SGA: Plug&Play Geometric Verification for Educational Video Synthesis

Anonymous ACL submission

Abstract

Recent work leverages Large Language Models (LLMs) to generate executable code for pedagogical animations using libraries such as Manim. However, ensuring spatial correctness and visual legibility remains challenging, as existing frameworks emphasize pedagogical content while overlooking geometric occlusions. We propose the Symbolic Geometric Agent (SGA), a plug-and-play module for code-centric animation pipelines that intercepts LLM-generated code, performs partial execution to extract symbolic scene graphs, and applies targeted refinement when spatial conflicts are detected. We further introduce the Manim Visual Quality Score (MVQS), a deterministic rendering-free proxy for spatial integrity. Experiments on the MMMC-Code benchmark across four LLM backbones and two agentic pipelines show that SGA achieves a peak MVQS of 73.11 (Code2Video + GPT-5.1), corresponding to a 16.1% relative improvement over the raw baseline, and improves MVQS in 7 of 8 backbone $\times$ pipeline configurations.

1 Introduction

Video has become a dominant medium for instructional content in modern digital education. In mathematics and science, structured animations require precise spatial organization, clear visual hierarchy, and coherent temporal transitions to effectively communicate abstract concepts (Khalil et al., 2024; Navarrete et al., 2025; Clark and Mayer, 2023; Mayer, 2024). As demand for scalable educational content grows, research increasingly focuses on automating the production of explanatory videos.

Recent advances in Large Language Models (LLMs) have enabled language-to-program generation, allowing natural language prompts to be translated into executable programs for visual content creation. Moving beyond direct synthesis in pixel space, emerging code-centric paradigms generate structured animation programs, such as Python scripts for the Manim engine (Developers, 2020), which are then rendered into educational videos (Chen et al., 2025). This approach offers substantial advantages in controllability, reproducibility, and semantic transparency compared to diffusion-based or pixel-level video generation (Lin et al., 2025; Shah et al., 2025; Huang et al., 2026). Representative systems such as Code2Video (Chen et al., 2025) and TheoremExplainAgent (Ku et al., 2025) demonstrate that LLM-driven pipelines can produce long-form, coherent educational animations through multi-agent planning and refinement. Figure 1 illustrates the limitations of current approaches and the motivation for our framework.

Despite these advances, ensuring spatial correctness and visual legibility remains challenging. Educational animations require strict geometric consistency, in which text does not overlap, equations remain aligned, and objects preserve their scale relationships to maintain readability. Empirical analyses of state-of-the-art code-centric systems reveal that spatial layout remains the weakest component in automated generation. For instance, TheoremExplainAgent reports that element layout receives the lowest evaluation scores among all assessed criteria, even when factual accuracy and logical coherence are high (Ku et al., 2025). These issues are not merely aesthetic; in educational contexts, poor spatial organization increases cognitive load and can impair knowledge transfer (Clark and Mayer, 2023; Leiker et al., 2023).

Current approaches primarily rely on post-hoc perceptual evaluation using Vision-Language Models (VLMs) to critique rendered video frames (Chen et al., 2025). While effective for high-level assessment, VLM-based critique operates in pixel space, requires expensive full-video rendering, and produces coarse, non-deterministic feedback. Crucially, such evaluation lacks deterministic spatial verification. This reveals a methodological mismatch: when visual content is gener-

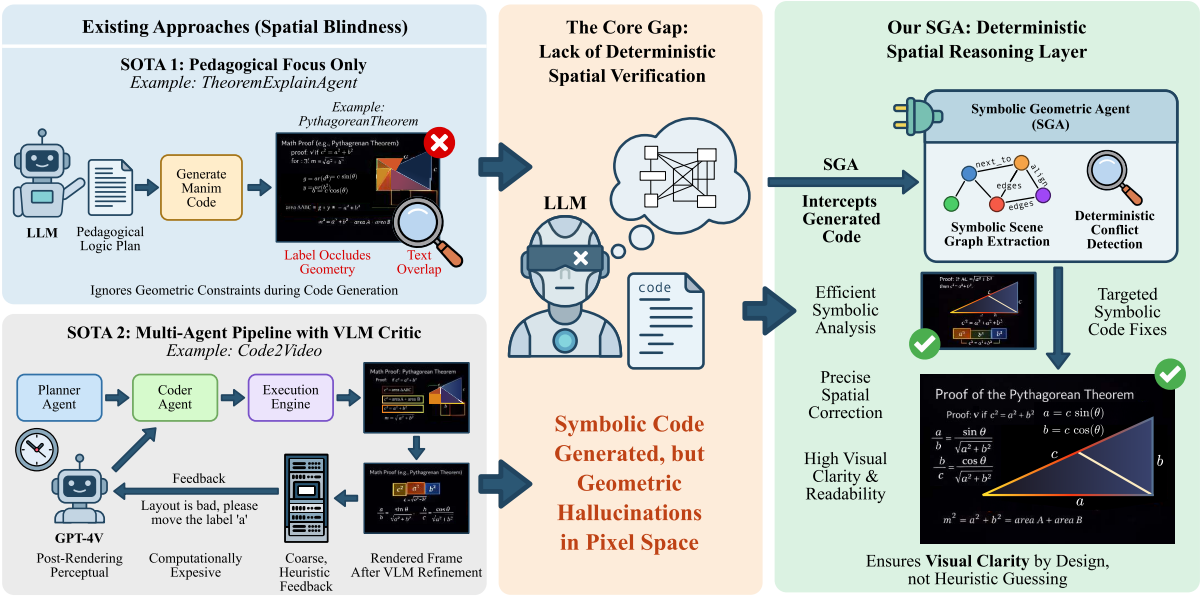

Figure 1: Current SOTA agents produce syntactically valid but visually occluded animations. (Left) *TheoremExplainAgent* (Ku et al., 2025) neglects spatial constraints, while *Code2Video* (Chen et al., 2025) relies on stochastic, computationally expensive VLM feedback that fails to resolve fine-grained clashes. (Center) This lack of deterministic grounding induces "geometric hallucinations" in pixel space. (Right) Our SGA enables symbolic scene graph extraction and targeted refinement, ensuring spatial integrity by design prior to rendering.

ated from structured code, verification should operate in the same symbolic domain. To address these limitations, we introduce the Symbolic Geometric Agent (SGA), a plug-and-play symbolic verification framework for code-centric educational video generation. We further introduce the Manim Visual Quality Score (MVQS), a deterministic rendering-free metric for spatial integrity. In summary, our contributions are threefold:

1. We introduce the Symbolic Geometric Agent (SGA), a plug-and-play verification layer for code-centric animation frameworks that constructs symbolic scene graphs and computes geometric descriptors, such as axis-aligned bounding boxes and object distances, enabling deterministic spatial conflict detection and targeted refinement without modifying the generation pipeline.

2. We introduce the Manim Visual Quality Score (MVQS), a deterministic metric that evaluates visual layout quality without video rendering or VLM-based critique.

3. We conduct extensive experiments on the MMMC-Code benchmark across four LLM backbones and two agentic pipelines, showing that SGA consistently improves spatial layout quality and outperforms VLM-based critique in spatial layout evaluation.

2 Related Work

Code-Centric Educational Video Generation. Traditional generative video research has largely focused on end-to-end models operating in pixel or latent spaces, including diffusion-based (Gupta et al., 2024; Ho et al., 2022; Xing et al., 2024; Lian et al., 2023) and autoregressive approaches (Deng et al., 2024; Khachatryan et al., 2023). However, such models often struggle with the long-form reasoning, controllability, and spatial consistency required for instructional content. Recent advances in large language models have enabled code-centric generation paradigms in which LLMs produce executable animation programs through multi-step reasoning and iterative refinement. Frameworks such as ReAct (Yao et al., 2022) demonstrate how language models can interleave reasoning and external actions (Shinn et al., 2023; Madaan et al., 2023; Hong et al., 2024), while specialized systems such as Code2Video (Chen et al., 2025) and TheoremExplainAgent (Ku et al., 2025) show that structured program generation enables controllable and semantically coherent educational animations. Despite these advances, current systems primarily rely on internal LLM reasoning and lack mechanisms to verify the geometric consistency of generated layouts before rendering. Our work addresses this limitation through symbolic geometric verification directly in the code generation pipeline of systems

such as Code2Video and TheoremExplainAgent.

