by the planner's predicted step count N. For atomic tasks (N=1), both methods produce similar outputs and **CollabVR** only removes incidental errors such as a wrong next color. As N grows, single-shot failures become structural. On *"sort them by size from smallest to largest and align them"*, VBVR-Wan2.2 substitutes and misorders the animal faces, whereas **CollabVR** reproduces the GT arrangement.

4.3 Ablation Study

Module 1 vs. Module 2 vs. Combined. We compare three configurations: (M1) progressive planning only (no verification or recovery), (M2) verification and failure-aware recovery only ($N_{\max}$=1, no progressive planning), and (M1+M2) the full pipeline. Figure 6 shows the human-annotated distribution of step counts N for the two benchmarks; this distribution reflects each benchmark's mix of task complexity and type, with Gen-ViRe dominated by multi-step tasks and VBVR-Bench dominated by single-step tasks. Table 3 reports the corresponding per-module gains: on Gen-ViRe, M1's progressive planning is the larger contributor (+0.120 vs. +0.045 for M2), whereas on VBVR-Bench, M2 is the larger contributor (+0.063 vs. +0.035 for M1). On Gen-ViRe, **CollabVR** frequently improves performance by adaptively decomposing complex tasks into sub-steps the VGM can satisfy individually. On VBVR-Bench, M2 corrects single-clip execution failures without unnecessarily splitting tasks that are already solvable in a single action. That the dominant module shifts with the benchmark's N profile indicates the framework adapts to task character rather than relying on a single fixed mechanism. M1 also presupposes M2. Without verification, each planned step has to

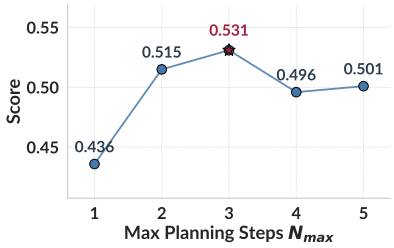


Figure 8: Effect of maximum planning steps $N_{\max}$ on Gen-ViRe.

Table 4: Comparison of test-time scaling methods and VLM choices across benchmarks.

Method	VLM	Gen-ViRe	VBVR-Bench
Pass@1 (baseline)	–	0.391	0.671
Pass@2	Gemini 2.5 Pro	0.398	0.694
Pass@4	Gemini 2.5 Pro	0.438	0.707
VideoTPO [4]	Gemini 2.5 Pro	0.488	0.650
CollabVR	Qwen3.5-9B	0.514	0.710
	Qwen3.5-27B	0.510	0.717
	Gemini 2.5 Pro	**0.531**	**0.757**

succeed in one shot as written. The reasoning-tuned VBVR-Wan2.2 manages this but Veo 3.1 does not, so M1 alone falls below single inference on Veo 3.1 ($0.481 \rightarrow 0.446$) while the full pipeline reaches 0.550 (Appendix B.5).

Effect of step count $N_{\max}$. We sweep the planning step count $N_{\max} \in \{1, \ldots, 5\}$ on Gen-ViRe with VBVR-Wan2.2 and observe a non-monotonic relationship: scores rise as $N_{\max}$ increases up to the level required by the task, then plateau or degrade as further splitting introduces step-boundary artifacts on already-simple sub-actions. This empirically motivates the adaptive N selection in Section 3.2.

VLM choice. We replace the default Gemini 2.5 Pro planner/verifier with open-source alternatives (Qwen3.5-27B, Qwen3.5-9B [27]) and report the final task accuracy on each benchmark (Table 4). Performance degrades gracefully with weaker VLMs, and notably even the smallest model we test, Qwen3.5-9B paired with **CollabVR**, surpasses every Pass@k and VideoTPO baseline that uses the proprietary Gemini 2.5 Pro on both benchmarks, confirming that the framework is not tied to a single proprietary model. We further examine in Section 4.4 how these end-to-end gaps are consistent with each VLM's per-axis supervisor quality (plan-depth, verification, evolution) on the human-annotated benchmark.

4.4 Analysis

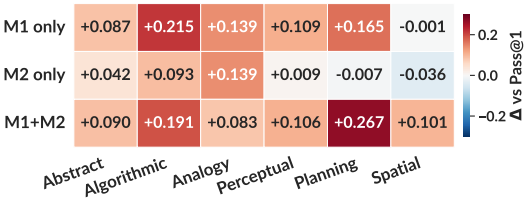


Figure 9: **Per-category Δ over Pass@1 on Gen-ViRe (VBVR-Wan2.2).** Module configurations follow Section 4.3.

Category-wise module effectiveness and limitation. CollabVR does not behave uniformly across reasoning types. Section 4.3 attributed module dominance to a benchmark's N profile. Here we examine the finer-grained role of each reasoning category (Figure 9). For instance, a Planning prompt such as *"use the kettle and teabag to make a cup of tea in the mug"* packs a chain of distinct physical actions into a deceptively short instruction; a single-shot VGM compresses these into one ambiguous clip and typically resolves only the first action, whereas the progressive planner exposes the chain as separate sub-goals that the VGM can satisfy individually (M1 alone $+0.165$). Conversely, an Analogy prompt such as *"generate the missing object in the lower right region and solve the visual analogy"* is a single atomic transformation: verifier-driven re-sampling alone is sufficient (M2 alone $+0.139$). Crucially, the full M1+M2 pipeline yields a positive gain in every category ($+0.083$ to $+0.267$), and only the combination meaningfully improves long-horizon Spatial tasks, indicating decomposition and recovery are complementary

Yet the framework cannot manufacture capabilities the VGM lacks: the smallest gains concentrate on categories whose target transformations are symbolic rather than physical (Analogy $\Delta +0.083$, Abstract $\Delta +0.090$), where decomposition has nothing to decompose into and verifier-driven re-sampling can only redraw from the VGM's existing distribution. On Analogy in particular, M1+M2 even underperforms either module alone (both $+0.139$), since forcing decomposition on an atomic transformation yields contrived intermediates that the verifier then rejects. This residual gap is orthogonal to test-time orchestration, and we view reasoning-oriented VGM training (e.g., physics-aware fine-tuning, symbolic-transformation pretraining) as a complementary future direction.

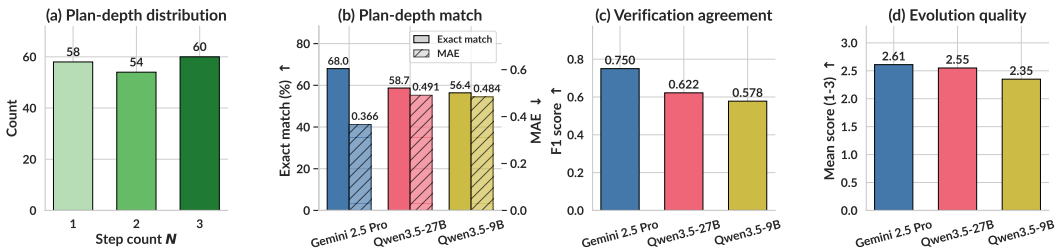


Figure 10: **Human-annotated analysis of planning, verification, and evolution.** (a) Distribution of human-annotated step counts N. (b) Plan-depth match per VLM: exact-match accuracy (left axis) and mean absolute error (MAE) (right axis). (c) Verification agreement: F1 score on a balanced 1:1 split. (d) Evolution quality: mean human rating on a three-point scale.

Human-annotated benchmark and VLM supervision. To externally check the VLM-as-supervisor assumption underlying the closed loop, we construct a human-annotated benchmark on VBVR-Wan2.2 output videos, with each item annotated along three axes that mirror the framework's decision points. *(i) Plan depth.* From the input image and prompt alone, annotators decide the appropriate step count N for solving the task, which reflects the task's reasoning complexity, compared against the VLM's adaptive N from Module 1. *(ii) Clip-level verification.* For 250 clips drawn at the verifier's decision points, annotators independently judge whether the clip realizes the intended sub-action; this is compared against the VLM verifier's accept/reject output from Module 2. *(iii) Evolution scoring.* For 240 rejected clips, annotators rate the suitability of the verifier's suggested repair (used by prompt evolution) on a three-point scale (1: poor, 2: moderate, 3: well-suited).

For reliability and fairness, the accept/reject ratio is held to 1:1 (preventing class-imbalance inflation of verifier accuracy) and the N distribution is balanced across reasoning categories. Each item is independently annotated by AI experts from two different affiliations, with disagreements resolved through cross-validation, providing a non-trivial inter-annotator baseline against which VLM agreement can be calibrated.

With Gemini 2.5 Pro [6] as the planner/verifier, the VLM aligns most closely with human annotators on all three axes (Figure 10), supporting its role as the default planner/verifier in our pipeline. The same axes are used in the VLM ablation (Section 4.3) to compare alternative open-source VLMs (Qwen3.5-27B, Qwen3.5-9B [27]) along an axis decoupled from final task accuracy. Detailed statistics, sample annotations, and per-VLM breakdowns are deferred to the Appendix.

5 Conclusion

We presented VLM-VGM Collaborative Video Reasoning (**CollabVR**), a closed-loop framework that pairs a VLM with a video generation model at step-level granularity: the VLM plans one sub-action at a time, inspects the clip the VGM produces, and adaptively chooses among accepting, regenerating, or further decomposing the action. This step-level coupling redirects test-time compute from sampling more videos toward refining the one being constructed, and it consistently improves both open-source (VBVR-Wan2.2) and closed-source (Veo 3.1) generators over single-inference, Pass@k, and prior test-time scaling baselines on Gen-ViRe and VBVR-Bench. A human-annotated benchmark further confirms that the VLM's plan-depth, verification, and evolution decisions align with expert annotators, supporting the use of a single VLM as an end-to-end supervisor for the loop.

Limitations. Test-time orchestration cannot overcome a VGM that lacks the underlying capability: abstract or symbolic transformations stay hard because the generator never approximates them (Section 4.4), and our gains diminish on lower-capability VGMs whose weak per-step instruction-following and poor preservation of the input frame compound errors across sub-clips faster than re-generation can repair (Section B.6). For the same reason, splitting a task into more steps does not extend the method to longer horizons (Appendix C.4). Progressive planning on its own also helps only generators that execute the requested intermediate states literally (Section 4.3). The verifier is also imperfect, allowing a fraction of failed clips to propagate downstream (Section 4.4). Future directions include reasoning-oriented VGM training and finer-grained failure localization, which can be orthogonally integrated into our test-time loop.

Acknowledgments and Disclosure of Funding

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00457882, AI Research Hub Project; No. RS-2025-02305581, Development of Vision-Language Model (VLM)-Based Intelligent Video Security Monitoring Technology; No. RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST)). It was also supported by the National Research Foundation of Korea (NRF) grant (No. RS-2023-00209060, A Study on Optimization and Network Interpretation Method for Large-Scale Machine Learning) funded by the Korea government (MSIT).

References

[1] Tim Brooks, Bill Peebles, et al. Video generation models as world simulators. https://openai.com/index/video-generation-models-as-world-simulators/, 2024. OpenAI Technical Report.

[2] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[3] Zefan Cai, Haoyi Qiu, Tianyi Ma, Haozhe Zhao, Gengze Zhou, et al. MMGR: Multi-modal generative reasoning. *arXiv preprint arXiv:2512.14691*, 2025.

[4] Harold Haodong Chen, Disen Lan, Wen-Jie Shu, Qingyang Liu, Zihan Wang, et al. TiViBench: Benchmarking think-in-video reasoning for video generative models. In *Computer Vision and Pattern Recognition (CVPR)*, 2026.

[5] Wenyan Cong, Hanqing Zhu, Peihao Wang, et al. Can test-time scaling improve world foundation model? In *Conference on Language Modeling (COLM)*, 2025.

[6] Gemini Team, Google DeepMind. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[7] Google DeepMind. Veo 3.1. Technical report, Google DeepMind, January 2026. URL https://blog.google/innovation-and-ai/technology/ai/veo-3-1-ingredients-to-video/. Released January 13, 2026.

