Code2Video:A Code-centric Paradigm for Educational Video Generation

Yanzhe Chen* Kevin Qinghong Lin* Mike Zheng Shou †

Show Lab, National University of Singapore

https://showlab.github.io/Code2Video/

Abstract

While recent generative models advance pixel-space video synthesis, they remain limited in producing professional educational videos, which demand disciplinary knowledge, precise visual structures, and coherent transitions, limiting their applicability in educational scenarios. Intuitively, such requirements are better addressed through the manipulation of a renderable environment, which can be explicitly controlled via logical commands (*e.g.,* code). In this work, we propose **Code2Video**, a code-centric agent framework for generating educational videos via executable Python code. The framework comprises three collaborative agents: *(i) Planner*, which structures lecture content into temporally coherent flows and prepares corresponding visual assets; *(ii) Coder*, which converts structured instructions into executable Python codes while incorporating scope-guided auto-fix to enhance efficiency; and *(iii) Critic*, which leverages vision-language models (VLM) with visual anchor prompts to refine spatial layout and ensure clarity. To support systematic evaluation, we build **MMMC**, a benchmark of professionally produced, discipline-specific educational videos. We evaluate MMMC across diverse dimensions, including VLM-as-a-Judge aesthetic scores, code efficiency, and particularly, **TeachQuiz**, a novel end-to-end metric that quantifies how well a VLM, after unlearning, can recover knowledge by watching the generated videos. Our results demonstrate the potential of Code2Video as a scalable, interpretable, and controllable approach, achieving 40% improvement over direct code generation and producing videos comparable to human-crafted tutorials. The code and datasets are available at https://github.com/showlab/Code2Video.

1 Introduction

"If you want to master something, teach it." — Richard Feynman

Recent advances in natural video generation have made remarkable progress in *pixel* space. End-to-end solutions, including diffusion-based [14, 49] and autoregressive architectures [48, 61], can synthesize visually compelling videos directly from text prompts (*i.e.,***Text2Video**), achieving fine appearance and short-form fidelity. Yet these models struggle when the task requires long-form reasoning or multi-entity interaction [23]. To overcome these limitations, recent works have moved toward multi-agent pipelines, where complex video generation is decomposed into collaborative subtasks, allowing iterative refinement, temporal structuring [62, 17, 53].

Educational videos that aim to teach subject-specific knowledge face unique challenges in the reasoning era. Unlike short-form entertainment, educational content must integrate deep domain expertise [6], carefully designed animations or transitions, and step-by-step reasoning [3] to support

* Equal contribution † Corresponding author: mike.zheng.shou@gmail.com

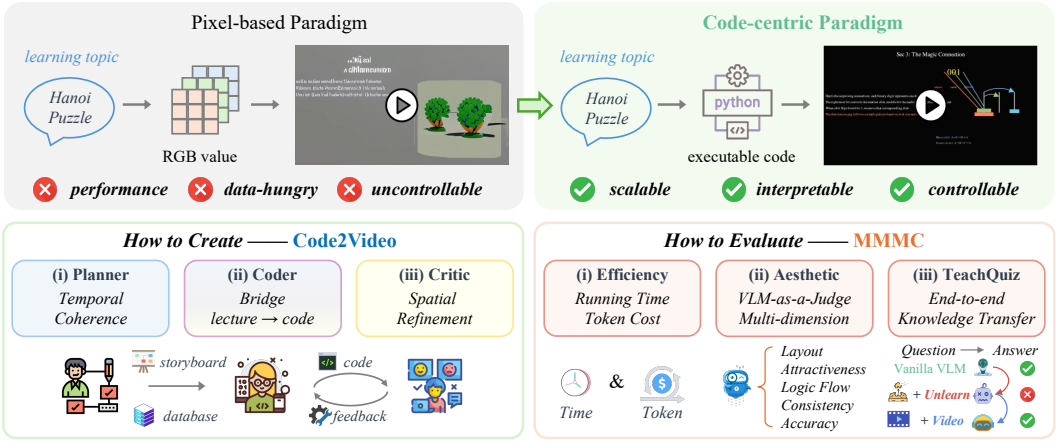


Figure 1: Overview of **Code2Video**. A code-centric paradigm for educational video generation, where Planner ensures temporal flow, Coder bridges instructions to executable animations, and Critic refines spatial layout. Evaluation is performed on **MMMC** with multi-dimensional metrics.

actual skill acquisition. This raises two fundamental challenges: **(i)** How to create high-quality educational videos that maintain both temporal coherence—concepts introduced, expanded, and reinforced in logical sequence—and spatial clarity—elements arranged legibly without occlusion; and **(ii)** How to evaluate educational videos beyond appearance, ensuring that they are educationally effective and semantically aligned with the intended learning topic. Existing video generation pipelines rarely satisfy these requirements, leaving a critical gap for agentic methods that unify temporal planning, spatial organization, and educational assessment.

We are motivated by the intuition that code provides a uniquely suitable substrate for educational video generation. Unlike black-box models, code-centric pipelines are *scalable*, since new visualizations and external assets can be modularly integrated; *interpretable*, as every sequence, layout, and rendering decision is explicitly scripted and thus auditable; and *controllable*, enabling precise temporal sequencing and spatial organization through programmatic specification.

Building on these insights, we propose **Code2Video**, an agentic, code-centric framework for generating high-quality educational videos. The system decomposes the task into three agents: the *Planner* sequences concepts, examples, and recaps into a coherent lecture flow; the *Coder* translates structured instructions into executable Manim code, yielding precise, editable visualizations with consistent layout and timing; and the *Critic* leverages multimodal feedback and visual anchor prompts to refine spatial organization and ensure alignment with learning objectives. This tri-agent design explicitly models the temporal and spatial structure of instruction, while grounding the entire pipeline in transparent, reproducible, and extensible code.

To evaluate this paradigm, we propose **MMMC**, a benchmark reflecting the distinct goal of educational videos: teaching new knowledge. It comprises professionally produced, discipline-specific Manim tutorials across 13 domains (*e.g.,*topology, physics). Evaluation covers three complementary dimensions: (i) VLM-as-a-Judge aesthetic and structural quality; (ii) code efficiency, measuring generation time and token consumption; and (iii) **TeachQuiz**, a novel end-to-end knowledge-transfer metric that enforces unlearning of the target concept in a VLM, and then measures how effectively the generated video restores it. This multi-dimensional protocol directly probes educational efficacy and grounds a code-centric paradigm for video generation. Our results reveal clear trends: pixel-based models struggle with fine details and coherence, while direct code-centric generation improves TeachQuiz by 30%. Our full Planner–Coder–Critic pipeline further delivers a stable 40% gain. In human studies on TeachQuiz scores, agentically generated videos even outperform professional human-made tutorials, underscoring *the effectiveness of code-centric, agentic generation*.

Our contributions are summarized as follows:

- **A New Paradigm for Video Generation.** We introduce a new code-centric paradigm for educational video generation, positioning executable code as the unifying medium for temporal sequencing and spatial organization.

- **Effective Designs for Visual Animation Agent.** We highlight a modular agent design with three key components: (i) Planner expands an external database for reference, enabling parallel yet consistent storyboard; (ii) Coder ensures compilable code via automatic debugging and scope-guided repair; (iii) Critic refines spatial layout and clarity using visual anchor prompts.
- **New Benchmark with Well-designed Evaluation.** We present MMMC, the first benchmark for code-centric educational video generation with multi-dimensional evaluation of efficiency, aesthetics, and end-to-end knowledge transfer.