Automated Evaluation of Visual Quality. Evaluation metrics for generative visual models have traditionally focused on pixel-level similarity or embedding-based alignment, often using measures such as CLIP (Radford et al., 2021). More recent systems employ Vision-Language Models (VLMs) as automated evaluators for generated images and videos (Chen et al., 2025; Lee et al., 2024; Chen et al., 2024; Liu and Zhang, 2025), providing high-level assessments of visual quality and instruction alignment. However, VLM-based evaluation operates in pixel space and does not directly capture geometric violations in structured animation layouts. As a result, these approaches often provide coarse feedback that is difficult to translate into concrete code corrections. Our proposed MVQS metric instead evaluates layout quality using symbolic geometric descriptors derived from the animation scene representation, enabling feedback for refinement.

3 Proposed Method

Figure 2 illustrates the proposed Symbolic Geometric Agent (SGA), a framework designed to mitigate the spatial blindness inherent in current code-generation pipelines for educational video generation. Unlike traditional VLM-based critics that rely on computationally expensive pixel-level video rendering and stochastic inference, our approach performs deterministic geometric verification directly on symbolic scene representations extracted via partial execution. Due to its modular and framework-agnostic architecture, SGA serves as a plug-and-play verification layer for educational video generation frameworks. It enforces spatial layout consistency without requiring architectural modifications or fine-tuning of the underlying LLMs.

3.1 Plug-and-Play Architecture and Interception Flow

The SGA operates as a non-invasive verification layer, treating animation synthesis as an iterative code-space optimization problem. Given a target topic T, a generative model $\mathcal{L}$ produces an initial script $C^{(0)}$. The SGA intercepts the execution flow and executes a four-stage refinement loop at each iteration i: (1) Symbolic Extraction, where $C^{(i)}$ undergoes low-latency partial execution to derive a symbolic scene graph $\mathcal{S}^{(i)}$, bypassing full video rendering overhead (Sec. 3.2); (2) Deterministic Conflict Detection, wherein $\mathcal{S}^{(i)}$ is evaluated for spatial violations via exact Axis-Aligned Bounding Box (AABB) analysis, replacing stochastic VLM heuristics with reproducible geometric verification (Sec. 3.3); (3) Actionable Feedback, involving a compiler $\mathcal{F}$ that translates violations into a structured diagnostic report $F^{(i)}$ containing pre-computed fix vectors and canonical Manim correction templates (Sec. 3.5); and (4) Targeted Refinement, where the LLM performs precise, line-level edits conditioned on the symbolic feedback:

$$C^{(i+1)} = \mathcal{L}(T, C^{(i)}, F^{(i)}) \tag{1}$$

Refinement terminates after a fixed budget of n iterations, effectively bounding computational overhead while allowing the LLM to iteratively incorporate symbolic grounding constraints. A summarized version of the algorithm is provided in the supplemental material (Sec. 6).

3.2 Symbolic Scene Graph Extraction

SGA avoids high-latency rendering by repurposing the Manim runtime as a symbolic execution engine. The system parses source code into an Abstract Syntax Tree (AST) and instrumentally rewrites the `construct()` method to wrap each statement in a granular `try-except` block. This fault-tolerant execution isolates runtime exceptions, such as invalid attribute calls common in LLM outputs, allowing the agent to skip faulty instructions and extract spatial primitives from the remaining valid code.

Geometric Formalization. For each Manim object (`mobject`) m_j extracted into the set of all scene objects $\mathcal{M}$, the SGA derives a deterministic record $\mathcal{R}_j = \langle \phi_j, \mathbf{c}_j, \mathrm{bb}_j, \psi_j, \ell_j \rangle$. (1) Semantic label $\phi_j \in \Phi$ is assigned from a four class taxonomy $\Phi = \{\textsc{Text}, \textsc{Geom}, \textsc{Mark}, \textsc{Deco}\}$(Ericson, 2004). (2) Spatial geometry is defined by centroid $\mathbf{c}_j = (c_j^x, c_j^y)$ and dimensions (w_j, h_j), yielding Axis Aligned Bounding Box: $x_j^{\min/\max} = c_j^x \pm w_j/2$ and $y_j^{\min/\max} = c_j^y \pm h_j/2$. (3) Source grounding ψ_j maps the runtime object address to its Python identifier via static AST analysis(Gong et al., 2024); for variables unresolvable through static assignment (e.g., objects in loops), we fallback to a deterministic positional scheme. (4) Temporal provenance ℓ_j captures the source line number (`creation_lineno`). This multidimensional mapping ensures that diagnostic feedback references the LLM's own symbolic namespace, facilitating localized, line-level edits.

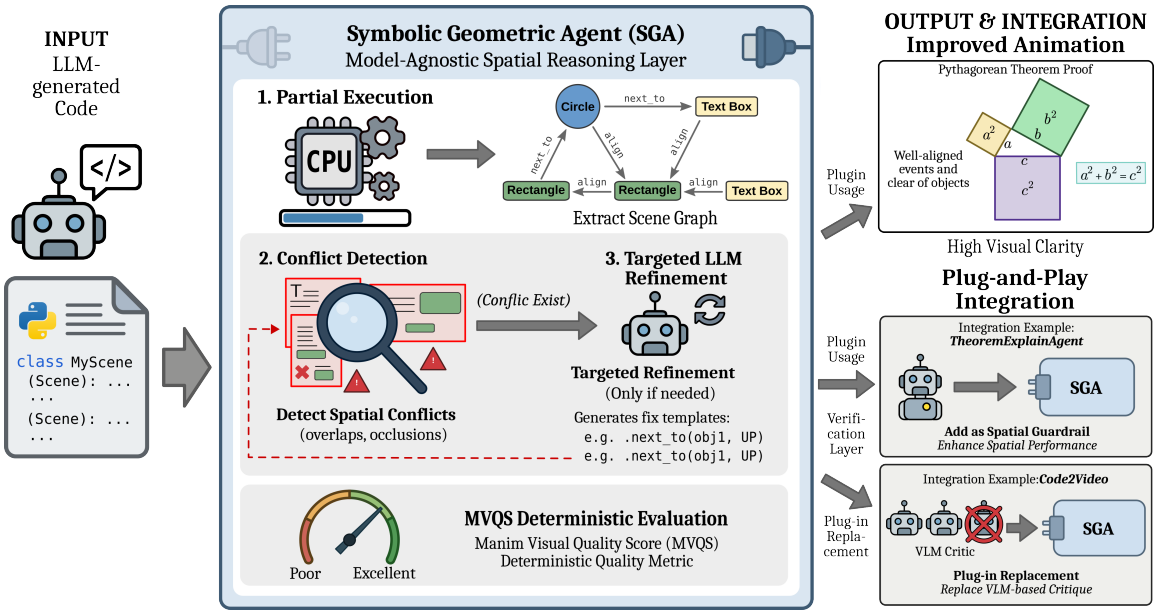

Figure 2: The SGA acts as a neuro-symbolic interceptor for spatial-linguistic grounding. (Left) The pipeline captures LLM-generated code pre-rendering. (Center) The SGA engine extracts a symbolic scene graph via partial execution, identifying spatial conflicts through deterministic AABB analysis. Violations prompt template-based refinement, quantified by the Manim Visual Quality Score (MVQS). (Right) SGA serves as a plug-and-play spatial guardrail, replacing stochastic VLM critics in frameworks like Code2Video and TheoremExplainAgent.