[8] Jiawei Gu, Yunzhuo Hao, Huichen Will Wang, Linjie Li, Michael Qizhe Shieh, Yejin Choi, Ranjay Krishna, and Yu Cheng. ThinkMorph: Emergent properties in multimodal interleaved chain-of-thought reasoning. *arXiv preprint arXiv:2510.27492*, 2025.

[9] Ziyu Guo, Xinyan Chen, Renrui Zhang, Ruichuan An, Yu Qi, et al. Are video models ready as zero-shot reasoners? an empirical study with the MME-CoF benchmark. *arXiv preprint arXiv:2510.26802*, 2025.

[10] Haoran He, Jiajun Liang, Xintao Wang, Pengfei Wan, Di Zhang, Kun Gai, and Ling Pan. Scaling image and video generation via test-time evolutionary search. *arXiv preprint arXiv:2505.17618*, 2025.

[11] Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[12] Ziqi Huang, Ning Yu, Gordon Chen, et al. VChain: Chain-of-visual-thought for reasoning in video generation. *arXiv preprint arXiv:2510.05094*, 2025.

[13] Sangwon Jang, Taekyung Ki, Jaehyeong Jo, Saining Xie, Jaehong Yoon, and Sung Ju Hwang. Self-refining video sampling. *arXiv preprint arXiv:2601.18577*, 2026.

[14] Bingyi Kang, Yang Yue, Rui Lu, Zhijie Lin, Yang Zhao, Kaixin Wang, Gao Huang, and Jiashi Feng. How far is video generation from world model: A physical law perspective. In *International Conference on Machine Learning (ICML)*, 2025.

[15] Chengzu Li, Wenshan Wu, Huanyu Zhang, et al. Imagine while reasoning in space: Multimodal visualization-of-thought. In *International Conference on Machine Learning (ICML)*, 2025.

[16] Chengzu Li, Zanyi Wang, Jiaang Li, Yi Xu, Han Zhou, et al. Thinking in frames: How visual context and test-time scaling empower video reasoning. *arXiv preprint arXiv:2601.21037*, 2026.

[17] Yifan Li, Yukai Gu, Yingqian Min, Zikang Liu, Yifan Du, Kun Zhou, Min Yang, Wayne Xin Zhao, and Minghui Qiu. Beyond the last frame: Process-aware evaluation for generative video reasoning. *arXiv preprint arXiv:2512.24952*, 2026.

[18] Han Lin, Abhay Zala, Jaemin Cho, and Mohit Bansal. VideoDirectorGPT: Consistent multi-scene video generation via LLM-guided planning. In *Conference on Language Modeling (COLM)*, 2024.

[19] Fangfu Liu, Hanyang Wang, Yimo Cai, Kaiyan Zhang, Xiaohang Zhan, and Yueqi Duan. Video-T1: Test-time scaling for video generation. In *International Conference on Computer Vision (ICCV)*, 2025.

[20] Xinxin Liu, Zhaopan Xu, Ming Li, Kai Wang, Yong Jae Lee, and Yuzhang Shang. Can world simulators reason? Gen-ViRe: A generative visual reasoning benchmark. *arXiv preprint arXiv:2511.13853*, 2025.

[21] Yang Luo, Xuanlei Zhao, Baijiong Lin, Lingting Zhu, Liyao Tang, Yuqi Liu, Ying-Cong Chen, Shengju Qian, Xin Wang, and Yang You. V-ReasonBench: Toward unified reasoning benchmark suite for video generation models. *arXiv preprint arXiv:2511.16668*, 2025.

[22] Nanye Ma, Shangyuan Tong, Haolin Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. In *Computer Vision and Pattern Recognition (CVPR)*, 2025.

[23] Sachit Menon, Richard Zemel, and Carl Vondrick. Whiteboard-of-thought: Thinking step-by-step across modalities. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

[24] Meta AI. Movie Gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

[25] NVIDIA Cosmos Team. World simulation with video foundation models for physical ai, 2025. URL https://arxiv.org/abs/2511.00062.

[26] Joonhyung Park, Hyeongwon Jang, Joowon Kim, and Eunho Yang. Progress by pieces: Test-time scaling for autoregressive image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 38091–38100, 2026.

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

[28] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[29] Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, et al. OpenThinkIMG: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*, 2025.

[30] Zhaochen Su, Peng Xia, Hangyu Guo, et al. Thinking with images for multimodal reasoning: Foundations, methods, and future frontiers. *arXiv preprint arXiv:2506.23918*, 2025.

[31] Jingqi Tong, Yurong Mou, Hangcheng Li, Mingzhe Li, Yongzhuo Yang, et al. Thinking with video: Video generation as a promising multimodal reasoning paradigm. In *Computer Vision and Pattern Recognition (CVPR)*, 2026.

[32] Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[33] Maijunxian Wang, Ruisi Wang, Juyi Lin, et al. A very big video reasoning suite. *arXiv preprint arXiv:2602.20159*, 2026.

[34] Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. VideoAgent: Long-form video understanding with large language model as agent. In *European Conference on Computer Vision (ECCV)*, 2024.

[35] Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners. *arXiv preprint arXiv:2509.20328*, 2025.

[36] Keming Wu, Yijing Cui, Wenhan Xue, Qijie Wang, Xuan Luo, Zhiyuan Feng, Zuhao Yang, Sudong Wang, Sicong Jiang, Haowei Zhu, Zihan Wang, Ping Nie, Wenhu Chen, and Bin Wang. WorldReasonBench: Human-aligned stress testing of video generators as future world-state predictors. *arXiv preprint arXiv:2605.10434*, 2026.

[37] Tsung-Han Wu, Long Lian, Joseph E. Gonzalez, Boyi Li, and Trevor Darrell. Self-correcting LLM-controlled diffusion models. In *Computer Vision and Pattern Recognition (CVPR)*, 2024.

[38] You-Zhe Xie, Yu-Hsuan Li, Jie-Ying Lee, Kaipeng Zhang, Yu-Lun Liu, and Zhixiang Wang. YoCausal: How far is video generation from world model? A causality perspective. *arXiv preprint arXiv:2605.30346*, 2026.

[39] Qiyao Xue, Xiangyu Yin, Boyuan Yang, et al. PhyT2V: LLM-guided iterative self-refinement for physics-grounded text-to-video generation. In *Computer Vision and Pattern Recognition (CVPR)*, 2025.

[40] Cheng Yang, Haiyuan Wan, Yiran Peng, Xin Cheng, Zhaoyang Yu, et al. Reasoning via video: The first evaluation of video models' reasoning abilities through maze-solving tasks. *arXiv preprint arXiv:2511.15065*, 2025.

[41] Ling Yang, Zhaochen Yu, Chenlin Meng, Minkai Xu, Stefano Ermon, and Bin Cui. Mastering text-to-image diffusion: Recaptioning, planning, and generating with multimodal LLMs. In *International Conference on Machine Learning (ICML)*, 2024.

[42] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, et al. UniSim: Learning interactive real-world simulators. In *International Conference on Learning Representations (ICLR)*, 2024.

[43] Xindi Yang, Baolu Li, Yiming Zhang, et al. VLIPP: Towards physically plausible video generation with vision and language informed physical prior. In *International Conference on Computer Vision (ICCV)*, 2025.

[44] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, et al. CogVideoX: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations (ICLR)*, 2025.

[45] Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Lulu Gu, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, Yu Qiao, and Ziwei Liu. VBench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*, 2025.

Appendix

A Implementation Details

A.1 Hyperparameters

Pipeline budget. We set $N_{\max}=3$ planning steps and $M=3$ per-step generation attempts as the default **CollabVR** configuration. The step-level verifier receives one frame per second uniformly sampled from each candidate clip. For Gemini 2.5 Pro we instead pass the raw video as input and let the model perform its own frame sampling internally.

Vision-Language Models (VLMs). All VLM calls (planner, step-level verifier, and failure router) use Gemini 2.5 Pro (`gemini-2.5-pro`) with decoding temperature 0.2. The same model identity also serves as the output reward model for Pass@k selection and as the critic for the VideoTPO baseline, keeping comparisons matched in VLM compute. Prompt templates are listed in Appendix A.2.

Video Generation Models (VGMs). For VBVR-Wan2.2, we use the released 14B image-to-video checkpoint (Wan2.2 fine-tuned on VBVR-Bench reasoning data), generating at a maximum area of 832×480 with aspect ratio preserved, at 16 fps, with 20 sampling steps and CFG scale 5.0. On Gen-ViRe, the first step produces a 6s clip (96 frames) and subsequent steps produce 3s clips (48 frames). On VBVR-Bench, we follow the official setup and match each clip to the duration of the corresponding ground-truth video. Inference runs on a single A100 GPU. For Veo 3.1, we use the `veo-3.1-fast-generate-preview` API at native resolution (16:9 with letterboxing for non-matching inputs) and 24 fps. The first step generates 8s of video and subsequent steps generate 4s. Due to API cost, we evaluate Veo 3.1 only on the smaller Gen-ViRe benchmark. For Cosmos-Predict-2.5, we use the released 14B post-trained checkpoint at 832×480 resolution and 16 fps, generating 5s clips per step. All other settings (sampling steps, guidance scale, negative prompt, scheduler, precision) follow the upstream default configuration of the released checkpoint.

Baselines. *Single Inference* generates one video per sample using the same clip length as the $N=1$ branch of **CollabVR**. *Pass@k* with $k \in \{2, 4\}$ runs k independent generations with seeds $\{1, \ldots, k\}$ under identical inference settings. Gemini 2.5 Pro is shown all k candidates in a single call and selects the best, acting as an output reward model. We also tested scoring each seed independently and choosing the highest scorer, but the single-call joint selection gave both higher accuracy and stronger alignment with human judgments. We expect independent scoring to become more competitive as k grows. *VideoTPO* follows the official setup with two prompt-rewrite iterations, two seeds per iteration, and a final iteration using only seed 1, yielding five total generations per sample. The LLM optimizer and the critic VLM are both Gemini 2.5 Pro to match **CollabVR**.

Evaluation. Gen-ViRe is judged by Gemini 2.5 Pro using the rubric prompts from the official repository, with each sample scored across three independent runs to absorb judge stochasticity. VBVR-Bench uses a deterministic rule-based protocol that compares each generated video against ground-truth references, so we report a single run. We report Cost as the total VGM-generated seconds per sample, since VLM compute is negligible relative to VGM compute.

A.2 Prompt Templates

We provide the verbatim prompt templates used by the VLM in our pipeline: (i) the *progressive planner*, which emits the next action prompt and a task-complete flag; and (ii) the *step verifier*, which returns an accept/reject judgment with a structured diagnosis d (textual reason, actionable suggestion, and a `good_fraction` estimate of how much of the clip executed correctly). Each VLM call additionally receives the input image and the history of accepted clip frames; the verifier additionally receives the candidate clip. Prompt evolution itself does not require a separate VLM call: the verifier's `suggestion` field is folded directly into the next action prompt by $\mathrm{evolve}(a_t, d)$.