2 Related Work

2.1 Video Generation

Early text-to-video generation methods ***(i)*** primarily develop **end-to-end** visual diffusion models into the temporal domain, employing architectures such as space–time UNets and latent 3D VAEs [49, 15, 41, 14]. Representative works have demonstrated impressive perceptual fidelity and longer durations [4, 57, 23, 55], yet their reliance on *pixel*-space synthesis poses significant challenges for educational video creation, where precise layout and symbolic alignment are essential for effective pedagogy. Parallel research has explored autoregressive and progressive schemes [27, 10, 45, 52] to enable long-form video synthesis [13, 31, 67], but these approaches continue to inherit the limitations of pixel-level control, making it difficult to achieve board-like composition and stepwise exposition required in educational contexts [18, 23, 30]. ***(ii)*** To address the complexity of structured content generation, recent studies leverage **multi-agent collaboration**, particularly within LLM applications, to decompose tasks, coordinate tool usage, and facilitate iterative self-improvement [62, 17, 16, 53, 40]. While multi-agent frameworks have proven effective in domains such as question answering and web interaction, their application to video generation remains limited and largely unexplored [22, 24, 51]. ***(iii)*** Built on top of the multi-agent, our approach introduces a ***code*–centric animation** for educational video synthesis. By elevating executable code as the generative substrate, our method enables symbolic layout, temporal structure, and deterministic reproducibility.

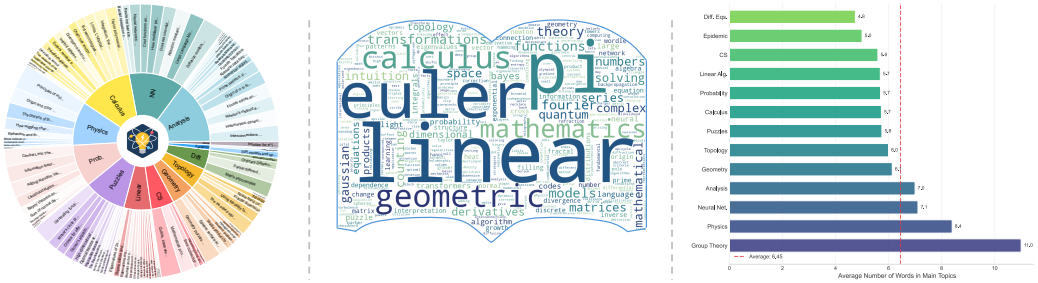
2.2 Coding Agents

Recent advances in LLM-based tool-use have enabled agents to autonomously call APIs, retrieve specifications, and verify outputs, laying the groundwork for neuro-symbolic modularity and robust task decomposition [2, 21, 58, 39, 44]. Representative methods demonstrate that integrating code execution and tool invocation can reliably extend language models beyond **text-only** reasoning, supporting complex workflows and project-level code generation [36, 29, 12, 7, 66]. These developments highlight the potential of LLM agents to coordinate external retrieval, maintain consistent memory across parallel processes, and incorporate feedback loops for iterative refinement [26, 25, 56, 64]. In parallel, research at the intersection of coding and visual reasoning has shown that generating and executing programs can yield structured perception and controllable rendering [35, 11, 42, 33, 28, 8]. **Visual programming** and visual-to-code approaches leverage program synthesis for compositional reasoning and spatial arrangement, as evidenced by benchmarks evaluating the translation of images or text into executable code for charts, plots, or graphical environments [50, 65, 20, 46, 60]. These works demonstrate the capacity of language models to bridge symbolic and visual domains, though they primarily focus on *static* figures or localized visual tasks [54, 38, 47, 59, 19]. Our work pioneers the integration of code generation and visual synthesis for *dynamic* educational **video creation**. By employing agents to generate executable Manim code, we achieve precise, temporally structured control over both content and layout throughout entire lectures. This approach represents an early effort to bridge coding and video generation, enabling the production of educational material that is both semantically rigorous and visually coherent.

3 MMMC Benchmark

3.1 Task Formulation

The task of code-centric educational video generation maps a learning query to executable *Manim* [32] code whose rendering yields a tutorial video. The challenge lies in multi-step reasoning, precise temporal sequencing, and spatial coherence, where even minor syntax errors can nullify execution.



(1) Categories and Distribution **(2) Learning Word Cloud** **(3) Learning Topic Length**

Figure 2: **MMMC overview**. (1) Left: distribution of 13 subject categories with exemplar learning topics; ring width encodes video duration. (2) Middle: learning topic word cloud highlighting core concepts. (3) Right: average learning topic length per category.

We adopt *Manim* for its fine-grained spatiotemporal control, symbolic expressivity, and demonstrated effectiveness in expert-produced instructional videos.

3.2 Data Curation and Statistics

We construct MMMC, a benchmark for code-driven educational video generation, under two criteria: (i) *educational relevance*—each learning topic is an established concept worth teaching; and (ii) *executable grounding*—each concept aligns with a high-quality Manim reference, ensuring practical realizability. We source from the complete 3Blue1Brown (3B1B) YouTube corpus, known for its instructional impact and expert Manim craftsmanship. After filtering out non-instructional items (e.g., Q&A), we curate 117 long-form videos spanning 13 subject areas, including *calculus*, *geometry*, *probability*, and *neural networks*. To enrich supervision, we segment videos using author-provided timestamps into 339 semantically coherent sub-clips, yielding 456 units in total. An LLM then extracts concise learning topics (avg. 6.3 words) from titles, descriptions, and metadata, producing a clean mapping from videos to educationally grounded units (details in §A.1.5). On average, a full-length video lasts 1014 seconds (∼16.9 minutes), while a segmented clip spans 201 seconds (∼3.35 minutes), thus balancing long-horizon reasoning with fine-grained supervision. Figure 2 visualizes topical diversity with a hierarchical donut plot: the inner ring denotes 13 categories, while the outer ring shows individual topics with arc width proportional to cumulative duration. This structure highlights both the breadth of coverage and the temporal richness of MMMC, establishing it as a challenging and representative benchmark for educational video generation.

3.3 Evaluation Metrics

Unlike conventional video generation, educational videos are valued less for visual fidelity than for how effectively they convey knowledge. This makes standard synthesis metrics inadequate. We therefore design a three-pronged evaluation across **aesthetics**, **knowledge convey**, and **efficiency**:

VLM-as-Judges. Since human judgments of video quality are inherently subjective, we adopt a VLM-as-judges protocol ($\mathcal{P}_{\text{aesth}}$) to approximate user perception across five axes: *(i) Element Layout (EL)* — clarity and spatial arrangement of visual components. *(ii) Attractiveness (AT)* — overall engagement and ability to capture learners' attention. *(iii) Logic Flow (LF)* — coherence in temporal presentation of concepts. *(iv) Visual Consistency (VC)* — stylistic stability across frames and sections. *(v) Accuracy & Depth (AD)* — correctness and richness of the presented knowledge. Each dimension is rated on a 100-point scale.

TeachQuiz. The goal of educational video generation is not merely visual plausibility, but effective knowledge transfer. To evaluate this, we introduce TeachQuiz, a two-stage protocol grounded in a quiz set $\mathcal{Q}(\mathcal{K}) = (q_i, y_i)_{i=1}^{N}$ for a given concept $\mathcal{K}$, and Y denotes ground-truth answers. We consider multiple teachers α, β, each producing a video $\mathcal{V}_\alpha, \mathcal{V}_\beta$. A student model ϕ is tasked with watching the video and answering questions:

$$S(\mathcal{V}_\alpha) = \mathbf{1}[\phi\,(Q, \mathcal{V}_\alpha) = Y] \tag{1}$$

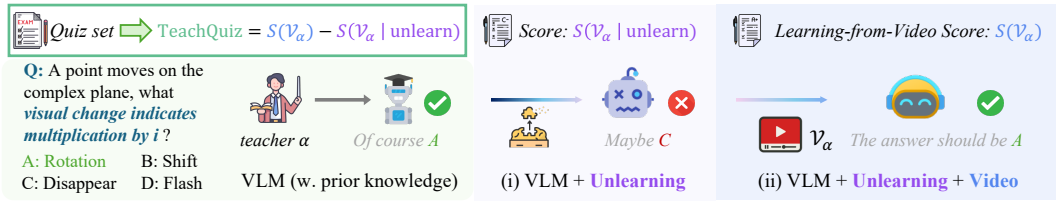


Figure 3: TeachQuiz: score gap between *Learning-from-Video* and *Unlearning* stages.