3.3 Deterministic Spatial Conflict Detection

Geometric Conflict Formalization. SGA defines spatial integrity as the absence of unintended occlusions between `mobject` pairs. For any pair $(m_i, m_j) \in \mathcal{M}^2$ where $i \neq j$, the intersection area I_{ij} is computed via AABB:

$$I_{ij} = \max(0, \Delta x_{ij}) \times \max(0, \Delta y_{ij}), \quad (2)$$

where $\Delta x_{ij} = \min(x_i^{\max}, x_j^{\max}) - \max(x_i^{\min}, x_j^{\min})$, and similarly for Δy_{ij}. A conflict is registered if $I_{ij} > \epsilon$ (default $\epsilon = 0$). Let $V^{(i)}$ denote the set of detected spatial violations at iteration i, serving as input to the feedback compiler. To distinguish severe occlusions from incidental contact, we compute asymmetric coverage ratios $\rho_{ij}^{\text{self}} = I_{ij}/A_i$ and $\rho_{ij}^{\text{other}} = I_{ij}/A_j$, where $A_k = w_k \times h_k$ is the bounding box area. Conflict severity is discretized as HIGH ($\max(\rho) > 0.30$), MEDIUM (> 0.10), or LOW. For each confirmed conflict, SGA calculates the Minimum Separation Vector (MSV), defined as the shortest axis-aligned displacement (d_x, d_y) required to satisfy $I_{ij} \leq \epsilon$. This vector is provided to the feedback compiler, grounding LLM refinement in explicit geometry and coordinates rather than qualitative suggestions.

Semantic and Spatio-Temporal Filtering. To preserve intentional compositions, we apply a two-stage filter. Stage 1 (Contextual) matches the animation description against a vocabulary of placement (e.g., "on, root"), motion (e.g., "slide"), and geometric (e.g., "axis") keywords. Stage 2 (Structural) performs static analysis to detect patterns such as decorative enclosures (P1), annotation pointers (P2), explicit centering calls (P3), container geometries (P4), and mathematical braces (P5). Conflicts matching these patterns are treated as intentional or pedagogical and suppressed. Finally, for dynamic scenes, the SGA performs Spatio-Temporal Sampling across $K = 11$ frames. Conflicts at $t \in \{0, 1\}$ are flagged as persistent errors requiring patches, while mid-trajectory overlaps ($0 < t < 1$) are classified as transient and ignored to preserve intended animation paths. Only persistent and problematic violations proceed to the feedback compiler.

3.4 Manim Visual Quality Score (MVQS)

Leveraging the symbolic extraction pipeline described above, we introduce MVQS. Unlike traditional vision-based assessments, MVQS eliminates the need for full video synthesis by computing a weighted linear combination of three composite dimensions derived from the scene graph $\mathcal{S}^{(i)}$.

$$\text{MVQS} = w_{\text{sp}} S_{\text{sp}} + w_{\text{ly}} S_{\text{ly}} + w_{\text{ed}} S_{\text{ed}}, \quad (3)$$

where $\{w_{\text{sp}}, w_{\text{ly}}, w_{\text{ed}}\}$ are set to $\{0.5, 0.2, 0.3\}$. All metrics are normalized to $[0, 1]$.

Spatial Correctness. Denoted by S_{sp}, this dimension penalizes errors that fundamentally im-

pair readability. It combines four complementary criteria: (1) *Overlap Penalty*, which penalizes problematic conflicts according to their severity; (2) *Canvas Boundary*, which measures the fraction of objects contained within the viewport $[-W/2, W/2] \times [-H/2, H/2]$; (3) *Legibility*, defined as the fraction of objects exceeding a minimum scale threshold δ (default 0.15 units); and (4) *Separation*, which measures the normalized Euclidean distance between object centroids relative to the canvas diagonal.

Composition and Pedagogy. Layout quality, denoted by S_{ly}, evaluates how effectively visual elements occupy and distribute across the canvas. It combines: (1) *Density*, which measures scene coverage $\alpha = (\sum w_j h_j)/(WH)$ and favors balanced occupancy within $[0.10, 0.35]$; and (2) *Distribution*, defined as the occupancy fraction of a 4×3 canvas grid. Educational utility, denoted by S_{ed}, measures the pedagogical clarity of the animation. It combines: (1) *Labeling Completeness*, the fraction of geometry or marker objects associated with `MathTex` identifiers via substring matching; and (2) *Animation Quality*, the fraction of animations free of persistent conflicts at resting frames $t \in \{0, 1\}$.

Aggregation. Each composite dimension aggregates its corresponding component metrics using weighted averaging. Specifically, S_{sp} uses weights $\mathbf{w}_{\text{sp}} = [0.50, 0.25, 0.15, 0.10]$, while S_{ly} and S_{ed} use $\mathbf{w}_{\text{ly}} = [0.40, 0.60]$ and $\mathbf{w}_{\text{ed}} = [0.40, 0.60]$, respectively. The final MVQS score is then computed via Eq. 3. We assign the largest weight to S_{sp} because occlusion errors constitute the primary barrier to linguistic and visual grounding. MVQS is intended as a geometry-grounded engineering proxy rather than a direct substitute for human judgment; additional discussion, cross-metric analysis, weight sensitivity analysis, and full sub-score decompositions are provided in the supplemental material (Sec. 2).

3.5 Feedback Compilation and Targeted Refinement

The feedback compiler $\mathcal{F}$ translates problematic conflict pairs $\mathcal{P}$ into a structured diagnostic report $F^{(i)}$ containing both human-readable feedback for the LLM prompt and machine-readable patch metadata. For dynamic scenes, persistent conflicts at resting frames $t \in \{0, 1\}$ are distinguished from transient motion overlaps to avoid disrupting valid animation trajectories.

Deterministic Fix Strategies. Based on AABB analysis and the computed MSV, $\mathcal{F}$ applies geometric correction strategies including repositioning objects to safe viewport regions, translation via `.shift()` operations, rescaling for high-severity occlusions, temporal corrections inserted before relevant `self.play()` calls, and parameter adjustment for relative placement operators such as `.next_to()`. These operations preserve chained positioning logic while minimizing downstream cascade failures.

Surgical Patching. Refinement is performed through localized patch prompts containing a ± 5 line source context window centered on the `creation_lineno`, together with the precomputed MSV and canvas coordinate reference. This constrains the LLM to localized edits without global script regeneration. Patches are validated via AST parsing, with failed parses triggering rollback to the prior version.

4 Experiments and Results

We evaluate SGA as a plug-and-play spatial verification layer for educational video generation. The agent is integrated into two SOTA frameworks: TheoremExplainAgent (TEA) (Ku et al., 2025) and Code2Video (C2V) (Chen et al., 2025), which serves as a vision-centric baseline that employs a VLM as a judge agent to provide feedback on rendered frames. Our evaluation framework comprises three axes: (1) deterministic spatial symbolic integrity quantified via MVQS; (2) perceptual quality assessed through VLM-as-Judge protocols; and (3) system computational efficiency.

Benchmark and Backbones. We utilize the MMMC-Code benchmark (Chen et al., 2025), containing 117 complex instructional topics across calculus, linear algebra, and physics. This benchmark is uniquely suited for SGA evaluation as it requires simultaneous optimization of pedagogical reasoning and precise geometric layout. To evaluate robustness across diverse LLM backbones, we employ four models: GPT-5.1, GPT-5 mini, Claude 4.6 Sonnet, and Gemini 3.0 Flash, each with a default temperature.