Progressive planner (`step_planner`)

```
You are a visual reasoning expert that plans video generation steps for an
↪  image-to-video model.

Given an input image and a task prompt, you must break the task into sequential
↪  action steps that a video generation model can execute one at a time. Each
↪  step should describe a short, concrete visual action (4-8 seconds of video).

Inputs
- TASK_PROMPT: The full task description
- CURRENT_IMAGE: The current state (input image or last frame of previous clip)
- COMPLETED_STEPS: What has already been done (empty if first step)
- STEP_NUMBER: Which step we are planning (1-indexed)

Rules
1. Each step must describe ONE clear visual action that a video generation model
↪  can simulate in 6 seconds.
2. DO NOT plan the entire task at once. Only plan the NEXT IMMEDIATE step.
3. Describe the action in terms of VISIBLE MOTION and CHANGE -- what should
↪  move, where, and how.
4. Include the EXACT target state: what the frame should look like when this
↪  step is done.
5. If the task appears to be already complete based on the current image, set
↪  "task_complete" to true.

Output (strict JSON):
{
  "observation": "Brief description of what you see in the current image",
  "remaining_goal": "What still needs to happen to complete the task",
  "task_complete": false,
  "instruction": "Detailed video generation prompt for the next step.",
  "target_state": "Visual description of what the last frame should look like
  ↪  after this step",
  "estimated_steps_remaining": 2
}
```

Step verifier (`step_verifier`)

```
You are a video quality judge evaluating whether a generated video clip executed
↪  its intended action correctly.

Inputs: TASK_PROMPT, PLANNED_ACTION, TARGET_STATE, VIDEO.

Critical Distinction: judge whether the ACTION was executed, NOT whether the
↪  full task is complete.

Evaluation Criteria
1. Did the intended motion/transformation START to happen in the correct
↪  direction?
2. Is the result CONSISTENT with the planned action (even if incomplete)?
3. Were there FUNDAMENTAL errors (wrong direction, wrong object, completely
↪  wrong action, scene collapse)?

What is NOT a rejection reason:
- Action happened but didn't fully complete (partial progress is fine)
- Minor rendering artifacts or small imprecisions
- The final task goal is not yet reached (planner's job)

On Rejection -- estimate "good_fraction" (0.0-1.0): the fraction of the video
↪  that was correct before the error occurs. We use this to do partial
↪  re-generation from the good portion.

Output (strict JSON):
```

```
If accept:
{
  "verdict": "accept",
  "action_executed": true,
  "progress": "partial or complete",
  "confidence": "high",
  "reason": "...",
  "suggestion": ""
}

If reject:
{
  "verdict": "reject",
  "action_executed": false,
  "progress": "none",
  "good_fraction": 0.3,
  "confidence": "high",
  "reason": "...",
  "suggestion": "..."
}
```

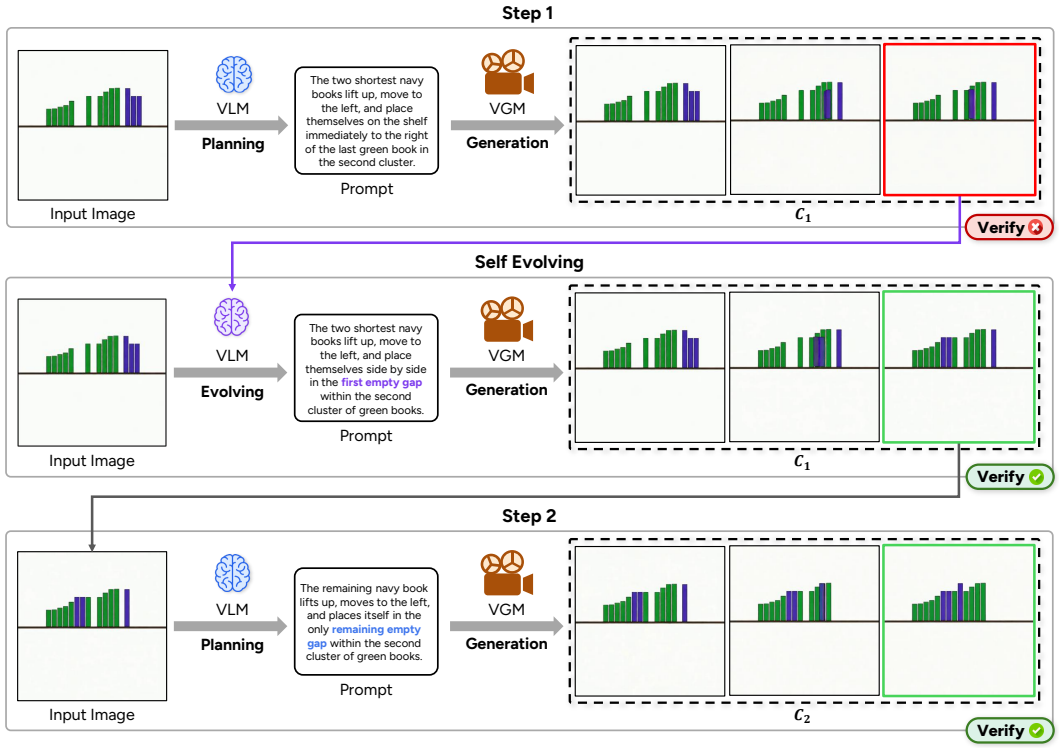


Figure 11: **Trace through one full CollabVR loop on a multi-step bookshelf task (VBVR-Wan2.2).** Within-step prompt evolution (M2) corrects C_1 in Step 1, after which across-step progressive planning (M1) advances to Step 2, sharing the same per-clip verifier.

A.3 Verifier Output Examples

Figure 11 instantiates one full execution of Algorithm 1 on a multi-step bookshelf task, exercising both branches of the inner loop in a single trace.

Step 1: the reject-and-evolve branch. The planner emits a_1 (*"...place the two navy books on the shelf immediately to the right of the last green book..."*) and the VGM produces c_1. The verifier rejects: the two navy books land flush against the cluster's right edge rather than inside an empty gap. Its diagnosis d_1, packaging a textual reason and a concrete suggestion, is folded into the next prompt

as $a_1' = \text{evolve}(a_1, d_1)$ (*"…first empty gap within the second cluster…"*, the Self-Evolving panel). The VGM regenerates c_1, and the verifier now accepts; the clip is committed to $\mathcal{H}$ and its last frame becomes the conditioning frame for Step 2.

Step 2: the accept-on-first-attempt branch. Conditioned on $\mathcal{H} = \{c_1\}$, the planner emits a_2 for the remaining navy book. The VGM produces c_2, the verifier accepts immediately, and the outer loop terminates at $N{=}2$ without ever entering the evolve branch.

Coverage of the (v, d) interface. Together, the three verifier calls cover both routes the per-clip (v, d) interface can take: `reject`-with-suggestion drives within-step prompt rewriting (M2), and `accept` commits the clip and advances the across-step plan (M1). The auxiliary failure router (Appendix A.5) is reached only when within-step recovery is exhausted, which does not happen here.

A.4 First-Frame Fidelity for VGM Selection

First-frame fidelity, which measures how closely the first frame of a generated clip matches the conditioning image, is a prerequisite for the step-by-step clip concatenation in our pipeline (each step's first frame must faithfully resume from the previous step's final frame) as well as for the partial re-generation auxiliary that re-enters the VGM at a mid-clip frame (Appendix E.4); we therefore use it as a key selection criterion for candidate VGMs. We compute the SSIM between the input image and the first frame of the generated video on Gen-ViRe.

Table 5: First-frame fidelity (SSIM) on Gen-ViRe.

VGM	SSIM (mean $\pm$ std)
VBVR-Wan2.2	0.970 ± 0.043
Cosmos-Predict 2.5	0.971 ± 0.037
Veo 3.1	0.977 ± 0.035
Sora 2	0.818 ± 0.137

VBVR-Wan2.2, Veo 3.1, and Cosmos-Predict 2.5 all achieve sufficient first-frame fidelity for our pipeline. Sora 2 uses an `input_reference` field rather than strict first-frame conditioning, so its first frame drifts from the input. We therefore exclude Sora 2 from main **CollabVR** results since downstream step concatenation cannot preserve information across clips, and Sora 2's upcoming service deprecation further reduced the value of additional engineering against it.

A.5 Auxiliary Failure Router

Our implementation extends the per-step prompt evolution of Section 3.3 with an auxiliary VLM call, the `failure_router`, that takes over when evolution alone cannot produce an accepted clip. After all M within-step evolution attempts return reject, the router examines the failed clip together with the verifier's diagnosis and chooses one of three follow-up actions. The simplest is *regen*, a single-shot retry that starts from the first failing frame whenever the verifier's `good_fraction` is moderate-to-high, so the correctly executed prefix is preserved (Appendix E.4). When the failure is structural rather than a one-off slip, the router triggers *split*, which decomposes the residual task into additional sub-steps and re-enters the progressive planner. Finally, when prior decomposition has produced step-boundary artifacts on what is fundamentally a single-shot transformation, *fallback* collapses to a single-inference run with $N{=}1$.

Veo 3.1's strong single-shot prior often suffices on its own. We therefore invoke the same router at the sample level before any per-step planning, choosing between accepting the single-shot baseline and proceeding with multi-step orchestration. The full prompt template is shown below.

Failure router (`failure_router`)

```
You are a visual reasoning expert deciding how to recover from a FAILED
↪  single-shot video generation. Choose among three recovery strategies.

Inputs: TASK_PROMPT, INPUT_IMAGE, FAILED_VIDEO, REJECT_REASON, SUGGESTION,
↪  GOOD_FRACTION.
```

```
Strategies:
- "regen": single-shot retry. Choose when the failure was an execution slip on a
↪  single coherent transformation; good_fraction moderate-to-high; partial
↪  regen will likely fix it.
- "split": decompose into multiple steps. Choose when the failure is structural
↪  -- the task fundamentally needs intermediate states (multi-action
↪  procedures, navigation with turns, assembly, drawing multiple objects), or
↪  good_fraction is very low and the failure indicates conflated steps.
- "fallback": collapse to single-inference (N=1). Choose when prior
↪  decomposition introduced step-boundary artifacts and the residual task is
↪  simple enough to be generated in a single clip.


Output (strict JSON):
{"action": "regen", "suggestion": "...", "reason": "..."}
or
{"action": "split", "estimated_steps": 3, "suggestion": "...", "reason": "..."}
or
{"action": "fallback", "suggestion": "...", "reason": "..."}
```

B Additional Quantitative Results

B.1 User Study

We additionally conduct a blind, side-by-side human study comparing **CollabVR** against Pass@1 and Pass@4 at matched VGM (VBVR-Wan2.2). The study is hosted on Prolific, and the per-trial UI is shown in Figure 12.

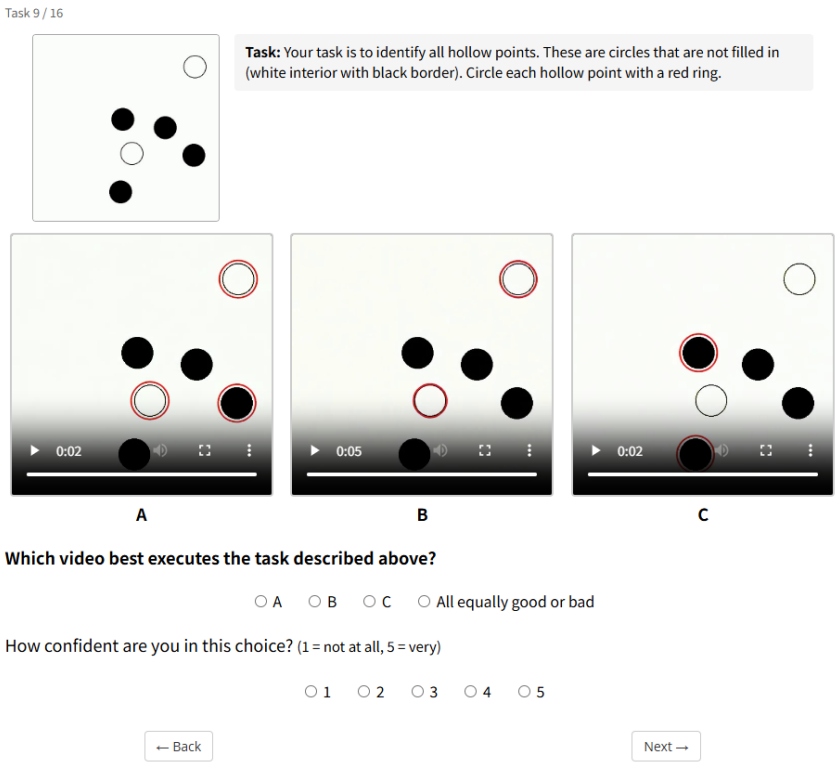


Figure 12: **Per-trial user-study UI.** The participant sees the task prompt and the input image, watches three blinded videos, and answers a forced-choice preference and a confidence rating. The condition→label mapping is randomized per task per participant.