If $S(\mathcal{V}_\alpha) > S(\mathcal{V}_\beta)$, then teacher α is the stronger instructor.

However, a key challenge is that *many quiz items are already be learn by top-performing VLMs (i.e.,**answer correctly without watching the video**)*. Thus, absolute accuracy alone does not measure teaching quality. Instead, a good educational video should improve knowledge acquisition relative to a controlled baseline. We enforce this through two steps: **(i) Unlearning.** Apply $\mathcal{P}_{\text{unlearn}}$ to block prior access to $\mathcal{K}$, yielding a knowledge-removed baseline. **(ii) Learning-from-Video.** Expose the model to $\mathcal{V}$ under $\mathcal{P}_{\text{learn}}$, testing whether the video itself enables recovery of the knowledge. The final *TeachQuiz* score measures relative improvement:

$$\widetilde{S}(\mathcal{V}_\alpha) = S(\mathcal{V}_\alpha) - S(\mathcal{V}_\alpha|\text{unlearn}) \tag{2}$$

which isolates the contribution of the video by subtracting the unlearned baseline. Higher $\widetilde{S}$ indicates stronger knowledge transfer induced by the generated video.

Token Cost and Generation Time. Beyond output quality, an equally important dimension is how economically a model can generate effective videos. We measure efficiency by *average code generation time* and *token usage per video*, reflecting scalability and feasibility in large-scale or interactive educational settings where latency and resource costs are critical.

4 Method: Code2Video

Overview. As illustrated in Fig. 4, given a topic query $\mathcal{Q}$, Code2Video output a video $\mathcal{V}$, which is consists of three stages: **(i) Planner** structures topics into storyboards with reference assets, **(ii) Coder** translates each section into executable Manim code using parallel synthesis and an effective debugging, and **(iii) Critic** refines rendered videos through a novel visual prompt and VideoLLM feedback to ensure spatial coherence and educational clarity.

4.1 Planner: query to storyboard

Generating coherent educational videos requires careful organization of temporal structure. We design the Planner to decompose a topic $\mathcal{Q}$ into two stages: outline generation for high-level ordering, and storyboard construction for stepwise realization. This preserves logical flow while capturing cross-section dependencies.

(i) Outline Generation. Given a topic $\mathcal{Q}$, the Planner produces an outline $\mathcal{O} = o_1, \ldots, o_n$, where each o_i contains a unique identifier, section title, content summary, and illustrative examples. Crucially, the Planner also considers the intended audience (*e.g.,*trigonometric functions for middle school, Fourier's law for undergraduates), ensuring level-appropriate structure. Formally, $\mathcal{O} \leftarrow \mathcal{P}_{\text{outline}}(\mathcal{Q})$, where $\mathcal{O} = \{o_1, \cdots, o_n\}$ and each o_i encodes the section-level metadata and educational intent. By explicitly specifying audience and structure, the outline establishes the temporal skeleton for the subsequent video, guiding both pacing and sequencing.

(ii) Storyboard Construction. The second stage converts the outline o into a detailed storyboard s. Each section in s includes title, lecture lines, and corresponding animations, with $s_i \leftarrow \mathcal{P}_{\text{storyboard}}(o_i)$. The storyboard specifies the temporal sequence of lecture lines and paired animations, bridging high-level planning with concrete visual content.

External Database. To enhance factual accuracy and visual fidelity, the Planner integrates an external database $\mathcal{D}$. It includes *(a) reference images* aligned with the topic to anchor complex concepts and reduce hallucination, and *(b) visual assets* (*e.g.,*icons, logos) that are difficult to generate from

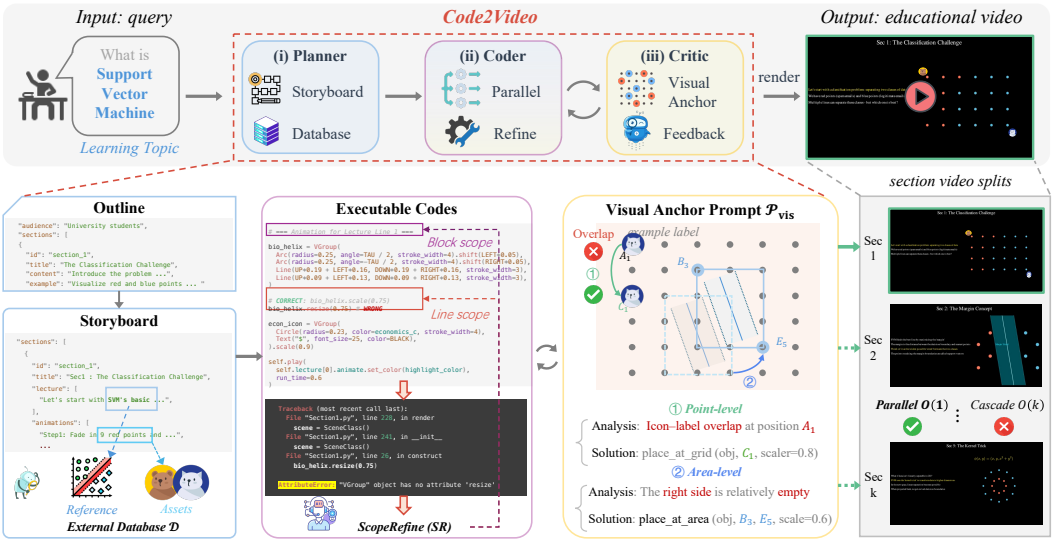


Figure 4: **Illustration of Code2Video.** Given a user inquiry, Code2Video aims to render an educational video via Manim code writing: **(i) the Planner** converts a learning topic into a storyboard and retrieves visual assets; **(ii) the Coder** performs parallel code synthesis with scope-guided refinement to ensure efficiency and temporal consistency; **(iii) the Critic** uses visual anchor prompts to iteratively adjust spatial layout and clarity, yielding reproducible, educationally structured videos.

scratch. These assets $\mathcal{A}$ are automatically identified via a prompt $\mathcal{P}_{\text{asset}}$ analyzing the storyboard, $a_i \leftarrow \mathcal{P}_{\text{asset}}(s_i)$, and stored in a persistent cache $\mathcal{D}_{\text{asset}}$. Caching enables reuse across sections, preventing redundant generation and ensuring visual consistency. Please refer to § A.1.6 for more details and examples about $\mathcal{D}$.

4.2 Coder: Storyboards to Executable Code

The Coder $\mathcal{G}$ translates each section of the storyboard s and the cached assets a into executable Manim code $C = \{c_1, \ldots, c_n\}$, where each c_i corresponds to a storyboard s_i.

(i) Parallel Code Generation. A central bottleneck in full-code synthesis is generation time, as end-to-end Manim code production for a single educational video—including generation, debugging, and rendering—can exceed two hours. To address this bottleneck, we parallelize the pipeline by decoupling serial steps—code generation, debugging, and refinement—so that each section is synthesized and fixed independently. Each section is conditioned on its storyboard and shared assets $\mathcal{A}$: $c_i = \mathcal{P}_{\text{coder}}(s_i, \mathcal{A})$. Notably, asset sharing across sections ensures temporal consistency while retaining the efficiency benefits of parallelization.