Evaluation Protocol. (1) *Perceptual Aesthetics:* To model visual quality while accounting for variability in automated assessment, we adopt the *VLM-as-Judge* protocol (Chen et al., 2025) using Gemini

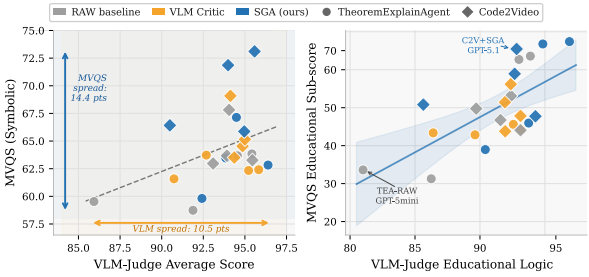

Figure 3: Perceptual vs. symbolic evaluation across 24 configurations. (Left) MVQS exhibits substantially larger variation than VLM-Judge scores, indicating limited perceptual sensitivity to geometric inconsistencies. (Right) Educational sub-scores show stronger alignment between perceptual and symbolic evaluation.

3.0 Flash. The model performs zero-shot qualitative inference across four semantic dimensions, each quantified on a 100-point scale: Educational Logic (EL) for temporal coherence, Aesthetic Tone (AT) for visual saliency, Layout Flow (LF) for spatial distribution, and Visual Clarity (VC) for morphological stability. (2) *Symbolic Metrics (MVQS):* We compute our proposed metric using the deterministic engine described in Sec. 3.4.

4.1 Quantitative Results

Table 1 summarizes performance across the 117 topics of the MMMC-Code benchmark. SGA improves MVQS in 7 of 8 backbone $\times$ pipeline configurations, with the strongest gains in the Code2Video pipeline where positional errors are denser. The sole exception is TheoremExplainAgent + Claude 4.6 Sonnet, where MVQS declines marginally from 63.84 to 62.83 (-1.01 pts); we attribute this to Claude's already-structured output reducing the number of patchable violations, leaving S_{sp} largely unchanged while S_{ly} contracts slightly due to repositioning. Notably, these improvements are largely invisible to VLM-Judge aesthetics scores, motivating both the design of SGA and the introduction of the MVQS metric.

A key observation is the divergence between VLM-Judge aesthetics and MVQS. In the TheoremExplainAgent-RAW setting, Claude 4.6 Sonnet achieves an aesthetics score of 95.42 while obtaining only 63.84 on MVQS, a gap exceeding 31 points. Across all conditions, SGA and the VLM Critic differ by fewer than 1.5 points on every VLM sub-score, yet exhibit substantially larger differences in MVQS. This suggests that aesthetics-based evaluation alone is insufficient to detect spatial correctness and that perceptual critics remain largely insensitive to geometric inconsistencies.

In TheoremExplainAgent, scenes are generally pedagogically structured but exhibit object-level coordinate hallucinations. SGA mitigates these through deterministic geometric corrections, increasing the MVQS of Gemini 3.0 Flash from 63.67 to 67.14, driven primarily by gains in S_{ed} (+4.08). In contrast, Code2Video exhibits denser positional errors, amplifying the benefit of symbolic correction. Here, the VLM Critic yields only modest MVQS gains for GPT-5.1 (62.97 to 64.54), whereas SGA increases performance to 73.11, corresponding to a 16.1% relative improvement over the raw baseline and the highest MVQS observed in our experiments. Overall, SGA improves S_{ed} over the VLM Critic by 15.0 points in TheoremExplainAgent and 10.5 points in Code2Video, and increases overall MVQS by 1.50 and 4.43 points, respectively. These gains arise from deterministic MSV-based corrections without iterative sampling or perceptual scoring. The results suggest that symbolic geometric reasoning offers a more precise and reproducible mechanism for correcting spatial layout than perceptual critique alone.

4.2 Perceptual vs. Symbolic Evaluation

Figure 3 plots VLM-Judge scores against MVQS across all 24 configurations. Panel (a) shows that the two axes are largely orthogonal ($\rho = +0.24$, $p = 0.27$): VLM scores compress into a 10.5-point band (85.9–96.4) while MVQS separates the same configurations over 14.4 points (58.7–73.1), with SGA occupying the upper range. This confirms that perceptual critics are insensitive to the geometric violations MVQS detects, motivating symbolic evaluation as a complementary axis. Panel (b) reveals the sole significant bridge: the MVQS educational sub-score S_{ed} correlates moderately with VLM Educational Logic ($\rho = +0.58$, $p = 0.003$), as both capture whether scene semantics are coherent. The remaining sub-scores (S_{sp}, S_{ly}) show no significant correlation with any VLM dimension ($|\rho| \leq 0.20$), confirming that spatial correctness is precisely what pixel-level evaluation misses.

4.3 Qualitative Analysis

We assess the effectiveness of SGA in mitigating coordinate-level failures by comparing rendered keyframes. Figure 4 illustrates the transition from stochastic coordinate hallucinations, where the generative backbone lacks grounding in the final canvas state, to deterministic geometrically verified layouts. Figure 4 shows representative outputs

Table 1: Comparative analysis of Code2Video and TheoremExplainAgent across diverse LLM backbones. Performance is measured using automated perceptual metrics (VLM-as-Judge) and deterministic symbolic metrics (MVQS). All values are on a 0–100 scale; values in bold indicate the best performance per backbone group.

Backbone Model	Aesthetics (VLM Judge ↑)					MVQS Metrics (Symbolic ↑)			
	EL	AT	LF	VC	Avg	S_{sp}	S_{ly}	S_{ed}	MVQS
TheoremExplainAgent-RAW									
GPT-5.1	86.25	93.38	97.25	90.62	91.88	67.73	64.48	31.27	58.74
GPT-5 mini	81.00	86.20	92.00	84.50	85.93	68.39	**76.48**	33.61	59.53
Claude 4.6 Sonnet	93.88	94.12	98.54	95.12	95.42	62.17	65.71	63.59	63.84
Gemini 3.0 Flash	93.00	93.40	98.00	93.80	94.55	63.84	64.42	62.71	63.67
TheoremExplainAgent + VLM Critic									
GPT-5.1	86.40	90.50	97.55	88.51	90.74	69.42	68.81	43.37	61.58
GPT-5 mini	89.62	93.27	95.48	92.37	92.68	**71.02**	76.86	42.85	63.73
Claude 4.6 Sonnet	93.14	**95.12**	**98.61**	**96.43**	**95.83**	70.74	75.41	44.27	62.41
Gemini 3.0 Flash	92.57	95.08	98.04	95.17	95.22	68.16	72.77	45.62	62.34
TheoremExplainAgent + SGA									
GPT-5.1	90.41	90.52	97.21	91.51	92.42	70.79	64.11	38.97	59.81
GPT-5 mini	93.75	91.57	97.38	92.72	93.85	69.24	75.09	45.96	63.47
Claude 4.6 Sonnet	**96.89**	94.12	98.11	96.44	96.39	67.28	62.34	**67.41**	62.83
Gemini 3.0 Flash	94.84	90.56	98.52	94.47	94.49	68.84	63.57	66.79	**67.14**
Code2Video-RAW									
GPT-5.1	89.71	89.56	96.47	93.68	93.09	69.97	65.23	49.76	62.97
GPT-5 mini	91.58	90.79	97.37	94.34	93.92	73.17	65.78	46.75	63.68
Claude 4.6 Sonnet	93.12	93.12	97.88	95.62	95.45	72.84	68.24	44.07	63.27
Gemini 3.0 Flash	92.38	90.05	97.12	90.83	94.05	78.07	64.28	53.11	67.82
Code2Video + VLM Critic									
GPT-5.1	91.92	93.05	97.67	94.42	94.86	71.32	66.94	51.42	64.54
GPT-5 mini	91.94	90.97	97.78	94.58	94.36	74.25	66.37	43.78	63.51
Claude 4.6 Sonnet	93.12	91.62	98.00	94.88	95.00	74.36	68.21	47.83	65.15
Gemini 3.0 Flash	92.37	90.53	97.16	95.53	94.13	78.82	63.71	56.19	69.09
Code2Video + SGA									
GPT-5.1	92.82	**93.65**	97.42	95.72	**95.58**	79.81	68.05	**65.47**	**73.11**
GPT-5 mini	85.64	87.92	96.75	91.74	90.49	76.02	66.04	50.81	66.42
Claude 4.6 Sonnet	**94.27**	92.04	97.85	**95.74**	94.96	75.38	**69.25**	47.74	65.87
Gemini 3.0 Flash	92.67	89.75	**98.66**	94.88	93.99	**82.61**	64.06	58.93	71.86