Sample selection. We curate 16 tasks spanning Gen-ViRe and VBVR-Bench (one task per Gen-ViRe category and a balanced In-Domain / Out-of-Domain split on VBVR). Within each category we pick the task whose per-sample score gap between **CollabVR** and Pass@1/Pass@4 is largest (top-$|\Delta|$), excluding cases where all three methods score 0 or 1 (no signal). For each task, three videos (Pass@1, Pass@4, **CollabVR**) are mapped to the labels A/B/C by a fresh random permutation per task per participant.

Procedure. On each task page, the participant sees the input image, the task description, and three blinded videos, and answers two questions: "which video best executes the task?" (A/B/C/equal) and a 1–5 confidence Likert.

Results. We collect n=40 valid submissions across the 16 tasks. *Equal* responses comprise 17.7% of all trials; aggregating the remaining decisive preferences: In head-to-head comparisons (excluding *Equal* votes), **CollabVR** is preferred over Pass@1 in **91.7**% and over Pass@4 in **78.9**% (Figure 13). These margins corroborate the automated-metric gains in Section 4: human raters on a blinded side-by-side comparison consistently prefer **CollabVR** outputs over both single-shot and best-of-4 baselines. Inter-rater agreement is moderate: average pairwise raw agreement is 66.3% (decisive-only) and Gwet's AC_1 is 0.575.

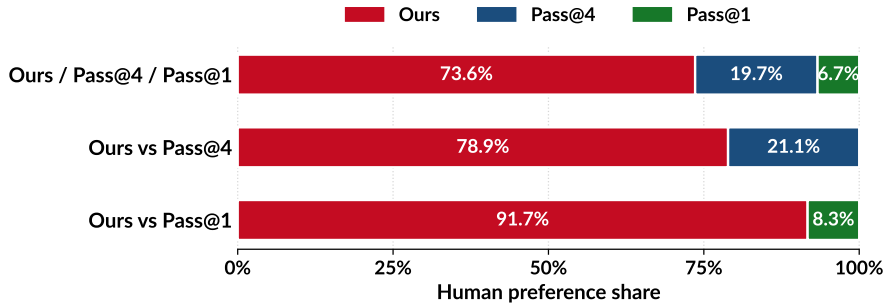


Figure 13: **Human preference share on the user study** (n=40 participants, 16 tasks; *Equal* responses excluded). Each row is a 100%-stacked bar restricted to the listed conditions. **CollabVR** is the dominant choice in all three views.

B.2 Effect of Per-Step Attempt Budget M

Effect of attempt budget M. We sweep the per-step attempt budget $M \in \{1, \ldots, 5\}$ on VBVR-Bench with VBVR-Wan2.2 under the M2-only configuration (Figure 14). The first two budget increments deliver most of the gain (+4.89% from M=1 to M=2, +1.59% from M=2 to M=3), after which each additional attempt yields below 1% (+0.97%, +0.51%) while cost continues to grow at a near-constant per-step rate. Beyond M=3, the cost-quality profile thus collapses to that of plain Pass@k re-sampling: extra compute buys little more than an additional independent draw. We therefore set M=3 as the default, retaining the cost-efficient regime where verifier-guided re-generation still provides meaningful repair.

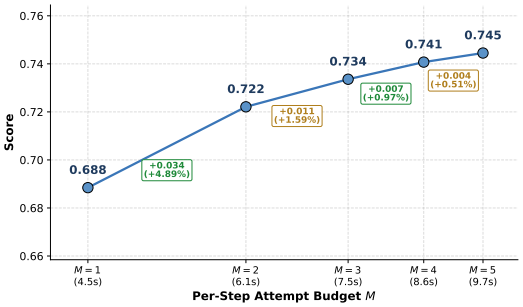


Figure 14: **Per-step attempt budget M on VBVR-Bench (VBVR-Wan2.2, M2-only).** Score grows monotonically with M, but per-step gains drop below 1% beyond M=3 while cost continues to scale nearly linearly.

B.3 Per-VLM Human-Alignment Breakdown

We provide the per-VLM, per-axis numbers underlying Section 4.4 in Table 6, pooling annotations from Gen-ViRe and VBVR-Bench (the latter contributed by collaborators). Three axes are evaluated: D1 plan-depth (N prediction), D2 verifier accept/reject agreement on step clips, and D3 evolution quality of the verifier's suggested repair (1=poor, 2=moderate, 3=well).

Table 6: Overall human-eval across 3 VLMs. D1 plan-depth on VBVR (n=100, GT video shown) and Gen-ViRe (n=72, image only). D2 verifier agreement on a balanced 125 accept + 125 reject sub-sample. D3 evolution-suggestion quality on n=80 reject clips per VLM.

	Gemini 2.5 Pro	Qwen3.5-27B	Qwen3.5-9B
D1 plan-depth (exact-match)			
VBVR-Bench	64.0%	64.2%	53.0%
Gen-ViRe	73.6%	55.6%	61.1%
Overall	**68.0%**	58.7%	56.4%
MAE (parseable)	0.366	0.491	0.484
D2 verifier agreement			
accept-recall	84.8%	82.4%	77.6%
reject-recall (failure detection)	**65.6%**	44.8%	40.8%
Overall	**75.2%**	63.6%	59.2%
Cohen's κ (Gen-ViRe)	0.676	0.432	0.304
D3 evolution quality (mean 1–3)			
Overall mean	**2.61**	2.55	2.35
≥ 2 (moderate or better)	93.8%	95.0%	86.3%
$= 3$ (top)	67.5%	60.0%	48.8%

Gemini 2.5 Pro leads on every axis. The closed-vs-open gap is largest on the verifier (D2 reject-recall: 65.6% vs. 40–45%, a $\sim$ 21-point gap; per-verdict confusion matrices in Table 7), suggesting the gap is in training data and objective rather than scale alone (the size lift from 9B to 27B contributes only +4 points on this axis). On the evolution axis (D3) the closed-vs-open gap shrinks to $\Delta = 0.06$ (Gemini–27B), within noise. D1 ranking flips between setups: 9B over-counts steps when given the GT video on VBVR-Bench (N=3 collapses to 2/20 correct) but matches Gemini's plan-depth distribution on image-only Gen-ViRe.

Table 7: Verdict confusion matrices, balanced 125:125 sub-sample.

	Gemini 2.5 Pro		Qwen3.5-27B		Qwen3.5-9B	
	H=acc	H=rej	H=acc	H=rej	H=acc	H=rej
pred=accept	106	43	103	69	97	74
pred=reject	19	82	22	56	28	51

B.4 Per-Category Δ Heatmap on VBVR-Bench

We mirror the per-category Δ analysis of Section 4.4 on VBVR-Bench (Figure 15). Module dominance flips relative to Gen-ViRe: M2 alone is now the larger single-module contributor (Table 3: Δ_{M2} +0.063 vs. Δ_{M1} +0.035), driven primarily by Abstract (M2 alone +0.131) where atomic perceptual targets are recoverable through verifier-driven re-sampling but admit no genuine sub-goals to expose. The two modules dominate different categories: M1 alone is actually the larger contributor on Spatial (+0.061 vs. M2 +0.019), where decomposition exposes navigation-style sub-goals that the VGM can satisfy individually, while Transformation is the clearest case in which neither module alone is sufficient (Δ_{M1} −0.040, Δ_{M2} +0.032) yet the combination recovers a clean positive gain (Δ_{M1+M2} +0.075). With both modules enabled, all five categories turn positive (+0.010 to +0.159), reproducing the per-instance adaptive activation observed on Gen-ViRe.

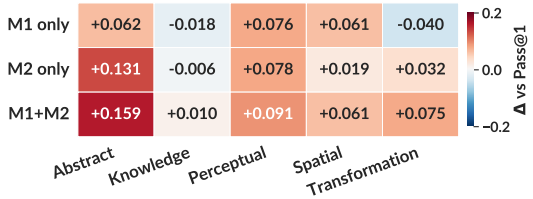


Figure 15: **Per-category Δ over Pass@1 on VBVR-Bench (VBVR-Wan2.2).** Module configurations follow Section 4.3; per-category overall is the sample-count-weighted mean of the In-Domain and Out-of-Domain entries (categories carry 115, 70, 150, 65, 100 samples with different ID/OOD splits).

Yet the symbolic limit identified in Section 4.4 surfaces here as well: Knowledge remains marginal across all three configurations (Δ $+0.010$ for M1+M2, an order of magnitude below the other categories), since neither decomposing nor re-sampling supplies the missing world knowledge—there is no sub-step to expose, and the verifier can only redraw from the VGM's existing distribution. This parallels the small Analogy ($+0.083$) and Abstract ($+0.090$) gains on Gen-ViRe, again pointing to knowledge-grounded VGM training as a complementary direction rather than additional planner or verifier modules.

B.5 Veo 3.1 Module Ablation

We ablate the same M1 / M2 modules on Gen-ViRe with Veo 3.1, mirroring Table 3 for VBVR-Wan2.2.

Table 8: Per-module ablation on Gen-ViRe with Veo 3.1. Δ over Pass@1. M1 / M2 columns indicate which module is enabled. The last row is an additional ablation that fixes $N{=}3$ for every sample, bypassing the planner's per-state adaptive N selection.

M1	M2	Cost (s)	Overall	Δ
✗	✗	8.0	0.481	–
✓	✗	10.1	0.446	-0.035
✗	✓	12.5	0.527	+0.046
✓	✓	21.4	**0.550**	**+0.069**
Ablation: fixed $N{=}3$ (bypass progressive N selection)				
✓	✓	24.5	0.450	-0.031

The verifier-and-regen module M2 alone has a single-shot ceiling. With $N{=}1$ and $M{=}2$ (no decomposition, regen only), Veo 3.1 reaches 0.527, a $+0.046$ gain over Pass@1 that is almost identical to the corresponding $+0.045$ gain on VBVR-Wan2.2 (Table 3). However, this number reflects only the tasks that Veo 3.1 can complete in a single clip; tasks requiring multiple sequential sub-actions remain out of reach for any number of regenerations.

M1 unlocks the tasks where single-shot is insufficient. The full M1+M2 configuration (max $N{=}3$, max $M{=}2$) reaches 0.550, a further $+0.023$ over the M2-only ceiling. The gain comes from samples on which the single-shot stream cannot complete the task: there, the planner emits a multi-step decomposition ($N{>}1$) that breaks the task into sub-actions Veo 3.1 can satisfy individually. M1's decomposition is therefore the active ingredient: it unlocks tasks that no amount of regeneration on a single clip could solve. Even with the maximum decomposition forced ($N{=}3$ for every sample, last row of Table 8), Veo 3.1 drops to 0.450, confirming that adaptive N selection, not just the existence of decomposition, is what makes M1 productive.

Why M1 alone helps VBVR-Wan2.2 but actually *hurts* Veo 3.1. M1 alone delivers most of the total gain on VBVR-Wan2.2 ($+0.120$ of $+0.140$; Table 3), yet on Veo 3.1 it falls 0.035 below Pass@1. The asymmetry follows from the two VGMs' design priors. VBVR-Wan2.2 is fine-tuned for visual reasoning, so the artificial intermediate states our planner emits (partial rotations, half-drawn lines, incremental fills) are inside its training distribution; each sub-step executes literally, leaving the next step's conditioning aligned with the planner's target. Veo 3.1's strong end-to-end motion prior gives it a higher single-shot Pass@1 (0.481 vs. 0.391 for VBVR-Wan2.2), but reinterprets these intermediate-state requests into outputs the prior considers natural, so with $M{=}0$ the deviation feeds the next step's conditioning unchecked and errors compound across the chain. M1 alone therefore hurts Veo 3.1 most on Planning (-0.069) and Spatial (-0.067), the categories where Veo 3.1's single-shot capacity was highest to begin with. M2 reconciles Veo 3.1's generalist prior with the planner's task-specific structure: the verifier rejects mid-state deviations before they propagate, and the regen loop pulls Veo 3.1's output back onto the trajectory. This is why M1+M2 reaches 0.550 even though M1 alone underperforms, and the same dynamic explains the Spatial-category synergy in Section 4.4: long-horizon simulation needs both modules, since M1 alone accumulates deviation while M2 alone hits the single-shot ceiling.