(ii) Effective Debugging. Even strong LLMs rarely produce fully executable code in one pass. Naïve strategies that concatenate all code with the full error log are costly in both time and tokens. We propose **ScopeRefine (SR)**, a hierarchical, scope-guided repair strategy, as illustrated in Fig.4 middle bottom: *(a) Line scope:* isolate the error line plus immediate context, $\mathcal{S}_1 = \text{line} \pm 1$, attempt up to K_1 local fixes. *(b) Block scope:* if unresolved, expand to the lecture-line block $\mathcal{S}_2 = \mathcal{B}_{i,j}$ with up to K_2 repair attempts. *(c) Global scope:* as a last resort, regenerate the full section c_i from s_i. This progressive *"Go-to style" repair* minimizes token usage and latency while ensuring high reliability, effectively bridging parallel generation with robust debugging.

4.3 Critic: Effective Visual Refinement

Even after debugging ensures executability, the generated code may still yield unsatisfactory visual outcomes. LLMs and VLMs often fail to provide actionable feedback due to **limited spatial awareness** [5, 63]. In practice, models can identify issues (*e.g.,*"the cat icon is misplaced") but struggle to provide actionable corrections. They often fail to indicate the direction or distance needed to adjust the element, which makes text-only refinement inadequate.

(i) Visual Anchor Prompt ($\mathcal{P}_{\text{vis}}$). We introduce $\mathcal{P}_{\text{vis}}$, a textual prompt that discretizes the 2D canvas into a 6×6 grid of predefined anchor points. Each grid cell is mapped to fixed Manim coordinates, allowing LLM-specified locations to be directly converted into executable code. Placement follows two granularities, as illustrated in Figure 5: *(a) point-level*, where small elements (*e.g.,*symbols, short labels) occupy a single anchor; and *(b) region-level*, where larger elements are assigned to a bounding box spanning multiple anchors. This discretization transforms the placement task from a *continuous positioning problem* into a *discrete anchoring problem*, serving as a visual debugging *"go-to"*, which substantially reduces the difficulty for LLMs to produce valid layouts.

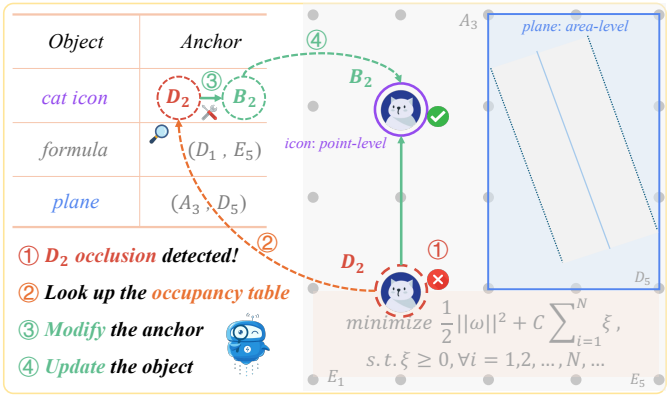


Figure 5: Illustration of *visual anchor prompt ($\mathcal{P}_{\text{vis}}$)*.

(ii) VideoLLM for Code Feedback. To detect violations and refine placement, the Critic inspects the rendered video $\mathcal{V}_i$ alongside its section code c_i. During parallel code generation, we maintain an *occupancy table* that records each element's assigned anchors (point or region), scaling factor, and corresponding code lines. This design serves two purposes: (a) it makes all assets indexable, allowing the Critic to quickly trace a visual issue back to its source code; and (b) it reveals available anchors, enabling conflict-free reallocation. With this structured view, the Critic efficiently detects three common issues: overlapping elements within a cell, lecture lines occluded by animations, and large unused regions creating visual imbalance. These findings are incorporated into a refinement prompt $\mathcal{P}_{\text{refine}}$, yielding optimized code: $\tilde{c}_i = \mathcal{P}_{\text{refine}}(c_i, \mathcal{V}_i)$ and final video $\widetilde{\mathcal{V}} = \text{Render}(\{\tilde{c}_i\}_{i=1}^{n})$. By combining anchor-based guidance, indexable, occupancy-aware adjustment, and multimodal feedback, the Critic overcomes the limitations of text-only debugging.

5 Experiment

5.1 Implementation Details

Baselines. We compare four types of approaches: ◇ *Human-crafted*, expert-designed Manim videos as an upper bound; ◇ *Pixel-based Diffusion*, text-to-video models: *OpenSora-v2* [37], *Wan2.2-T2V-A14B* [43], and *Veo3* [9]; ◇ *CodeLLM Generation*, where an LLM directly generates Manim code from a learning topic; ◇ *Agentic Generation (**ours**)*, a Planner–Coder–Critic pipeline. We evaluate across diverse models: *Claude Opus 4.1* [1], *GPT-4o*, *GPT-o4 mini*, *GPT-4.1*, *GPT-5* [34], *Gemini-2.5 Pro* [18], with *Gemini-2.5 Pro* serving as Critic for refinement. **Evaluation.** Aesthetics are judged by *Gemini-2.5 Pro* (VLM-as-a-Judge), and quantify knowledge transfer with TeachQuiz. **Resources.** Reference images are retrieved from Google Images, and visual assets from Iconfinder[1]. All prompts are documented in § A.2.

5.2 Main Results

Table 1 compares Code2Video with human-crafted videos, pixel-based models, and code LLM baselines, evaluated on Efficiency, Aesthetics (AES), and knowledge transfer (TeachQuiz). Our analysis yields several insights: **(i) Pixel-based models underperform.** They obtain the lowest scores on both AES and TeachQuiz, particularly struggling with LF due to weak control over text grounding, animation timing, and cross-frame coherence. **(ii) Direct code-centric generation delivers clear improvements**. Rendering videos from LLM-produced Manim code outperforms pixel-based models, underscoring code as an effective medium for controllable and coherent educational video generation. **(iii) Our agentic framework delivers stable and consistent improvements.** Across different backbone LLMs, Code2Video achieves significant performance boosts. For instance, with Claude Opus 4.1, AES improves by 50% and TeachQuiz by 46%. These gains arise from distinct components:

[1]https://www.iconfinder.com

Table 1: Results across Efficiency, Aesthetics, and TeachQuiz (Quiz). Efficiency: Time (**avg minutes** per topic) and Token (avg **token consumption** per topic). Aesthetics: Element Layout (EL), Attractiveness (AT), Logic Flow (LF), Visual Consistency (VC), Accuracy & Depth (AD).