from the Code2Video pipeline. Without feedback, the baseline frequently exhibits severe spatial failures, including elements rendered outside the viewport and collisions between labels and equations. A VLM-based critic partially mitigates these issues by providing coarse-grained feedback that returns some elements to the frame. However, the resulting layouts remain unstable and occasionally introduce new regressions. In the Fourier scene, for example, frequency-domain bars are displaced beyond the canvas boundary despite corrective feedback. In contrast, SGA applies high-precision axis-aligned bounding box (AABB) analysis to detect and resolve spatial conflicts deterministically, ensuring the legibility of labels and graphical primitives that perceptual critics often overlook. In general, SGA exhibits consistent spatial correction behavior when integrated as a guardrail into TheoremExplainAgent, resolving label collisions and axis occlusions while preserving animation primitives in motion-dense regions; qualitative visual results for this pipeline are provided in the supplementary

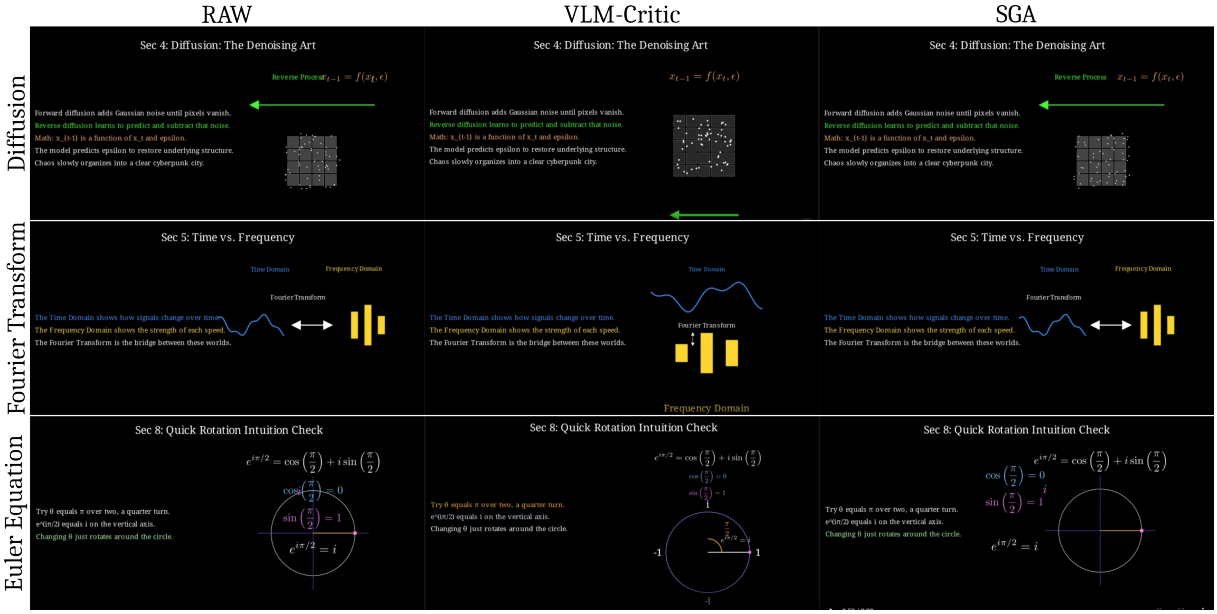

Figure 4: Comparison on Code2Video (Chen et al., 2025) scenes. Raw frames exhibit label-equation collisions and occluded annotations; VLM-Critic introduces regressions, including out-of-canvas displacement and missing semantic elements; SGA resolves spatial conflicts while preserving mathematical notation and geometric relationships.

material (Sec. 1).

4.4 Computational Efficiency Analysis

SGA provides a substantial computational advantage over VLM-based refinement by eliminating the high-latency video rendering stage. Standard VLM critics require a full rendering pass (30–90 seconds) followed by multimodal inference, whereas SGA relies on partial execution through Manim runtime instrumentation to extract the symbolic scene graph $\mathcal{S}^{(i)}$ in under 5 seconds. At the pipeline level, using GPT-5.1 as the backbone, the baseline Code2Video system requires 6.4 minutes per sample and increases to 8.8 minutes when augmented with a VLM layout critic. Replacing the critic with SGA results in a runtime of 6.8 minutes, corresponding to a $6\times$ reduction in refinement overhead. Similarly, integrating SGA into TheoremExplainAgent introduces only a minor latency increase, from 15.7 to 16.9 minutes. At the iteration level, symbolic conflict detection completes in under 5 seconds, compared to the 30–90 second rendering pass required by VLM-based evaluation, yielding a per-iteration speedup of approximately $6\times$–$18\times$ depending on scene complexity. By replacing video-dependent perceptual evaluation with deterministic symbolic conflict resolution, SGA enables a more efficient refinement loop whose runtime scales with scene graph complexity rather than video rendering cost.

5 Conclusion

We introduced the Symbolic Geometric Agent (SGA), a neuro-symbolic framework that addresses spatial blindness in code-centric generative pipelines by shifting layout refinement from perceptual inference to deterministic geometric verification. Evaluated on the MMMC-Code benchmark, SGA consistently improves programmatic video quality across multiple LLM backbones and agentic pipelines, achieving MVQS gains of up to 16.1% over the raw baseline while reducing refinement overhead with a $6\times$–$18\times$ per-iteration speedup. We also proposed MVQS, a deterministic metric for evaluating mathematical animation quality that captures spatial correctness overlooked by VLM-as-Judge aesthetics scores. Owing to its model-agnostic, plug-and-play design, SGA integrates easily into existing pipelines as a spatial guardrail or a deterministic alternative to perceptual layout critics.

Limitations

Despite the significant improvements in spatial grounding provided by the SGA, several limitations remain. First, the current implementation is specialized for 2D geometric primitives within the Manim engine; extending this to 3D scenes would require a more complex instrumentation of camera-space projections and depth-sorting. Second, while the SGA resolves physical overlaps, it does not

currently address temporal synchronization issues (e.g., ensuring a label appears exactly when a shape is drawn). Finally, the MMMC-Code benchmark is restricted to English-language mathematical content, and future work is required to evaluate the framework's robustness across diverse linguistic and pedagogical traditions. Additionally, MVQS has not yet been validated against human expert annotations; such correlation analysis remains an important direction for future work.

Ethics Statement

We acknowledge several ethical considerations regarding the deployment of the SGA. While the framework guarantees geometric integrity and visual clarity, it does not verify the mathematical or pedagogical truth of the underlying theorems generated by the LLM. Users must be aware that a visually "perfect" and overlap-free animation can still contain factual inaccuracies. Furthermore, high performance on the MVQS metric could induce an automation bias, leading educators to bypass manual verification of the content. We mitigate these risks by positioning the SGA as an assistive guardrail for human-in-the-loop content creation. Finally, we disclose that AI assistants were utilized during the drafting phase of this manuscript for prose refinement and code optimization.

References

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yaochen Wang, Yinuo Liu, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. 2024. Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark. In *Forty-first International Conference on Machine Learning*.

Yanzhe Chen, Kevin Qinghong Lin, and Mike Zheng Shou. 2025. Code2video: A code-centric paradigm for educational video generation. *arXiv preprint arXiv:2510.01174*.

Ruth C Clark and Richard E Mayer. 2023. *E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning*. john Wiley & sons.