Leave-one-out ablation. Table 9 reports the complementary ablation, in which one module is removed from the full pipeline. Removing either module lowers the score in all three settings. On

Veo 3.1, removing M1 costs 0.023 even though M1 alone is below Pass@1, because M1 lets the full pipeline solve tasks that cannot be completed within a single clip.

Table 9: Leave-one-out ablation. Each entry is the score after removing one module from the full pipeline, with the difference in parentheses.

Setting	Full	w/o M1	w/o M2
Gen-ViRe, VBVR-Wan2.2	0.531	0.436 (−0.095)	0.511 (−0.020)
VBVR-Bench, VBVR-Wan2.2	0.757	0.734 (−0.023)	0.706 (−0.051)
Gen-ViRe, Veo 3.1	0.550	0.527 (−0.023)	0.446 (−0.104)

B.6 Cosmos-Predict-2.5 Detailed Results

CollabVR improves Cosmos-Predict 2.5 (14B) on VBVR-Bench (Table 2, $0.308 \rightarrow 0.403$) but degrades on Gen-ViRe ($0.287 \rightarrow 0.182$, $\Delta = -0.105$ vs. Pass@4; Table 10). This contrast follows each benchmark's task profile: VBVR-Bench is dominated by single-step tasks, where **CollabVR** mostly invokes single-clip verification with at most a few re-generations, both of which Cosmos handles competently. Gen-ViRe is dominated by multi-step reasoning tasks (Section 4.3), so **CollabVR** forces Cosmos to execute decomposed sub-actions across $N \leq 3$ clips. Here Cosmos's weaker per-step instruction-following becomes the bottleneck and decomposition compounds rather than absorbs the error across clips.

Table 10: Cosmos-Predict 2.5 per-category on Gen-ViRe, fair intersection of **CollabVR** and Pass@k.

Method	Abstract	Algo	Analogy	Perc.	Planning	Spatial	Avg
Pass@1 (baseline)	0.204	0.346	0.042	0.305	0.352	0.200	0.246
Pass@4	0.278	0.432	0.042	0.288	0.444	0.217	0.287
CollabVR (N=3)	0.009	0.249	0.375	0.000	0.368	0.131	0.182

The single Gen-ViRe category where **CollabVR** still helps Cosmos is Analogy ($0.042 \rightarrow 0.375$, $\Delta = +0.333$): Pass@1 is essentially zero there, so any successful sub-step accumulates to a positive end-state. This delineates the regime where decomposition still helps a weak VGM: when single-shot performance is so low that any partial progress beats the baseline.

Scene collapse in Cosmos-Predict-2.5. To find where Cosmos breaks down, we manually inspected all of its clips on Gen-ViRe, including re-generations. Each clip was shown as four frames and mixed with VBVR-Wan2.2 clips so that the annotator did not know which model had produced it, and was labeled as collapsed, severely damaged, or intact. The most frequent failure is scene collapse, in which the clip loses the input frame and turns into an unrelated scene. Of the Cosmos clips, 31.7% are collapsed and 58.8% are collapsed or severely damaged, compared with about 1% for VBVR-Wan2.2. Because each step starts from the last frame of the previous one, a collapsed clip also corrupts the input of the next step. This explains why Cosmos degrades on Gen-ViRe, where most tasks require several steps, while it still improves on VBVR-Bench (+0.095). The labels were assigned by a single annotator from still frames.

Limitation. **CollabVR**'s gain on a given VGM is bounded below by the VGM's per-step instruction-following reliability. We frame **CollabVR** as orthogonal to, rather than a substitute for, training stronger VGMs: the framework requires a generator that meets a minimum per-step instruction-following floor before decomposition becomes profitable.

C Robustness Analysis

Gemini 2.5 Pro serves both as the VLM inside **CollabVR** and as the rubric judge of Gen-ViRe. We therefore repeat the Gen-ViRe comparison with a different judge, with a different in-loop VLM, and over multiple generation runs, and we also evaluate larger step budgets on long-horizon tasks. All experiments in this section use VBVR-Wan2.2.

C.1 Evaluation with GPT-5 as the Judge

We re-score the videos from the main experiment with GPT-5 as the rubric judge, keeping the outputs selected for Pass@k unchanged (Table 11). Since the GPT-5 API does not take video input, each video is given as eight uniformly sampled frames, and each sample is scored once. Scores are lower overall under GPT-5, but the five methods keep the same order as under Gemini 2.5 Pro, and the margin of **CollabVR** over VideoTPO is almost unchanged ($+0.042$ against $+0.043$).

Table 11: Results on Gen-ViRe judged by GPT-5. The videos are the same as in Table 1.

Method	Avg.	Abst.	Algo.	Analog.	Perc.	Plan.	Spat.
Pass@1	0.345	0.375	0.441	0.250	0.280	0.371	0.353
Pass@2	0.382	0.479	0.457	0.250	0.298	0.442	0.367
Pass@4	0.403	0.542	0.415	0.333	0.280	0.479	0.369
VideoTPO	0.435	0.458	0.431	0.500	0.306	0.562	0.353
CollabVR	**0.477**	0.406	0.620	0.417	0.323	0.654	0.444

C.2 Effect of the In-Loop VLM

We next replace the VLM inside every method with GPT-5 or Qwen3.5-27B. This VLM selects the output for Pass@k, critiques the video in VideoTPO, and serves as the planner and verifier in **CollabVR**, while the judge remains Gemini 2.5 Pro (Table 12). **CollabVR** achieves the highest score with all three VLMs. With GPT-5 in the loop it reaches 0.541, slightly above the 0.531 obtained with Gemini 2.5 Pro, so the Gemini 2.5 Pro judge does not favor outputs that were produced with Gemini 2.5 Pro in the loop.

Table 12: Results on Gen-ViRe with different VLMs inside each method, judged by Gemini 2.5 Pro.

In-loop VLM	Method	Avg.	Abst.	Algo.	Analog.	Perc.	Plan.	Spat.
–	Pass@1	0.391	0.479	0.415	0.250	0.261	0.554	0.387
Gemini 2.5 Pro	Pass@2	0.398	0.576	0.437	0.278	0.257	0.481	0.357
	Pass@4	0.438	0.622	0.418	0.250	0.275	0.604	0.462
	VideoTPO	0.488	0.535	0.443	0.417	0.313	0.671	0.552
	CollabVR	**0.531**	0.569	0.606	0.333	0.367	0.821	0.488
GPT-5	Pass@2	0.406	0.556	0.447	0.222	0.274	0.572	0.368
	Pass@4	0.423	0.656	0.387	0.250	0.272	0.564	0.409
	VideoTPO	0.452	0.392	0.497	0.417	0.338	0.661	0.409
	CollabVR	**0.541**	0.507	0.608	0.472	0.502	0.682	0.491
Qwen3.5-27B	Pass@2	0.398	0.503	0.379	0.250	0.280	0.547	0.427
	Pass@4	0.403	0.611	0.345	0.250	0.260	0.564	0.388
	VideoTPO	0.494	0.608	0.528	0.333	0.428	0.583	0.486
	CollabVR	**0.510**	0.562	0.540	0.417	0.267	0.773	0.503

C.3 Variance over Generation Runs

All Gen-ViRe scores in the paper are averaged over three judge runs, which accounts for variance in judging but not in generation. We therefore run **CollabVR** with three different generation seeds (Table 13). The mean score is 0.538 with a standard deviation of 0.007, and every run stays above VideoTPO (0.488).

C.4 Effect of Larger Step Budgets

To examine tasks that need longer decompositions, we take the 52 of the 72 Gen-ViRe samples that use all three steps under the default $N_{\max}$=3 and rerun them with $N_{\max}$=4 and $N_{\max}$=5 (Table 14). Neither setting improves over $N_{\max}$=3. Rejections concentrate on the last step regardless of the budget, with final-reject rates at the last step of 46.2%, 51.0%, and 61.9% for $N_{\max}$=3, 4, and 5, and 58% of the rejected cases still fail after all five steps. These failures arise in executing individual sub-actions, and additional steps do not resolve them. Decomposition reduces the difficulty of

Table 13: **CollabVR** on Gen-ViRe over three generation runs. The last row reports the mean and standard deviation.

Method	Avg.	Abst.	Algo.	Analog.	Perc.	Plan.	Spat.
Pass@1	0.391	0.479	0.415	0.250	0.261	0.554	0.387
Pass@4	0.438	0.622	0.418	0.250	0.275	0.604	0.462
VideoTPO	0.488	0.535	0.443	0.417	0.313	0.671	0.552
CollabVR (seed 1)	0.531	0.569	0.606	0.333	0.367	0.821	0.488
CollabVR (seed 2)	0.544	0.663	0.556	0.444	0.301	0.742	0.561
CollabVR (seed 3)	0.537	0.649	0.556	0.444	0.297	0.726	0.550
CollabVR (mean)	0.538 ± 0.007	0.627 ± 0.051	0.573 ± 0.029	0.407 ± 0.064	0.322 ± 0.039	0.763 ± 0.051	0.533 ± 0.039

chaining several actions in order, but it does not help when a single sub-action is already beyond what the generator can render, and with current VGMs this trade-off favors about three steps.

Table 14: **CollabVR** with larger step budgets on the 52 Gen-ViRe samples that use all three steps at $N_{\max}$=3.

$N_{\max}$	3	4	5
Gen-ViRe subset (n=52)	**0.517**	0.463	0.482

D Module Diagnostics

D.1 Pipeline Statistics

We aggregate the runtime behaviour of **CollabVR**+VBVR-Wan2.2 on Gen-ViRe and VBVR-Bench under the default configuration ($N_{\max}$=3, M=3, i.e. up to two re-generations per step).

Per-sample summary. Each Gen-ViRe sample takes on average 2.56 planning steps, 1.46 re-generations, 4.01 generated clips, and 6.79 VLM calls; on VBVR-Bench the same quantities drop to 1.48, 0.90, 2.38, and 3.85 respectively (Table 15), tracking the shorter trajectories on VBVR-Bench.

Table 15: Per-sample runtime summary on Gen-ViRe (72 samples) and VBVR-Bench (500 samples), under default config.

Quantity	Benchmark	Mean	Std	Median	Range
Steps taken	Gen-ViRe	2.56	0.76	3	[1, 3]
	VBVR-Bench	1.48	0.77	1	[1, 3]
Re-generations	Gen-ViRe	1.46	1.72	1	[0, 6]
	VBVR-Bench	0.90	1.26	0	[0, 6]
Generated clips	Gen-ViRe	4.01	2.18	3	[1, 9]
	VBVR-Bench	2.38	1.96	1	[1, 9]
VLM calls	Gen-ViRe	6.79	2.49	6	[2, 12]
	VBVR-Bench	3.85	2.69	2	[2, 12]

Step trajectory length. The planner terminates early at N=1 for 16.7% of Gen-ViRe samples (single-shot accept via task-complete signal), at N=2 for 11.1%, and reaches the cap N=3 for the remaining 72.2%. On VBVR-Bench the trajectory is much shorter (mean N=1.48 vs. 2.56 on Gen-ViRe; Table 16), consistent with VBVR-Bench's reasoning-heavy but visually constrained tasks where many problems admit a single closed-form action.

Table 16: Step trajectory length distribution on Gen-ViRe and VBVR-Bench.

N (steps used)	1	2	3 (cap)	Mean
Gen-ViRe share	16.7%	11.1%	72.2%	2.56
VBVR-Bench share	69.8%	12.8%	17.4%	1.48

Re-generation distribution. Table 17 reports the per-sample re-generation count distribution on both benchmarks. On Gen-ViRe, the distribution decays gradually from 0 to the cap of 6 regens (mean 1.46). On VBVR-Bench, the distribution is bimodal: a primary mode at 0 regens (58.6%) from samples the planner finishes in a single shot, and a secondary mode at 2 regens (26.4%) from multi-step trajectories that fail mid-way (mean 0.90, contributing 451 extra clips overall).