Method	Efficiency (↓)		Aesthetics (↑)						Quiz (↑)
	Time	Token (K)	EL	AT	LF	VC	AD	**Avg**	
Human-made 3B1B	–	–	98.3	100	100	100	100	99.7	97.1
Pixel-based Diffusion									
OpenSora-v2	27.6	–	0.0	5.0	0.0	0.0	13.3	3.7	0.0
Wan2.2-T2V-A14B	17.4	–	0.0	10.0	0.0	0.0	20.0	6.0	0.0
Veo3	2.3	–	0.0	15.0	0.0	5.0	25.0	9.0	2.5
Code LLM									
GPT-5	1.8	1.1	27.0	28.0	28.0	54.5	26.0	32.7	36.5
GPT-4.1	2.1	1.2	30.5	34.5	39.0	42.0	24.8	34.2	37.0
Claude Opus 4.1	2.8	2.3	47.5	40.0	26.5	56.6	18.4	37.8	40.0
Code2Video Agent (Ours)									
Code2Video $_{\text{Gemini-2.5 Pro}}$	15.5	41.8	70.3	60.3	44.3	37.6	74.7	57.4	72.0
Code2Video $_{\text{GPT-4o}}$	14.1	32.7	70.3	58.3	54.6	48.5	68.3	60.0	44.0
Code2Video $_{\text{GPT-o4 mini}}$	16.8	49.2	77.0	52.8	73.0	57.2	79.0	67.8	48.5
Code2Video $_{\text{GPT-5}}$	8.8	19.3	75.5	60.5	81.8	63.6	79.7	72.2 +39.5	80.0 +43.5
Code2Video $_{\text{GPT-4.1}}$	15.4	30.8	82.8	65.6	95.0	68.0	83.7	79.0 +44.8	82.0 +45.0
Code2Video $_{\text{Claude Opus 4.1}}$	13.8	43.1	90.6	79.7	93.3	84.2	91.9	**87.9** +50.1	**86.0** +46.0

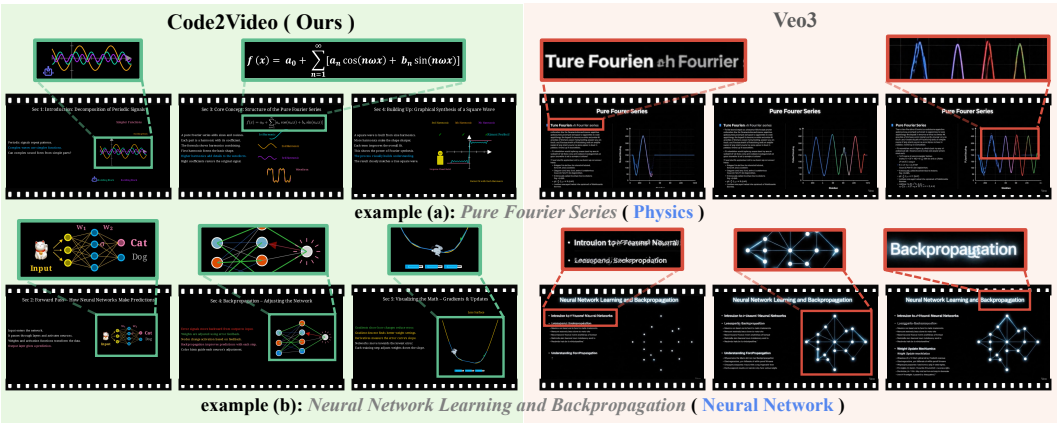


Figure 6: Qualitative comparison between *Code2Video* and *Veo3*. Our approach generates videos with coherent logic flow, consistent semantics, and interpretable layouts.

visual anchor points drive improvements in element layout, while the Planner enhances LF and AD. However, limitations remain in AT and VC, pointing to opportunities for refinement. **(iv) Human-made videos remain strong.** Although Code2Video narrows the gap, professional videos still lead in storytelling, nuanced sequencing, and explanatory depth. This highlights the next frontier: advancing agentic pipelines toward ***professional-quality long educational videos***.

Qualitative Analyses. Figure 6 illustrates that our code-driven pipeline produces videos with clear text and formulas, stable layouts without occlusions, and stepwise alignment with lecture lines. In contrast, the pixel-based model (Veo3) often generates blurry or corrupted text, inconsistent styles, and drifting visuals, weakening semantic grounding. Overall, code-driven synthesis ensures better spatial stability and clearer knowledge presentation. Additional cases are provided in § A.1.7.

5.3 Ablation Studies

Effects by Individual Components. Table 2 highlights several observations. First, TeachQuiz is more sensitive than Aesthetics, revealing *knowledge-transfer gaps even when videos still look*

visually acceptable. Second, the Planner is essential: removing it collapses both metrics (≈ 41 points), underscoring that high-level lecture planning and temporal sequencing are the backbone of effective teaching videos. Third, other modules provide complementary gains: the External Database improves conceptual grounding, Visual Anchors stabilize layouts, and the Critic ensures refinement—each modest alone, but jointly essential for robustness. These results **highlight that structured visual guidance and iterative refinement are crucial** for producing visually clear videos that effectively convey knowledge.

Table 2: Effect of different components on quality: TeachQuiz / Aesthetics avg. score.

Method	Aesthetics	Quiz
Code2Video $_{\text{Chat-4.1}}$ ($\diamond$)	**79.0**	**82.0**
$\diamond$ w/o Planner	38.1 $_{-40.9}$	40.5 $_{-41.5}$
$\diamond$ w/o External Database	68.1 $_{-10.9}$	52.0 $_{-30.0}$
$\diamond$ w/o Visual Anchor	69.2 $_{-9.8}$	55.2 $_{-26.8}$
$\diamond$ w/o Critic	72.5 $_{-6.5}$	60.7 $_{-21.3}$

Table 3: Effect of efficiency components: runtime avg. time / token consumption.

Method	Time (m)	Token (K)
Code2Video $_{\text{Chat-4.1}}$ ($\diamond$)	**15.4**	**30.8**
$\diamond$ w/o parallel	86.6 $_{5.6\times}$	30.8
$\diamond$ w/o SR $\rightarrow$ w. Retry	42.9 $_{2.8\times}$	49.8 $_{1.6\times}$
$\diamond$ w/o SR $\rightarrow$ w. Debug	39.2 $_{2.5\times}$	42.1 $_{1.4\times}$
$\diamond$ w/o parallel & SR	149.8 $_{9.7\times}$	52.6 $_{1.7\times}$

Efficiency Components. Table 3 evaluates efficiency-oriented modules. Removing parallel execution greatly increases latency ($15.4 \rightarrow 86.6$ minutes). Without ScopeRefine (SR), we test two alternative debugging methods: (i) *Retry*, which regenerates the section upon any error; (ii) *Full-code Debug*, which feeds the entire code and error log to the LLM to regenerate the section. In both cases, error correction is costly, highlighting the importance of SR's localized, scope-aware repair. Removing both mechanisms produces prohibitive overheads. These results underscore that parallel synthesis and scope-aware repair are essential for scalable, code-centric video generation.

Table 4: **Human study** on Aesthetics, TeachQuiz (Quiz), Completion Willingness (CW), and Average Ranking (AR). Results align with VLM-based trends but show sharper score contrast, lower tolerance for layout errors, and reduced engagement in longer-duration videos.

Method	Duration	Aesthetics (↑)						Quiz (↑)	CW (↑)	AR (↓)
		EL	AT	LF	VC	AD	Avg			
Human-made 3B1B	16.9 min	98.9	97.2	91.3	98.0	97.0	96.5	78.8	36.2	1.2
Pixel-based $_{\text{Veo3}}$	8.0 s	12.6	4.4	1.1	24.4	1.1	8.5	8.0	46.8	5.0
Code LLM $_{\text{Claude Opus 4.1}}$	0.9 min	16.1	41.1	55.6	71.1	72.2	51.2	56.6	15.0	3.9
Code2Video $_{\text{Gemini-2.5 Pro}}$	1.6 min	26.7	68.3	78.1	90.2	81.0	68.9	65.3	47.4	3.1
Code2Video $_{\text{Claude Opus 4.1}}$	2.0 min	60.2	89.3	84.6	92.0	83.1	81.8	**80.3**	**64.0**	1.8