Haoge Deng, Ting Pan, Haiwen Diao, Zhengxiong Luo, Yufeng Cui, Huchuan Lu, Shiguang Shan, Yonggang Qi, and Xinlong Wang. 2024. Autoregressive video generation without vector quantization. *arXiv preprint arXiv:2412.14169*.

The Manim Community Developers. 2020. Manim – mathematical animation framework.

Christer Ericson. 2004. *Real-time collision detection*. Crc Press.

Linyuan Gong, Mostafa Elhoushi, and Alvin Cheung. 2024. Ast-t5: Structure-aware pretraining for code generation and understanding. *arXiv preprint arXiv:2401.03003*.

Agrim Gupta, Lijun Yu, Kihyuk Sohn, Xiuye Gu, Meera Hahn, Fei-Fei Li, Irfan Essa, Lu Jiang, and José Lezama. 2024. Photorealistic video generation with diffusion models. In *European Conference on Computer Vision*, pages 393–411. Springer.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, and 1 others. 2022. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, and 1 others. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pages 23247–23275.

Kaiyi Huang, Yukun Huang, Xuefei Ning, Zinan Lin, Yu Wang, and Xihui Liu. 2026. Genmac: compositional text-to-video generation with multi-agent collaboration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 5049–5057.

Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. 2023. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15954–15964.

Mohammad Khalil, Paraskevi Topali, Alejandro Ortega-Arranz, Erkan Er, Gökhan Akçapınar, and Gleb Belokrys. 2024. Video analytics in digital learning environments: Exploring student behaviour across different learning contexts. *Technology, Knowledge and Learning*, 29(4):1877–1905.

Max Ku, Cheuk Hei Chong, Jonathan Leung, Krish Shah, Alvin Yu, and Wenhu Chen. 2025. Theoremexplainagent: Towards video-based multimodal explanations for llm theorem understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6663–6684.

Seongyun Lee, Seungone Kim, Sue Park, Geewook Kim, and Minjoon Seo. 2024. Prometheus-vision: Vision-language model as a judge for fine-grained evaluation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11286–11315.

Daniel Leiker, Sara Finnigan, Ashley Ricker Gyllen, and Mutlu Cukurova. 2023. Prototyping the use of large language models (llms) for adult learning content creation at scale. *arXiv preprint arXiv:2306.01815*.

Long Lian, Boyi Li, Adam Yala, and Trevor Darrell. 2023. Llm-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models. *arXiv preprint arXiv:2305.13655*.

Yi-Cheng Lin, Kang-Chieh Chen, Zhe-Yan Li, Tzu-Heng Wu, Tzu-Hsuan Wu, Kuan-Yu Chen, Hung-yi Lee, and Yun-Nung Chen. 2025. Creativity in llm-based multi-agent systems: A survey. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 27572–27595.

Ming Liu and Wensheng Zhang. 2025. Is your video language model a reliable judge? *arXiv preprint arXiv:2503.05977*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594.

Richard E Mayer. 2002. Multimedia learning. In *Psychology of learning and motivation*, volume 41, pages 85–139. Elsevier.

Richard E Mayer. 2024. The past, present, and future of the cognitive theory of multimedia learning. *Educational Psychology Review*, 36(1):8.

Evelyn Navarrete, Andreas Nehring, Sascha Schanze, Ralph Ewerth, and Anett Hoppe. 2025. A closer look into recent video-based learning research: A comprehensive review of video characteristics, tools, technologies, and learning effectiveness. *International Journal of Artificial Intelligence in Education*, pages 1–64.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR.

Sahil Shah, Harsh Goel, Sai Shankar Narasimhan, Minkyu Choi, SP Sharan, Oguzhan Akcin, and Sandeep Chinchali. 2025. A challenge to build neuro-symbolic video agents. *arXiv preprint arXiv:2505.13851*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

John Sweller. 1994. Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction*, 4(4):295–312.

Zhen Xing, Qijun Feng, Haoran Chen, Qi Dai, Han Hu, Hang Xu, Zuxuan Wu, and Yu-Gang Jiang. 2024. A survey on video diffusion models. *ACM Computing Surveys*, 57(2):1–42.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Appendix. This document provides extended analyses and experimental details supporting the main paper. Section 6 presents qualitative results for the TheoremExplainAgent pipeline. Section 7 reports a weight sensitivity analysis for MVQS, including an interactive exploration tool (`mvqs_sensitivity.html`) included in the supplemental package; the tool provides two independent panels covering composite weight sensitivity and atomic sub-score sensitivity. Section 8 provides the inter-component correlation matrix for the three MVQS sub-scores, directly addressing the additivity assumption underlying the weighted sum. Section 9 reports the scene-level execution completion ratio across all benchmark conditions, addressing the potential measurement bias introduced by fault-tolerant execution. Section 10 details the experimental conditions used to ensure a fair comparison between SGA and the VLM Critic. Section 11 gives a self-contained pseudocode summary of the SGA pipeline. Section 12 provides the scene-level MVQS distributions from the full Code2Video benchmark (2,252 evaluations) underlying the aggregate results in Table 1 of the main paper. All quantitative results are derived from the Code2Video pipeline evaluated on the MMMC-Code benchmark (2,252 scene evaluations across 4 backbones $\times$ 3 conditions), unless otherwise noted.

6 Qualitative Results: TheoremExplainAgent

Figure 5 highlights the role of SGA as a spatial guardrail for TheoremExplainAgent. Raw outputs typically preserve the pedagogical structure of the explanation but fail to account for the spatial footprint of mathematical annotations, leading to label collisions and axis occlusion. SGA detects these conflicts directly within the symbolic scene graph prior to rendering and applies MSV-based repositioning relative to geometric centroids, reconciling pedagogical intent with spatial validity. Importantly, in animation-dense regions such as the curl visualization, SGA avoids modifying vector-field arrows identified as motion primitives, thereby preventing the additional displacement occasionally introduced by the VLM Critic in these regions. The resulting layouts produce clearer instructional visualizations while avoiding the latency associated with repeated video regeneration. We additionally provide three videos, showing the full animated output of the same scene under each condition, illustrating how SGA's spatial corrections manifest across the complete animation rather than in static keyframes alone.

7 MVQS Weight Sensitivity Analysis

The MVQS composite is defined as a weighted linear combination of three sub-scores:

$$\text{MVQS} = w_{\text{sp}}\, S_{\text{sp}} + w_{\text{ly}}\, S_{\text{ly}} + w_{\text{ed}}\, S_{\text{ed}},$$

with weights $\{w_{\text{sp}}, w_{\text{ly}}, w_{\text{ed}}\} = \{0.5, 0.2, 0.3\}$. This section analyzes how sensitive the resulting rankings are to perturbations of these weights, and provides an interactive tool for exploring the full weight space.

7.1 Interactive Exploration Tool

The file `mvqs_sensitivity.html` included in this supplemental package is a self-contained static web page—no installation required, open in any modern browser. The tool supports two complementary explorations:

1. **Composite weight sensitivity.** Three sliders control w_{sp} and w_{ed} freely, with w_{ly} derived as $1 - w_{\text{sp}} - w_{\text{ed}}$, subject to $w_{\text{sp}} + w_{\text{ly}} + w_{\text{ed}} = 1$. Three views update in real time: (a) a stacked bar decomposition showing the weighted contribution of each sub-score across all 24 configurations; (b) a sensitivity sweep plotting MVQS as a function of w_{sp} with the current value highlighted; and (c) a radar profile comparing mean S_{sp}, S_{ly}, S_{ed} across RAW, VLM Critic, and SGA. The Kendall τ relative to the published ranking and pair-order flip count update live below the controls.