Table 17: Re-generation distribution per sample on Gen-ViRe and VBVR-Bench. Each row reports the share of samples whose total re-generation count across the trajectory falls in the given bin.

Re-gens / sample	0	1	2	3	4	5	6	Mean
Gen-ViRe (n=72)	48.6%	9.7%	12.5%	13.9%	9.7%	2.8%	2.8%	1.46
VBVR-Bench (n=500)	58.6%	7.6%	26.4%	2.0%	4.2%	0.2%	1.0%	0.90

Per-category breakdown on Gen-ViRe. Algorithmic samples consume the most compute per task (5.0 clips on average) due to a higher re-generation rate (2.50/sample), while Analogy and Planning converge fastest (Table 18).

Table 18: Per-category compute breakdown on Gen-ViRe.

Category	Avg N	Avg re-gens	Avg clips	Avg VLM calls
Abstract	2.25	1.75	4.00	6.67
Algorithmic	2.50	2.50	5.00	7.83
Analogy	2.00	1.08	3.08	5.67
Perceptual	3.00	1.50	4.50	7.50
Planning	2.75	0.67	3.42	6.17
Spatial	2.83	1.25	4.08	6.92

D.2 Verifier Run-time Behavior

We characterize the verifier on the same Gen-ViRe run.

Step-level outcomes. The verifier accepts **49.5%** of clips on the first attempt and recovers another **14.7%** through re-generation, for a **64.1%** final acceptance rate at an average retry depth of **0.57** per step. The remaining **35.9%** are rejected even after the maximum two re-generations and propagated downstream as the best-of-attempts clip.

Per-step rejection trend. Final-reject rate is comparable at the first two steps but rises sharply at the deepest step (Table 19). The step-3 spike reflects cumulative visual drift: each step's first frame is the previous step's last frame, so any mild deformation accepted earlier makes the next step harder to satisfy. The same pattern shows up in the diminishing returns of N beyond 3 on Gen-ViRe (Section 4.3) and even more sharply on Cosmos-Predict 2.5 (Appendix B.6). Overall, the verifier is exercised aggressively (about 37% of steps trigger at least one re-generation) without saturating the budget, supporting our default choice of M=3.

Table 19: Verifier final-reject rate by step index on Gen-ViRe.

Step index	Step 1	Step 2	Step 3
Final reject rate	31.9%	31.7%	46.2%

D.3 Cost Decomposition

We verify the "VLM compute is negligible relative to VGM compute" claim in Section 4.1 along two axes: GPU wall-clock time on the open-source backbone (VBVR-Wan2.2, single A100) and API $ cost on the closed-source backbone (Veo 3.1).

VLM per-call profile. We replayed representative Gemini 2.5 Pro calls on Gen-ViRe (planner: 1 image + task description; verifier: 1 instruction + 1 video clip) and recorded latency and token counts, averaged over 3 trials per call type.

Call type	Latency	Input tok	Output tok
Planner	10.15 s	800	186
Step verifier	10.60 s	2,724	117

Table 20: Per-call Gemini 2.5 Pro profile. The verifier's 2.7 K-token input is dominated by the inline video; the planner's 800-token input is the prompt template, the task description, and a single image.

Component	Per-sample wall-clock	Share
VBVR-Wan2.2 VGM (A100)	979.6 s (16.3 min)	93.5%
Gemini 2.5 Pro VLM (planner + verifier)	68.5 s (1.14 min)	6.5%
Total	1048.1 s (17.5 min)	100%
VGM / VLM ratio	$\approx 14\times$	

Table 21: Per-sample wall-clock decomposition (VBVR-Wan2.2 + **CollabVR** on a single A100, averaged over Gen-ViRe).

Closed-source: API cost. Veo 3.1 Fast is priced at \$0.15/s of generated video, so the per-sample 21.4 s reported in Table 1 costs \$3.21, compared to \$0.026 for the VLM, a $\sim 125\times$ ratio (Table 22).

Component	Per-sample \$	Share
Veo 3.1 VGM	\$3.210	99.2%
Gemini 2.5 Pro VLM (planner + verifier)	\$0.026	0.8%
Total	\$3.236	100%
VGM / VLM ratio	$\approx 125\times$	

Table 22: Per-sample API cost decomposition (Veo 3.1 + **CollabVR** on Gen-ViRe).

Per-sample VLM aggregate. With 2.56 planner and 4.01 verifier calls per sample (Section D.1), each Gen-ViRe sample triggers ~ 6.6 VLM calls totalling ~ 68.5 s of API wall-clock and ~ 13 K input tokens. At Gemini 2.5 Pro pricing (\$1.25/M input, \$10/M output), this works out to \$0.026 per sample.

Open-source: GPU wall-clock time. On a single A100 at 480p with 20 inference steps, VBVR-Wan2.2 generates the per-sample 17.8 s of video in approximately 980 s of wall-clock, while the VLM contributes only ~ 68.5 s, a $\sim 14\times$ ratio (Table 21).

On both axes the VGM dominates by an order of magnitude or more, justifying the convention in Section 4.1 of reporting Cost as VGM-generated seconds per sample as a faithful proxy for total compute on either deployment regime.

E Additional Qualitative Results

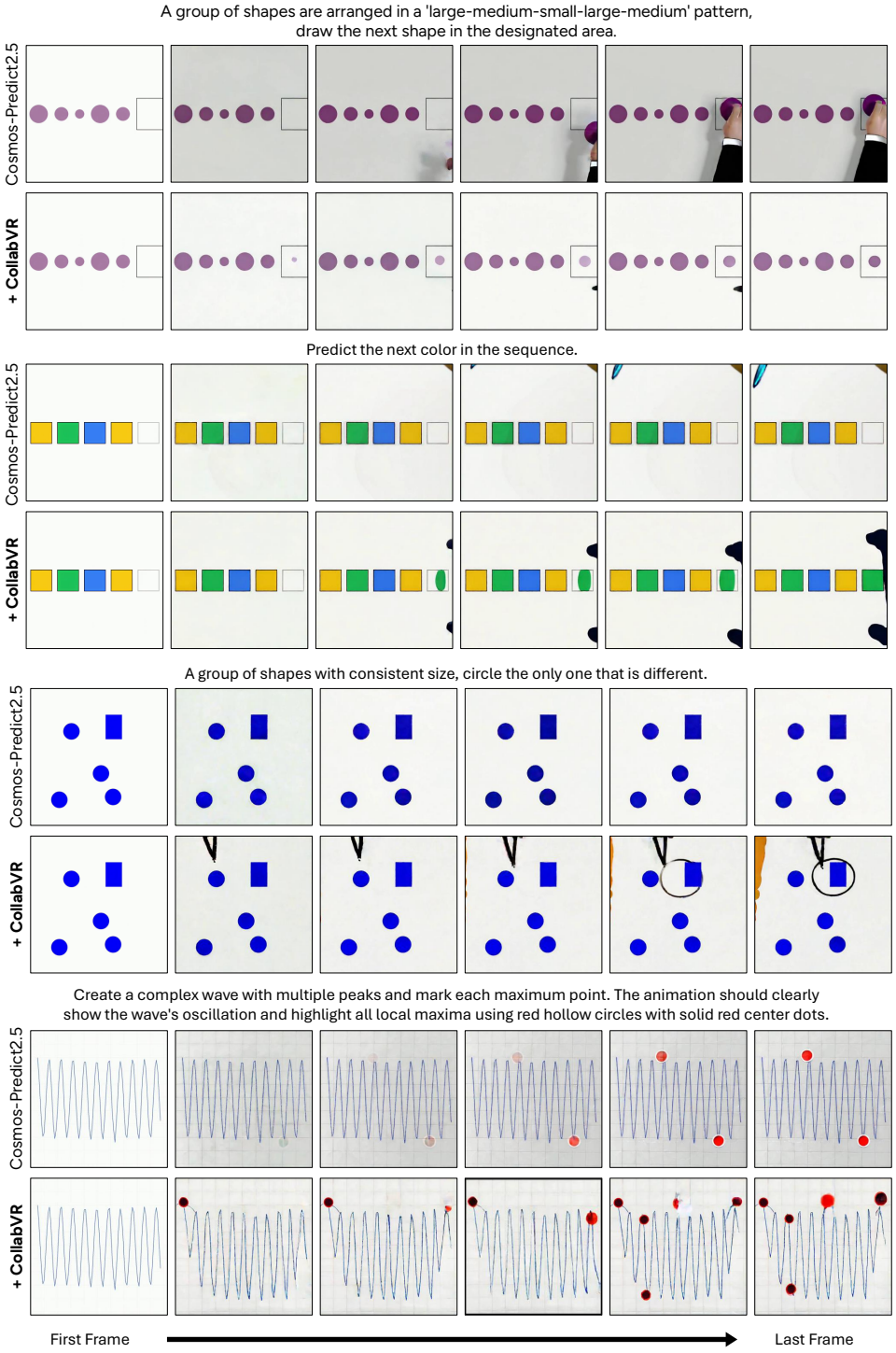


Figure 16: **CollabVR generalizes to Cosmos-Predict-2.5 on VBVR-Bench.** Each two-row block contrasts Cosmos-Predict-2.5 alone (top) with Cosmos-Predict-2.5+**CollabVR** (bottom) over a six-frame sequence (first → last frame).

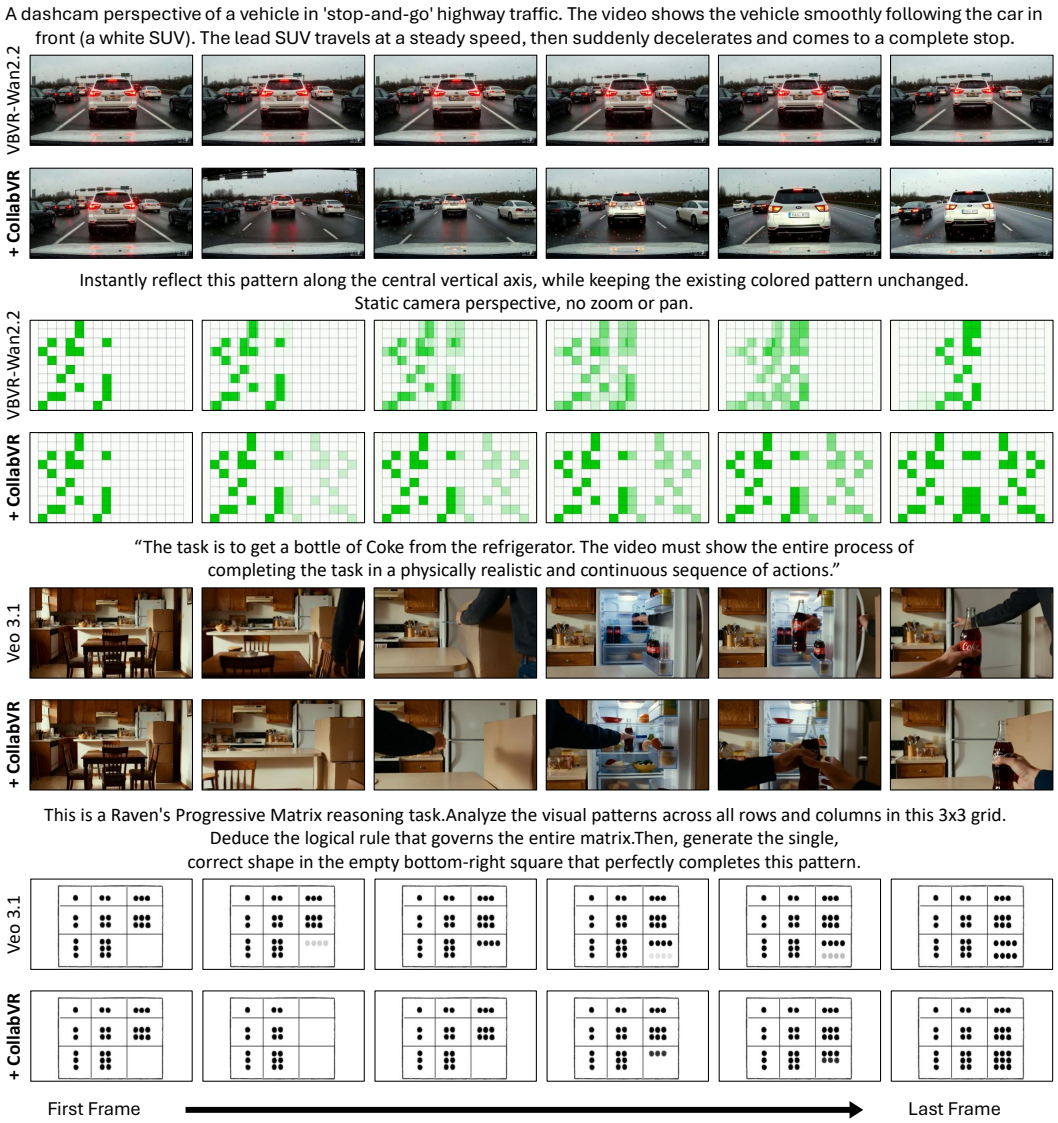


Figure 17: **CollabVR works across open- and closed-source VGMs on Gen-ViRe.** Each two-row block contrasts the base VGM (top) with **+CollabVR** (bottom); the upper two tasks use VBVR-Wan2.2 and the lower two use Veo 3.1.