Human Study Evaluation. We conduct a five-group user study (6 middle school, 2 undergraduate volunteers per group), where each participant watches one video type and answers 5 quiz questions for 20 learning topics. We measure Completion Willingness (**CW**, proportion finishing the video before answering, max score is 100) and Average Ranking (**AR**, mean preference across video types, 1 is the best). Table 4 reveals four patterns: **(i) Clearer separation.** Human ratings follow the same overall trends as VLM-based scores but with stronger contrast: high-quality videos are rated in the upper range (> 90), while low-quality videos cluster near the lower bound (< 10). **(ii) Sensitivity to layout errors.** Participants give lower layout scores (EL) to videos from Code2Video, as humans are highly sensitive to even brief occlusions, whereas VideoLLMs often miss such frame-level issues. **(iii) Attention span limits.** Human attention is inherently limited: to perform well on the quiz, participants must follow the full flow of knowledge details in the video. This requires not only *strong logical coherence* and *engaging presentation* but also a *reasonable duration* that allows sustained high attention for effective knowledge absorption. **(iv) Strong consistency.** Aesthetics and TeachQuiz scores are strongly correlated($r = 0.971$, $p = 0.0059$): visually appealing videos keep students engaged, leading to higher learning outcomes. Overall, the human study underscores that both structural clarity and visual appeal are decisive levers for learning efficacy, complementing the automated metrics. *Future work requires agent designs that explicitly account for* ***human attention and patience****, ensuring videos maintain* ***fine-grained details*** *while* ***minimizing perceptual fatigue****.*

6 Conclusion

We have introduced a novel, code-centric paradigm for educational video generation, establishing executable code as the unifying medium for both temporal sequencing and spatial organization. Building on this paradigm, our tri-agent architecture *Code2Video* enables controllable and interpretable generation with multimodal feedback. To systematically evaluate this paradigm, we introduce *MMMC*, targeting efficiency, aesthetics, and knowledge transfer. Together, our paradigm, architecture, and benchmark chart a clear path for future research on leveraging code as a medium for high-quality, structured, and interpretable educational content generation. Future work includes broadening the video scope and developing more lightweight, scalable agent frameworks.

References

[1] Anthropic. Claude opus 4.1. https://www.anthropic.com/claude, 2025.

[2] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *International conference on learning representations*, 2019.

[3] Lei Bao, Tianfan Cai, Kathy Koenig, Kai Fang, Jing Han, Jing Wang, Qing Liu, Lin Ding, Lili Cui, Ying Luo, et al. Learning and scientific reasoning. *Science*, 323(5914):586–587, 2009.

[4] Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, et al. Lumiere: A space-time diffusion model for video generation. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024.

[5] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. Spatialrgpt: Grounded spatial reasoning in vision-language models. *Advances in Neural Information Processing Systems*, 37:135062–135093, 2024.

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

[7] Yihong Dong, Xue Jiang, Jiaru Qian, Tian Wang, Kechi Zhang, Zhi Jin, and Ge Li. A survey on code generation with llm-based agents. *arXiv preprint arXiv:2508.00083*, 2025.

[8] Difei Gao, Siyuan Hu, Zechen Bai, Qinghong Lin, and Mike Zheng Shou. Assisteditor: Multi-agent collaboration for gui workflow automation in video creation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 11255–11257, 2024.

[9] Google DeepMind. Veo 3: Generative video model. https://deepmind.google/technologies/veo/, 2025.

[10] Yuchao Gu, Weijia Mao, and Mike Zheng Shou. Long-context autoregressive video modeling with next-frame prediction. *arXiv preprint arXiv:2503.19325*, 2025.

[11] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14953–14962, 2023.

[12] Tanmay Gupta, Luca Weihs, and Aniruddha Kembhavi. Codenav: Beyond tool-use to using real-world codebases with llm agents. *arXiv preprint arXiv:2406.12276*, 2024.

[13] Roberto Henschel, Levon Khachatryan, Hayk Poghosyan, Daniil Hayrapetyan, Vahram Tadevosyan, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Streamingt2v: Consistent, dynamic, and extendable long video generation from text. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2568–2577, 2025.

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

[15] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

[16] Panwen Hu, Jin Jiang, Jianqi Chen, Mingfei Han, Shengcai Liao, Xiaojun Chang, and Xiaodan Liang. Storyagent: Customized storytelling video generation via multi-agent collaboration. *arXiv preprint arXiv:2411.04925*, 2024.

[17] Kaiyi Huang, Yukun Huang, Xuefei Ning, Zinan Lin, Yu Wang, and Xihui Liu. Genmac: compositional text-to-video generation with multi-agent collaboration. *arXiv preprint arXiv:2412.04440*, 2024.

[18] Muhammad Imran and Norah Almusharraf. Google gemini as a next generation ai educational tool: a review of emerging educational technology. *Smart Learning Environments*, 11(1):22, 2024.

[19] Vyoman Jain, Shiva Golugula, Motamarri Sai Sathvik, et al. Manimator: Transforming research papers into visual explanations. *arXiv preprint arXiv:2507.14306*, 2025.

[20] Yilei Jiang, Yaozhi Zheng, Yuxuan Wan, Jiaming Han, Qunzhong Wang, Michael R Lyu, and Xiangyu Yue. Screencoder: Advancing visual-to-code generation for front-end automation via modular multimodal agents. *arXiv preprint arXiv:2507.22827*, 2025.

[21] Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, et al. Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445*, 2022.

[22] Max Ku, Thomas Chong, Jonathan Leung, Krish Shah, Alvin Yu, and Wenhu Chen. Theoremexplainagent: Towards video-based multimodal explanations for llm theorem understanding. *arXiv preprint arXiv:2502.19400*, 2025.

[23] Chengxuan Li, Di Huang, Zeyu Lu, Yang Xiao, Qingqi Pei, and Lei Bai. A survey on long video generation: Challenges, methods, and prospects. *arXiv preprint arXiv:2403.16407*, 2024.

[24] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008, 2023.

[25] Xinzhe Li. A review of prominent paradigms for llm-based agents: Tool use (including rag), planning, and feedback learning. *arXiv preprint arXiv:2406.05804*, 2024.

[26] Xinzhe Li. A review of prominent paradigms for llm-based agents: Tool use, planning (including rag), and feedback learning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9760–9779, 2025.

[27] Zongyi Li, Shujie Hu, Shujie Liu, Long Zhou, Jeongsoo Choi, Lingwei Meng, Xun Guo, Jinyu Li, Hefei Ling, and Furu Wei. Arlon: Boosting diffusion transformers with autoregressive models for long video generation. *arXiv preprint arXiv:2410.20502*, 2024.

[28] Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19498–19508, 2025.

[29] Kaiyuan Liu, Youcheng Pan, Yang Xiang, Daojing He, Jing Li, Yexing Du, and Tianrun Gao. Projecteval: A benchmark for programming agents automated evaluation on project-level code generation. *arXiv preprint arXiv:2503.07010*, 2025.

[30] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024.

[31] Yu Lu, Yuanzhi Liang, Linchao Zhu, and Yi Yang. Freelong: Training-free long video generation with spectralblend temporal attention. *Advances in Neural Information Processing Systems*, 37:131434–131455, 2024.

[32] Manim Community Dev. Manim community v0.19.0. https://github.com/ManimCommunity/manim, 2025.

[33] Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A Rodriguez, Montek Kalsi, Rabiul Awal, Nicolas Chapados, M Tamer Özsu, Aishwarya Agrawal, David Vazquez, et al. Ui-vision: A desktop-centric gui benchmark for visual perception and interaction. *arXiv preprint arXiv:2503.15661*, 2025.

[34] OpenAI. Chatgpt-series. https://openai.com, 2025. Large language model released by OpenAI.

[35] Wei Pang, Kevin Qinghong Lin, Xiangru Jian, Xi He, and Philip Torr. Paper2poster: Towards multimodal poster automation from scientific papers. *arXiv preprint arXiv:2505.21497*, 2025.