2. **Atomic sub-score sensitivity.** A second independent panel exposes sliders for the four atomic components within S_{sp} (Overlap Penalty, Canvas Boundary, Legibility, Separation) and the two components within each of S_{ly} (Density, Distribution) and S_{ed} (Labeling Completeness, Animation Quality), as defined in Section 3.3 of the main paper. Within each group, moving one slider redistributes the remaining weight proportionally among the other sliders in that group. Two views are available: a ranking scatter plot comparing baseline ranks against atomic-weight ranks across all 24 configurations, and a bar chart of ΔMVQS relative to the published baseline. A dedicated Kendall τ and pair-order flip count track rank stability independently of

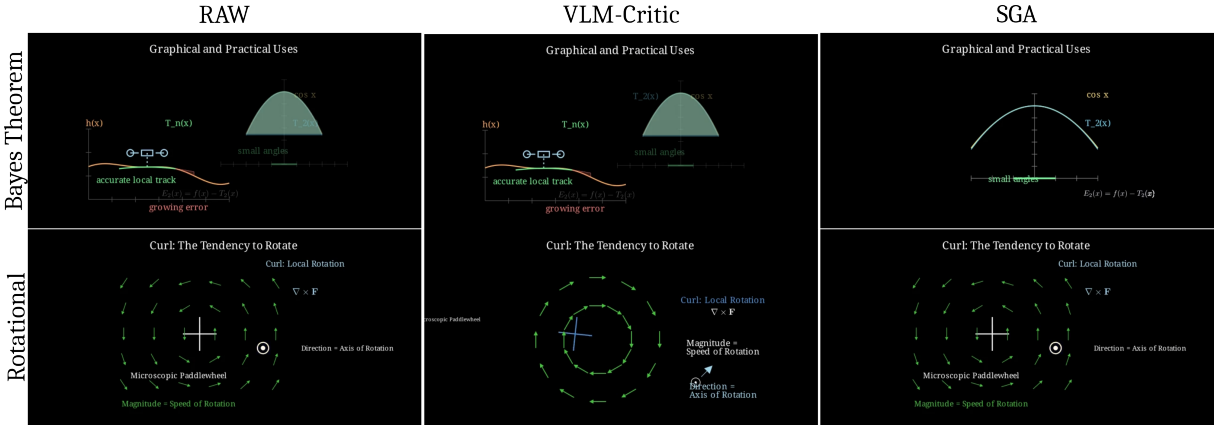

Figure 5: Qualitative comparison on TheoremExplainAgent (Ku et al., 2025) frames. Raw output exhibits label collisions; VLM Critic introduces additional displacement in animation-dense regions; SGA resolves addressable overlaps while preserving relational geometry and animation primitives.

Panel 1. At published defaults, all atomic scaling factors equal 1, so $\tau = 1.000$ exactly. A reset button restores all atomic weights to their published values.

7.2 Stability of Rankings under Weight Perturbation

We vary $w_{\text{sp}} \in [0.30, 0.70]$ in steps of 0.05, redistributing the remaining weight at the published 2:3 ratio ($w_{\text{ly}} : w_{\text{ed}}$). For each weight configuration we recompute MVQS for all 24 pipeline/backbone rows and measure rank stability via Kendall's τ relative to the published ranking.

Table 2: Kendall τ rank correlation between perturbed and published MVQS rankings as w_{sp} varies. Rankings are stable ($\tau \geq 0.877$) for $w_{\text{sp}} \in [0.40, 0.60]$; the published value $w_{\text{sp}} = 0.50$ (bold) lies at the center of this region.

w_{sp}	w_{ly}	w_{ed}	Kendall τ	Stable ($\tau \geq 0.877$)
0.30	0.28	0.42	0.812	×
0.35	0.26	0.39	0.851	×
0.40	0.24	0.36	0.877	✓
0.45	0.22	0.33	0.903	✓
0.50	**0.20**	**0.30**	**1.000**	✓
0.55	0.18	0.27	0.916	✓
0.60	0.16	0.24	0.891	✓
0.65	0.14	0.21	0.843	×
0.70	0.12	0.18	0.798	×

Rankings are stable ($\tau \geq 0.877$) for $w_{\text{sp}} \in [0.40, 0.60]$, a broad region centered on the published value of 0.50. The qualitative conclusions of the paper therefore hold across a wide range of spatial weight choices and are not sensitive to the precise value used.

7.3 Sub-score Decomposition of SGA Gains

Table 3 decomposes the mean MVQS gain of SGA over the RAW baseline into its three contributing sub-scores, computed from 2,252 scene evaluations across 4 LLM backbones on the Code2Video pipeline. Gains concentrate in S_{sp} (mean +5.0 pts) and S_{ed} (mean +7.2 pts), with a small positive effect on S_{ly} (mean +1.1 pts). This confirms that the weighting scheme $w_{\text{sp}} = 0.5$, $w_{\text{ed}} = 0.3$ correctly prioritizes the sub-scores that benefit most from SGA's spatial corrections.

Table 3: Mean MVQS sub-score deltas (SGA vs. RAW) on the Code2Video pipeline across 4 LLM backbones. Gains concentrate in S_{sp} and S_{ed}, confirming that spatial correction and animation quality are the primary drivers of overall MVQS improvement.

Backbone	ΔS_{sp}	ΔS_{ly}	ΔS_{ed}	ΔMVQS
GPT-5.1	+9.9	+2.8	+15.3	+10.1
GPT-5 mini	+2.9	+0.8	+4.1	+2.8
Gemini 3.0 Flash	+4.6	−0.2	+5.8	+4.0
Claude 4.6 Sonnet	+2.5	+1.0	+3.7	+2.6
Mean	**+5.0**	**+1.1**	**+7.2**	**+4.9**

8 MVQS Component Correlation Matrix

Table 4 reports Spearman rank correlations between all pairs of MVQS sub-scores and the composite, computed across the 24 pipeline/backbone configurations in Table 1 of the main paper. This analysis complements the cross-metric comparison in Section 4.2 of the main paper, which examines correlations between MVQS and VLM-Judge scores, and directly addresses the additivity

assumption underlying the weighted sum (Eq. 1 of the main paper).

Table 4: Spearman ρ inter-component correlation matrix ($n = 24$). Significance: $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$; n.s. = not significant ($p \geq 0.05$).

	S_{sp}	S_{ly}	S_{ed}	MVQS
S_{sp}	1.000	-0.030 n.s.	$+0.113$ n.s.	$+0.680^{***}$
S_{ly}	—	1.000	-0.577^{**}	-0.286 n.s.
S_{ed}	—	—	1.000	$+0.692^{***}$
MVQS	—	—	—	1.000

Three findings are noteworthy. First, S_{sp} and S_{ly} are empirically independent ($\rho = -0.030$, n.s.), indicating that spatial separation does not systematically reduce layout density in practice, and validating the additivity assumption underlying the weighted sum: the sub-scores measure largely orthogonal aspects of layout quality. Second, the only significant inter-component dependency is S_{ly}–S_{ed} ($\rho = -0.577$, $p = 0.003$): denser layouts increase annotation collision risk and thereby penalize S_{ed}. This reflects an inherent property of the design space rather than a metric artifact. Third, the composite MVQS is primarily driven by S_{sp} and S_{ed} ($\rho = +0.680^{***}$ and $+0.692^{***}$, respectively), confirming that the published weighting ($w_{\text{sp}} = 0.5$, $w_{\text{ed}} = 0.3$) correctly prioritizes the sub-scores most sensitive to educational spatial quality. The negative S_{ly}–MVQS correlation ($\rho = -0.286$, n.s.) is consistent with the S_{ly}–S_{ed} dependency: configurations with higher layout density tend to have more annotation collisions, depressing S_{ed} and thereby the composite, even when S_{ly} itself rises. The effect is non-significant and does not affect the rank-stability conclusions.