E.1 Examples by VGM

CollabVR applies to a range of generators beyond VBVR-Wan2.2 through the same interface.

Generalization to another open-source VGM. Figure 16 shows that the same gains carry over to Cosmos-Predict-2.5: although the alone model exhibits its own characteristic failures (e.g., hallucinating an external hand or pen instead of acting on the canvas), pairing it with **CollabVR** consistently drives the generator toward the intended in-canvas action across all four tasks.

Open- and closed-source on Gen-ViRe. Figure 17 broadens the comparison to Gen-ViRe with an open-source (VBVR-Wan2.2) and a closed-source (Veo 3.1) generator, spanning real-world (dashcam, refrigerator) and symbolic-pattern tasks (mirror reflection, Raven's matrices). **+CollabVR** produces visibly more faithful executions than the base VGM in every case, supporting the results in Tables 1 and 2.

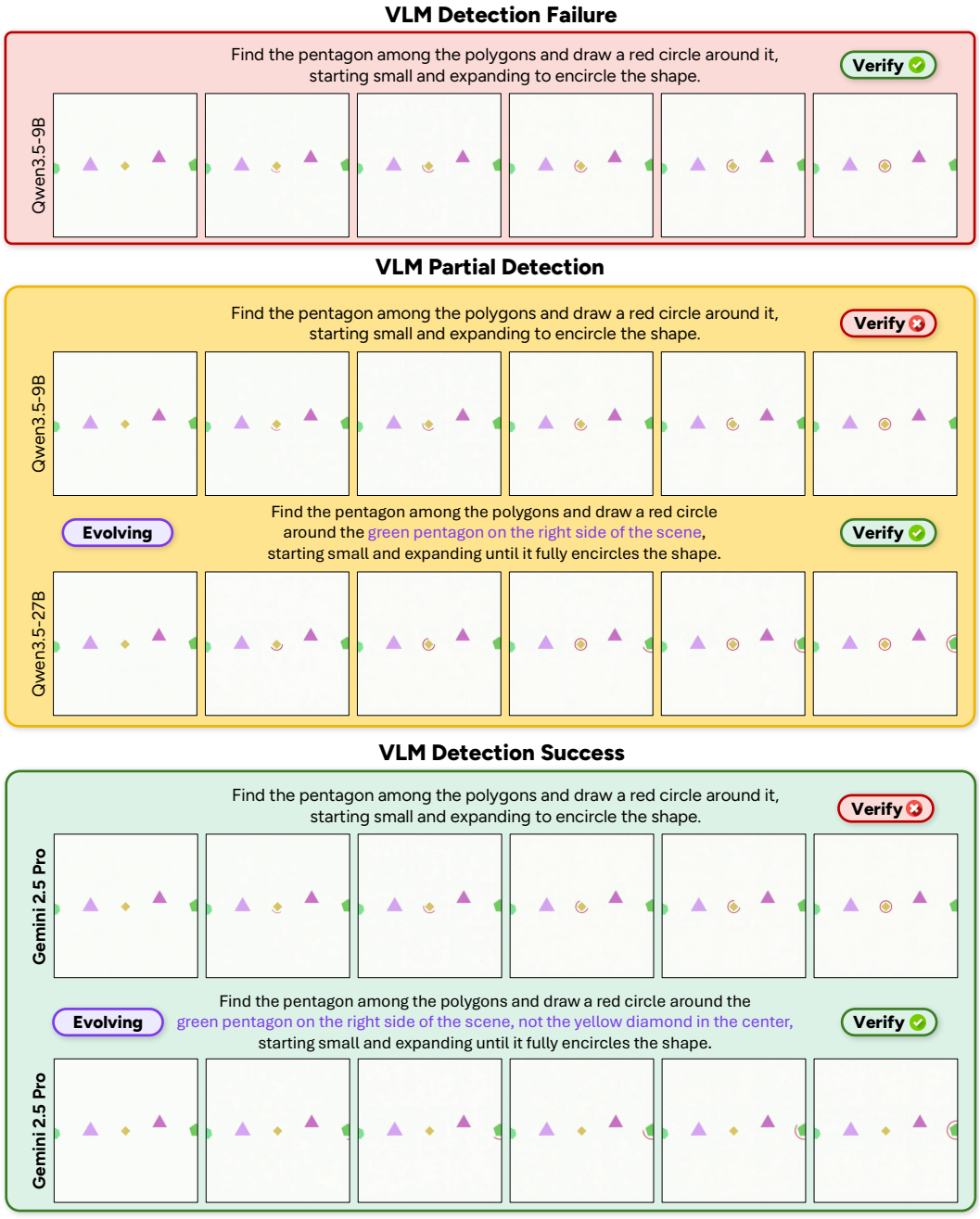


Figure 18: **Verifier VLM choice shapes the recovery loop on a single VBVR-Bench trace (VBVR-Wan2.2 as the VGM).** The same prompt is verified by Qwen3.5-9B (top, false-accept), Qwen3.5-27B (middle, evolution with a coarse positional cue), and Gemini 2.5 Pro (bottom, evolution that explicitly excludes the distractor).

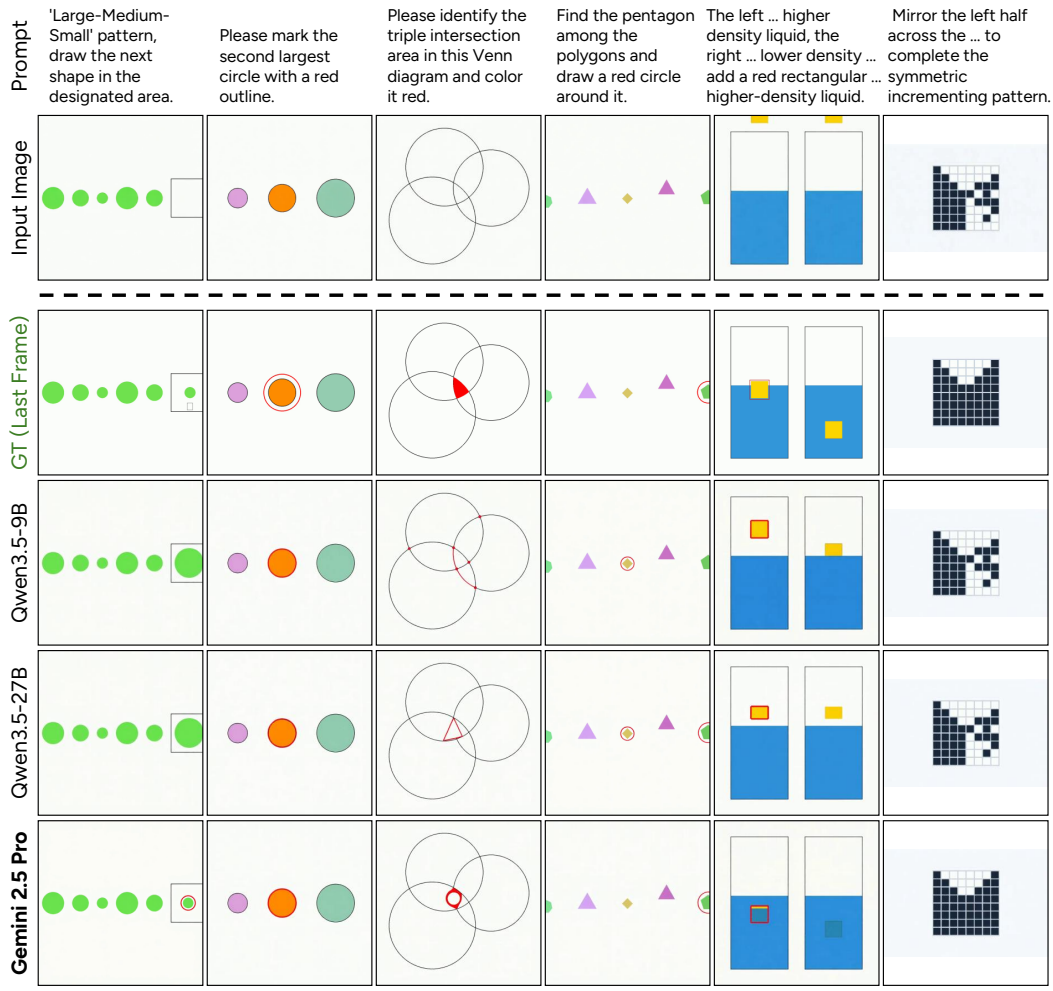


Figure 19: **Final +CollabVR outputs track verifier capability across VBVR-Bench tasks (VBVR-Wan2.2 as the VGM).** Each row shows the last frame produced when the verifier is Qwen3.5-9B, Qwen3.5-27B, or Gemini 2.5 Pro, with the input image and GT last frame at the top.

E.2 Examples by VLM

The verifier-quality gaps quantified in Section 4.4 translate into qualitatively distinct recovery behaviors that propagate to the final output.

A single trace under three verifiers. Figure 18 traces one regeneration loop on the prompt *"find the pentagon among the polygons and draw a red circle around it"*. Qwen3.5-9B false-accepts a mis-localized target, so no recovery is triggered. Qwen3.5-27B rejects but evolves the prompt with only a coarse positional cue (*"on the right side"*), and the regenerated clip is only partially correct. Gemini 2.5 Pro both rejects and articulates the precise distractor (*"not the yellow diamond in the center"*), and the regenerated clip recovers the correct red circle around the pentagon.

Aggregate behavior across tasks. Figure 19 confirms this is not a single-trace artifact: across multiple VBVR-Bench tasks, proximity of the final **+CollabVR** output to the GT last frame increases monotonically with verifier capability, mirroring the score ordering in Table 4.