[36] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

[37] Xiangyu Peng, Zangwei Zheng, Chenhui Shen, Tom Young, Xinying Guo, Binluo Wang, Hang Xu, Hongxin Liu, Mingyan Jiang, Wenjun Li, et al. Open-sora 2.0: Training a commercial-level video generation model in 200 k. *arXiv preprint arXiv:2503.09642*, 2025.

[38] Sina Rismanchian, Yasaman Razeghi, Sameer Singh, and Shayan Doroudi. Turtlebench: A visual programming benchmark in turtle geometry. *arXiv preprint arXiv:2411.00264*, 2024.

[39] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

[40] Leixian Shen, Haotian Li, Yun Wang, and Huamin Qu. From data to story: Towards automatic animated data video creation with llm-based multi-agent systems. In *2024 IEEE VIS Workshop on Data Storytelling in an Era of Generative AI (GEN4DS)*, pages 20–27. IEEE, 2024.

[41] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

[42] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11888–11898, 2023.

[43] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[44] Hongru Wang, Cheng Qian, Manling Li, Jiahao Qiu, Boyang Xue, Mengdi Wang, Heng Ji, and Kam-Fai Wong. Toward a theory of agents as tool-use decision-makers. *arXiv preprint arXiv:2506.00886*, 2025.

[45] Yuqing Wang, Tianwei Xiong, Daquan Zhou, Zhijie Lin, Yang Zhao, Bingyi Kang, Jiashi Feng, and Xihui Liu. Loong: Generating minute-level long videos with autoregressive language models. *arXiv preprint arXiv:2410.02757*, 2024.

[46] Jingxuan Wei, Cheng Tan, Qi Chen, Gaowei Wu, Siyuan Li, Zhangyang Gao, Linzhuang Sun, Bihui Yu, and Ruifeng Guo. From words to structured visuals: A benchmark and framework for text-to-diagram generation and editing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 13315–13325, 2025.

[47] Chao Wen, Jacqueline Staub, and Adish Singla. Program synthesis benchmark for visual programming in xlogoonline environment. *arXiv preprint arXiv:2406.11334*, 2024.

[48] Wenming Weng, Ruoyu Feng, Yanhui Wang, Qi Dai, Chunyu Wang, Dacheng Yin, Zhiyuan Zhao, Kai Qiu, Jianmin Bao, Yuhui Yuan, et al. Art-v: Auto-regressive text-to-video generation with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7395–7405, 2024.

[49] Zejia Weng, Xitong Yang, Zhen Xing, Zuxuan Wu, and Yu-Gang Jiang. Genrec: Unifying video generation and recognition with diffusion models. *Advances in Neural Information Processing Systems*, 37:108851–108876, 2024.

[50] Chengyue Wu, Yixiao Ge, Qiushan Guo, Jiahao Wang, Zhixuan Liang, Zeyu Lu, Ying Shan, and Ping Luo. Plot2code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. *arXiv preprint arXiv:2405.07990*, 2024.

[51] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

[52] Desai Xie, Zhan Xu, Yicong Hong, Hao Tan, Difan Liu, Feng Liu, Arie Kaufman, and Yang Zhou. Progressive autoregressive video diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6322–6332, 2025.

[53] Zhifei Xie, Daniel Tang, Dingwei Tan, Jacques Klein, Tegawend F Bissyand, and Saad Ezzini. Dreamfactory: Pioneering multi-scene long video generation with a multi-agent framework. *arXiv preprint arXiv:2408.11788*, 2024.

[54] Guangming Xing, Tawfiq Salem, and Gongbo Liang. Chartcode: A flowchart-based tool for introductory programming courses. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2*, pages 1665–1666, 2025.

[55] Jinbo Xing, Menghan Xia, Yuxin Liu, Yuechen Zhang, Yong Zhang, Yingqing He, Hanyuan Liu, Haoxin Chen, Xiaodong Cun, Xintao Wang, et al. Make-your-video: Customized video generation using textual and structural guidance. *IEEE Transactions on Visualization and Computer Graphics*, 31(2):1526–1541, 2024.

[56] Weikai Xu, Chengrui Huang, Shen Gao, and Shuo Shang. Llm-based agents for tool learning: A survey: W. xu et al. *Data Science and Engineering*, pages 1–31, 2025.

[57] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

[58] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[59] Hui Ye, Chufeng Xiao, Jiaye Leng, Pengfei Xu, and Hongbo Fu. Mographgpt: Creating interactive scenes using modular llm and graphical control. *arXiv preprint arXiv:2502.04983*, 2025.

[60] Ryan Yen, Jian Zhao, and Daniel Vogel. Code shaping: Iterative code editing with free-form ai-interpreted sketching. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2025.

[61] Hangjie Yuan, Weihua Chen, Jun Cen, Hu Yu, Jingyun Liang, Shuning Chang, Zhihui Lin, Tao Feng, Pengwei Liu, Jiazheng Xing, et al. Lumos-1: On autoregressive video generation from a unified model perspective. *arXiv preprint arXiv:2507.08801*, 2025.

[62] Zhengqing Yuan, Yixin Liu, Yihan Cao, Weixiang Sun, Haolong Jia, Ruoxi Chen, Zhaoxu Li, Bin Lin, Li Yuan, Lifang He, et al. Mora: Enabling generalist video generation via a multi-agent framework. *arXiv preprint arXiv:2403.13248*, 2024.

[63] Jirong Zha, Yuxuan Fan, Xiao Yang, Chen Gao, and Xinlei Chen. How to enable llm with 3d capacity? a survey of spatial reasoning in llm. *arXiv preprint arXiv:2504.05786*, 2025.

[64] Zhehao Zhang, Ryan Rossi, Tong Yu, Franck Dernoncourt, Ruiyi Zhang, Jiuxiang Gu, Sungchul Kim, Xiang Chen, Zichao Wang, and Nedim Lipka. Vipact: Visual-perception enhancement via specialized vlm agent collaboration and tool-use. *arXiv preprint arXiv:2410.16400*, 2024.

[65] Xuanle Zhao, Xianzhen Luo, Qi Shi, Chi Chen, Shuo Wang, Zhiyuan Liu, and Maosong Sun. Chartcoder: Advancing multimodal large language model for chart-to-code generation. *arXiv preprint arXiv:2501.06598*, 2025.

[66] Zixiao Zhao, Jing Sun, Zhe Hou, Zhiyuan Wei, Cheng-Hao Cai, Miao Qiao, and Jin Song Dong. Mactg: Multi-agent collaborative thought graph for automatic programming. *arXiv preprint arXiv:2410.19245*, 2024.

[67] Yupeng Zhou, Daquan Zhou, Ming-Ming Cheng, Jiashi Feng, and Qibin Hou. Storydiffusion: Consistent self-attention for long-range image and video generation. *Advances in Neural Information Processing Systems*, 37:110315–110340, 2024.

A Supplementary Material

A.1 Additional Implementation Details and Experiments

A.1.1 Unlearning Details and TeachQuiz

To probe whether generated tutorial videos genuinely transfer knowledge, we integrate a selective unlearning–relearning protocol into the TeachQuiz evaluation.

Model choice. We adopt *Gemini-2.5 Pro* [18], one of the current state-of-the-art models in video understanding. Its closed-source nature precludes parameter-level interventions for unlearning; thus, we rely on a prompt-based strategy, a standard approach for steering proprietary models.