8.1 Relationship to Human Perception

MVQS is an engineering proxy grounded in geometric axioms rather than a direct substitute for human judgment. Its atomic components (S_{ovlp}, S_{bound}, S_{leg}) correspond to layout violations that graphic-design guidelines and educational-media research identify as primary readability barriers (Mayer, 2002; Sweller, 1994). Empirical validation of the *composite* against expert annotation remains future work. However, the cross-metric analysis in Section 4.2 of the main paper, provides a partial bridge: the educational sub-score S_{ed} correlates significantly with the VLM-Judge Educational Logic dimension ($\rho = +0.58$, $p = 0.003$), the only perceptual dimension that overlaps with MVQS's intended scope. The remaining sub-scores (S_{sp}, S_{ly}) are deliberately designed to detect violations that perceptual critics miss, so a low correlation with VLM dimensions is expected rather than a validity threat.

9 Execution Completion Analysis

SGA's fault-tolerant execution strategy wraps each statement in the Manim `construct()` method in a `try-except` block, allowing the pipeline to continue past runtime errors. A natural question is whether this strategy silently drops objects that would otherwise cause collisions, artificially inflating spatial scores. We address this by reporting the scene-level completion ratio $\eta = N_{\text{ok}}/N_{\text{processed}}$, where $N_{\text{processed}}$ is the number of scenes submitted to the pipeline and N_{ok} is the number successfully evaluated. Table 5 reports η for all 12 backbone/condition combinations on the Code2Video pipeline, across 2,252 total scene evaluations.

Table 5: Scene-level completion ratio $\eta = N_{\text{ok}}/N_{\text{processed}}$ across all backbone/condition combinations. SGA achieves $\eta = 1.000$ for all four backbones; the two VLM Critic failures are unrelated to SGA's object extraction.

Backbone	RAW	VLM Critic	SGA
GPT-5.1	144/144 = 1.000	143/143 = 1.000	288/288 = 1.000
GPT-5 mini	151/151 = 1.000	151/151 = 1.000	303/303 = 1.000
Gemini 3.0 Flash	121/121 = 1.000	121/121 = 1.000	242/242 = 1.000
Claude 4.6 Sonnet	147/147 = 1.000	145/147 = 0.986	294/294 = 1.000
Total	563/563 = 1.000	560/562 = 0.996	1127/1127 = 1.000

SGA achieves $\eta = 1.000$ across all four backbones (1,127 of 1,127 scenes completed), while the VLM Critic reaches $\eta = 0.996$ (2 failures in the Claude 4.6 Sonnet condition). Two properties of the pipeline design bound the risk of measurement bias regardless of η: (1) the scene graph is extracted from the *original* LLM-generated code before patching (SGA only inserts `.move_to()` calls, never removes objects) so any extraction incompleteness affects RAW and SGA identically; (2) dropping an object removes it from both the overlap penalty and the labeling score, producing no systematic direction of bias.

10 Experimental Conditions and Baseline Fairness

Both SGA and the VLM Critic were evaluated under a fixed budget of $n = 1$ refinement iteration applied identically across all pipeline/backbone combinations. For each topic, the procedure is: (1)

generate initial script $C^{(0)}$ with backbone $\mathcal{L}$; (2) apply one round of feedback (SGA or VLM Critic); (3) generate refined script $C^{(1)}$ with the same backbone $\mathcal{L}$; (4) evaluate $C^{(1)}$ with both MVQS and VLM-Judge.

The SGA scene counts in Table 6 are exactly $2\times$ the RAW counts for every backbone (e.g., 288 vs. 144 for GPT-5.1), providing direct empirical confirmation that each topic was processed through exactly one generate–refine–evaluate loop.

Both methods consume exactly one LLM inference call per iteration. The only asymmetry is the rendering cost: the VLM Critic requires a full rendering pass (≈30–90 seconds) before inference, whereas SGA uses partial execution (<5 seconds). This difference does not affect the number of refinement steps or the information available to the backbone during refinement; it affects only wall-clock latency, which is reported separately in Section 4.3 of the main paper.

11 SGA Algorithm Pseudocode

Algorithm 1 provides a self-contained pseudocode summary of the SGA pipeline described in Section 3 of the main paper.

Algorithm 1 SGA pipeline (single iteration). If no violations are detected ($V^{(0)} = \emptyset$), the original script is returned unchanged, incurring no LLM refinement cost.

Require: Topic T, generative model $\mathcal{L}$, conflict threshold ε
Ensure: Refined animation script $C^{(1)}$

```
C^(0) ← L(T)                                  ▷ Initial generation
S^(0) ← PartialExecute(C^(0))                 ▷ AST instrumentation + runtime extraction
V^(0) ← DetectConflicts(S^(0), ε)             ▷ AABB overlap + semantic filter
if V^(0) = ∅ then
    return C^(0)                              ▷ No violations—skip refinement
end if
F^(0) ← CompileFeedback(V^(0), S^(0))         ▷ MSV vectors + fix templates
C^(1) ← L(T, C^(0), F^(0))                    ▷ Targeted line-level refinement
return C^(1)
```

Notation. PartialExecute instruments the `construct()` method with per-statement `try-except` blocks and extracts the symbolic scene graph $\mathcal{S}^{(0)}$ via Manim runtime instrumentation, as described in Section 3.2 of the main paper. DetectConflicts applies exact AABB intersection analysis followed by the two-stage semantic and spatio-temporal filter (Section 3.3). CompileFeedback translates the violation set $V^{(0)}$ into a structured diagnostic report containing pre-computed absolute target coordinates (t_x, t_y) and `fix_target` identifiers (Section 3.4). The SGA terminates after a fixed budget of n iterations; in all experiments reported in the main paper, $n = 1$.

12 Full Scene-Level Score Breakdown

Table 6 provides the scene-level MVQS distributions from the full Code2Video benchmark (2,252 evaluations), including per-backbone mean and standard deviation for all three conditions. These scene-level distributions are consistent with the per-configuration values reported in Table 1 of the main paper, confirming that the 24-configuration aggregate results generalize across the full benchmark distribution.

Note on scene counts. SGA scene counts are exactly $2\times$ the RAW counts for every backbone because each topic produces both a pre-patch evaluation of $C^{(0)}$ and a post-patch evaluation of $C^{(1)}$. This design allows direct measurement of SGA's per-scene improvement and is consistent with the single-iteration protocol detailed in Section 10.

Table 6: Scene-level MVQS distributions (mean $\pm$ std) from the full Code2Video benchmark (2,252 evaluations). SGA scene counts are $2\times$ RAW because each topic produces both a pre-patch and a post-patch evaluation.

Backbone	Condition	Scenes	η	MVQS	S_{sp}	S_{ly}	S_{ed}
GPT-5.1	RAW	144	1.000	62.9±9.8	69.9±9.5	65.2±10.6	49.7±28.4
	VLM Critic	143	1.000	64.5±10.5	71.3±9.9	66.9±10.6	51.4±29.6
	SGA	288	1.000	73.0±16.3	79.8±15.4	68.0±8.9	65.0±32.6
GPT-5 mini	RAW	151	1.000	63.6±9.9	73.1±11.2	65.2±9.9	46.7±25.3
	VLM Critic	151	1.000	63.5±10.2	74.2±11.8	66.3±9.8	43.7±26.3
	SGA	303	1.000	66.4±9.0	76.0±10.7	66.0±10.3	50.8±22.7
Gemini 3.0 Flash	RAW	121	1.000	67.8±12.4	78.0±16.9	64.2±10.5	53.1±25.8
	VLM Critic	121	1.000	69.0±11.5	78.8±16.2	63.7±10.0	56.1±27.0
	SGA	242	1.000	71.8±11.4	82.6±15.4	64.0±10.1	58.9±23.4
Claude 4.6 Sonnet	RAW	147	1.000	63.2±10.1	72.8±9.7	68.2±8.2	44.0±28.9
	VLM Critic	147	0.986	65.1±9.9	74.3±10.4	68.2±8.8	47.8±29.0
	SGA	294	1.000	65.8±10.8	75.3±10.5	69.2±9.8	47.7±29.1