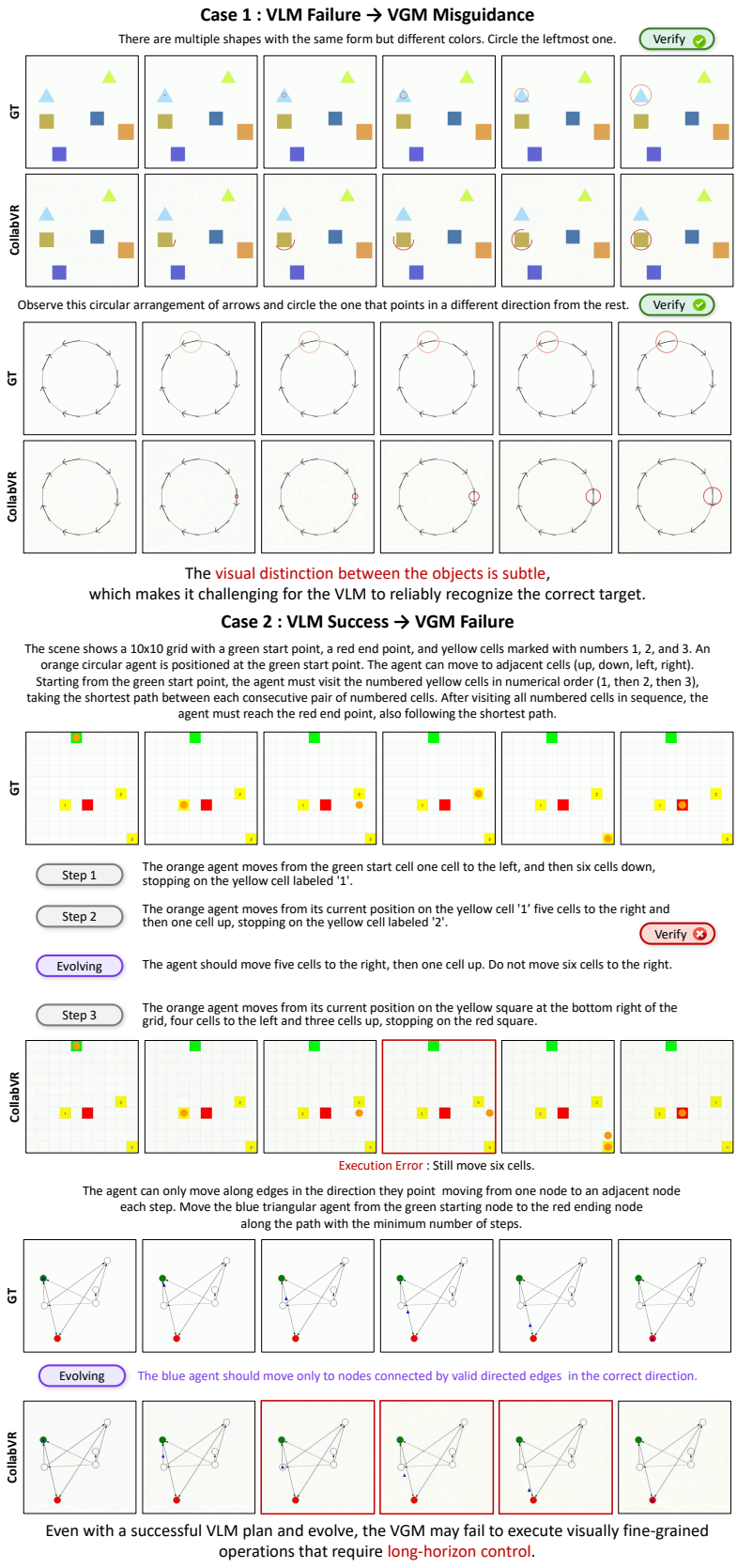


Figure 20: **Two distinct ceilings limit full CollabVR on VBVR-Bench.** *Case 1*: the VLM verifier fails to detect the issue, so no recovery is triggered. *Case 2*: the VLM diagnoses the failure and evolves the prompt correctly, but the VGM cannot execute the fine-grained operation.

E.3 Failure Cases

Figure 20 characterizes the residual failures of the full **CollabVR** pipeline, decomposing them into two complementary causes that test-time orchestration alone cannot resolve.

Case 1: VLM detection failure. The verifier itself fails to identify the issue, so no recovery is triggered and the incorrect clip is finalized. **CollabVR** circles a yellow square instead of the leftmost light-blue triangle, and circles an in-distribution arrow instead of the only outward-pointing one—both failures pass the verifier silently.

Case 2: VGM execution failure. The verifier correctly diagnoses the failure and the prompt evolves with the right semantic content (*"Do not move six cells to the right."*, *"valid directed edges in the correct direction"*), yet the VGM still misexecutes the fine-grained operation. Here the bottleneck has moved from the supervisor to the generator, and additional retries cannot help.

These two ceilings map onto the small-Δ categories of Figure 9: the symbolic-category gap (Knowledge, Analogy) is largely a Case 1 issue, while the residual Spatial / Transformation gap reflects a Case 2 ceiling. They point respectively to stronger VLM grounding and reasoning-oriented VGM training as complementary directions.

E.4 Partial vs. Full Re-generation: Maze Case Study

We additionally explore a recovery design specific to navigation-style tasks: on rejection, the VGM is re-invoked from the first failing frame f_τ rather than re-rolling from scratch, so that the prefix already produced correctly is preserved across attempts.

Figure 21 compares the two modes on a maze instance. Full re-generation fails to reach the goal across four independent attempts because each resampled trajectory discards previously accumulated progress. Partial re-generation from f_τ reuses the correct prefix and successfully converges on the goal cell, illustrating that test-time compute is much more effective when it targets the failed suffix rather than the entire trajectory.

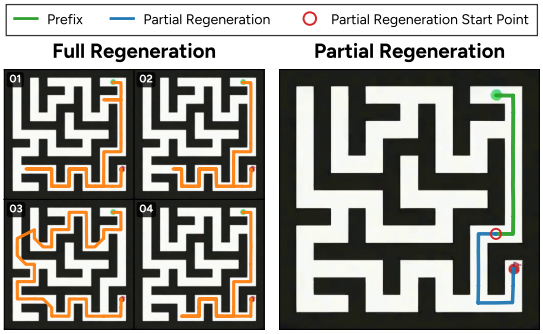


Figure 21: **Partial re-generation from f_τ outperforms full re-generation on a maze task.** Full re-generation is run for four independent attempts (left); partial re-generation reuses the correct prefix up to the first failing frame (right).

We position partial re-generation as an auxiliary mode applicable when the prefix carries useful information toward the goal, with precise failure detection as a promising future direction.

F Broader Impact

Positive impacts. Video reasoning is becoming a central AI capability for dynamic, temporally-grounded processes that static-image or pure-text reasoning cannot express, with applications spanning educational and scientific demonstrations, procedural walkthroughs, navigation in synthetic environments, and embodied-agent simulation. **CollabVR** contributes to this direction by showing that strong general-purpose VLMs and VGMs, while individually imperfect for goal-directed video generation, can be composed at step-level granularity into a closed-loop reasoning system that improves task fidelity without any additional training, and that the resulting trajectories are more interpretable and auditable than single-shot VGM outputs because each step's verification decision and prompt evolution are exposed as discrete artifacts. We see the next frontier of video reasoning coming less from a single ever-larger end-to-end generator than from richer inference-time collaborations between strong specialised models such as VLMs as step-level supervisors, VGMs as visual simulators, and other modality-specific reasoners, with **CollabVR** as one concrete instance of this design pattern.

Potential negative impacts. The framework inherits the misuse risks of the underlying generative models, including higher-fidelity synthetic video that could be used for deceptive or non-consensual content. **CollabVR** adds no new generative capability beyond the base VGM; it composes existing models for closed-loop reasoning rather than training new generators.

Mitigations. We use only off-the-shelf VGMs (Wan2.2, Cosmos-Predict-2.5, Veo 3.1) and VLMs (Gemini 2.5 Pro, Qwen3.5) under their original licenses and existing safeguards. The released artifacts (orchestration code and the human-annotated reliability benchmark) carry no novel high-risk generative capability beyond what is already publicly available.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Abstract and Section 1 state the three contributions (progressive planning, collaborative reasoning, matched-compute gains on both open- and closed-source VGMs); Sections 3 and 4 validate them on Gen-ViRe and VBVR-Bench.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 5 contains a Limitations paragraph covering VGM capability ceilings (Sec. 4.4, Sec. B.6) and verifier imperfection (Sec. 4.4), pointing to reasoning-oriented VGM training, finer-grained failure localization, and stronger verifier supervision as complementary directions.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper presents an empirical training-free framework; no theoretical results, theorems, or formal proofs are claimed.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Algorithm 1 fully specifies the closed-loop procedure. Appendix A.1 lists all hyperparameters ($N_{\max}$=3, M=3, sampling settings) for both **CollabVR** and baselines, and Appendix A.2 provides verbatim VLM prompt templates for planning, verification, and evolution. Implementation code is included in the supplementary zip.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: Source code (pipeline, Pass@1 / Pass@k baselines, and evaluation scripts) is provided as an anonymized zip in the supplementary material, with a README describing the environment, exact reproduction commands, and runtime estimates. Gen-ViRe [20], VBVR-Bench [32], and VBVR-Wan2.2 [32] are publicly available under their original licenses.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 4.1 and Appendix A.1 specify benchmark splits, baseline configurations (Pass@k, VideoTPO), VGM/VLM versions, sampling settings, and step/attempt budgets ($N_{\max}$=3, M=3). Our method is training-free, so no optimizer settings apply.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

Justification: All Gen-ViRe scores are averaged over three independent VLM-judge runs to mitigate judge stochasticity (Sec. 4.1); the VBVR-Bench rule-based protocol is deterministic, so a single-pass evaluation yields the true score without variance to report. We did not run multiple full pipeline rollouts per sample due to API budget constraints (closed-source VGMs/VLMs) and server runtime limits (open-source VGMs), which together exceeded the resources available within our timeline to demonstrate full statistical significance. To partially compensate, we evaluate at scale on VBVR-Bench and across diverse reasoning domains on Gen-ViRe (six categories: Spatial, Planning, Perceptual, Abstract, Algorithmic, Analogy), which together provide empirical generalizability across both sample-size and domain-coverage axes.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix D.3 (Cost Decomposition) reports per-call and per-sample VGM/VLM compute in seconds and USD for **CollabVR** and all baselines, with GPU specifications for open-source VGMs and API rate-limit settings for closed-source models. Wall-clock totals for the full benchmark sweep are also tabulated.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics and the work conforms in every respect. User study participants were compensated through Prolific and provided informed consent (Appendix B.1); no identifiable personal data was collected.

Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: Appendix F discusses positive impacts (educational/scientific applications of reliable step-level video reasoning, more interpretable trajectories) and potential negative impacts (synthetic-video misuse risks inherited from the underlying VGMs), with mitigations through off-the-shelf model composition rather than training new generative capability.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper releases only orchestration code over existing third-party VGMs (Wan2.2, Cosmos-Predict-2.5, Veo 3.1) and VLMs (Gemini, Qwen3.5), each governed by their own safeguards and license terms. No new high-risk generative models or scraped datasets are released.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All third-party assets are cited and used under their original licenses: Wan2.2 [32] (Apache 2.0), Cosmos-Predict-2.5 [25] (NVIDIA Open Model License), Veo 3.1 [7] and Gemini 2.5 Pro [6] (Google API Terms of Service), Qwen3.5 [27] (Apache 2.0), Gen-ViRe [20], and VBVR-Bench [33]. Versions are listed in Appendix A.1.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: We release a human-annotated reliability benchmark (Sec. 4.4) covering plan-depth, clip-level verification, and evolution-routing axes, with annotation guidelines and a JSON schema bundled in the anonymized supplementary zip alongside the orchestration code.

    Guidelines:

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

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

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

Answer: [Yes]

Justification: Appendix B.1 (User Study) describes the Prolific blind side-by-side study with n=40 participants (12 tasks each), the full instruction text, and a screenshot of the task interface. Total compensation across the study was approximately £80 (about £2 per submission), exceeding the UK minimum wage given the typical task duration.

Guidelines:

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

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

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

Answer: [N/A]

Justification: The user study involved only blind preference rating of generated videos, with no identifiable personal data collected beyond Prolific's anonymous participant IDs. Given the minimal-risk task and Prolific's standard ethics process, formal IRB review was not required at our institution.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: VLMs (Gemini 2.5 Pro by default; Qwen3.5-9B/27B as alternatives) are central to the proposed method, serving as planner and verifier in the closed-loop framework (Sec. 3). All VLM roles, prompts, and computational costs are explicitly described in Sec. 3 and Appendix A.2.

Guidelines:

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