Unlearning stage. We design a parameter-free pipeline $\mathcal{P}_{\text{unlearn}}$ tailored for closed-source models. Given a target concept $\mathcal{K}$, we define a shadow knowledge set $\mathcal{B}(\mathcal{K})$ consisting of canonical definitions, formulas, aliases, and exemplars associated with $\mathcal{K}$. During inference, $\mathcal{P}_{\text{unlearn}}$ enforces: (i) *contextual masking*, where $\mathcal{B}(\mathcal{K})$ is silently identified and treated as inaccessible; (ii) *uncertainty injection*, where the model must output "*INSUFFICIENT EVIDENCE*" whenever the reasoning chain depends on elements of $\mathcal{B}(\mathcal{K})$; (iii) *progressive forgetting validation*, where queries of increasing difficulty $\{q_i\}_{i=1}^{N}$ are used to test suppression not only at recall-level but also across multi-step reasoning. Formally, the model's answer distribution is constrained to

$$f(q_i \mid \mathcal{P}_{\text{unlearn}}) \in \{y_i, \texttt{NULL}\}, \tag{3}$$

where NULL indicates blocked inference. This layered design obstructs both direct recall and indirect reconstruction, ensuring that performance degradation reflects genuine unlearning rather than prompt compliance artifacts.

Relearning stage. We then expose the model to an educational video $\mathcal{V}$ and apply a relearning prompt $\mathcal{P}_{\text{learn}}$, which restricts evidence scope to $\mathcal{V}$ while maintaining the block on $\mathcal{B}(\mathcal{K})$. The answering constraint becomes

$$f(q_i \mid \mathcal{P}_{\text{learn}}, \mathcal{V}) \in \{y_i, \texttt{NULL}\}, \tag{4}$$

with justification required to reference only cues present in $\mathcal{V}$. This ensures that any gain after relearning is attributable solely to video-grounded evidence rather than residual prior knowledge.

Evaluation setup. For each learning topic, we construct 10 multiple-choice questions with four options (A–D), each containing exactly one correct answer. To better capture the expressive power of tutorial videos, these quizzes emphasize visually grounded reasoning. For instance, rather than simply asking *"What is the definition of a complex number?"*, a question may ask *"When a point moves on the complex plane, what visual transformation corresponds to multiplication by i?"*. Such queries demand alignment between knowledge and its visual instantiation.

Metric. Given a concept $\mathcal{K}$, we construct N multiple-choice questions $\{q_i\}_{i=1}^{N}$ with ground-truth answers $\{y_i\}_{i=1}^{N}$. The selective unlearning baseline $S_1(\mathcal{K})$ denotes the fraction of correctly answered questions under P_{unlearn}, where access to prior knowledge of $\mathcal{K}$ is explicitly blocked. We then compute the relearning accuracy $S_2(\mathcal{K}, \mathcal{V})$, defined as the fraction of correct answers when re-prompted with P_{learn} while exposing the model to the generated educational video $\mathcal{V}$. Formally,

The *TeachQuiz* score is then defined as:

$$\mathrm{TQ}(\mathcal{K}, \mathcal{V}) = S_2(\mathcal{K}, \mathcal{V}) - S_1(\mathcal{K}),$$

which captures the relative gain in accuracy attributable solely to $\mathcal{V}$. Intuitively, S_1 reflects how well the model resists using forbidden prior knowledge, while S_2 reflects how much can be recovered from the video. A higher TQ thus indicates stronger video-induced knowledge acquisition.

Ablation on evidence sources. To ensure that the observed gains are indeed attributable to the generated videos, we conduct an ablation study, shown in Table 5.

First, when providing only **Text-only** lecture lines (akin to PDF-style slides without animation), performance improves moderately compared to the unlearn baseline but falls short of full video-based relearning, highlighting that textual scaffolding alone is insufficient.

Second, with **Animation-only** inputs (animations without accompanying lecture text), accuracy also rises above unlearn but remains lower than the full condition, suggesting that temporal visual cues contribute substantially but require textual grounding for maximum effect.

Table 5: Ablation on unlearning. Accuracy reports correct concept judgments; $\Delta = \text{TQ}$ denotes the improvement in TeachQuiz confidence from the Unlearn setting to the Relearn setting. Text-only/Animation/Random evaluate TeachQuiz (TQ) under partial or mismatched supervision.

Method	Accuracy			TeachQuiz (TQ)		
	Unlearn	Relearn	$\Delta = \text{TQ}$	Text-only	Animation	Random
Code2Video $_{\text{GPT-5}}$	5.0	85.0	80.0	27.2	72.1	2.0
Code2Video $_{\text{GPT-4.1}}$	5.0	87.0	82.0	22.1	75.0	5.0
Code2Video $_{\text{Claude Opus 4.1}}$	5.0	91.0	86.0	24.0	76.6	4.0

Finally, in the **Random-video** setting, where the VLM is paired with an unrelated topic video, performance collapses to the unlearn level (or lower), confirming that improvements do not stem from superficial video exposure but rather from semantically aligned educational content.

Overall, these results provide evidence that the generated videos drive knowledge reacquisition: text and animation are complementary, and their synergy yields the strongest TeachQuiz gains.

A.1.2 Human Study: Middle School vs. Undergraduate Comparison

Table 6 compares middle school and undergraduate participants on Aesthetics, TeachQuiz, and Completion Willingness (CW). As TeachQuiz measures knowledge acquisition, middle school students—closer to a true "unlearned" state—benefit more from effective videos, showing substantial TeachQuiz gains (e.g., Code2Video boosts middle school TeachQuiz to 88.1 versus 55.0 for undergraduates). Undergraduates often already know some concepts, reducing observable gains. Across both groups, Code2Video achieves high Aesthetics and CW, outperforming pixel-based models by large margins. Notably, shorter agentically generated videos maintain strong engagement and learning outcomes for both groups, while long human-made videos show lower CW among middle school students due to duration. Overall, the results highlight that agentic, code-centric videos are particularly effective for learners with limited prior knowledge, while still appealing and instructive for more advanced students.

Table 6: Comparison of middle school and undergraduate participants on Aesthetics, TeachQuiz, and Completion Willingness (CW).

Method	Duration	Middle School			Undergraduate		
		Aesthetics	TeachQuiz	CW	Aesthetics	TeachQuiz	CW
Human-made 3B1B	16.9 min	96.3	**86.3**	34.9	97.5	56.0	40.2
Pixel-based $_{\textbf{Veo3}}$	8.0 s	10.7	**6.0**	55.6	2.0	14.0	20.5
Code2Video $_{\textbf{Claude Opus 4.1}}$	2.0 min	81.7	**88.1**	76.0	82.2	55.0	58.2

A.1.3 Ablation on Visual Anchor Point Granularity

Table 7: Ablation on anchor point granularity $\mathcal{P}_{\text{vis}}$. Structured anchors significantly improve layout and aesthetics, with a 6×6 grid yielding the best trade-off. Finer grids (e.g., 8×8) cause clutter, while unconstrained (Self-directed) placement underperforms due to inconsistent spacing.

# Anchor Points	AES			AES Avg
	Element Layout (EL)	Attractivenss (AT)	(EL + AT) / 2	
w/o Visual Anchor Prompt	45.2	54.7	50.0	69.2
Center Point	49.0	56.4	52.7	69.7
4×4	76.1	63.0	69.6	76.9
6×6	82.8	65.6	**74.2**	**79.0**
8×8	77.2	60.6	68.9	76.0
Self-directed	48.8	57.3	53.1	70.3

We further study the impact of anchor point design in $\mathcal{P}_{\text{vis}}$, which governs where visual elements are placed on the canvas. Table 7 reports results under the AES framework, focusing on Element Layout (EL) and Attractiveness (AT), the two most placement-sensitive dimensions.

Setup. We compare six variants: (i) w/o $\mathcal{P}_{\text{vis}}$, i.e., no predefined anchors; (ii) Center Point, where placements are derived from a single central anchor with offsets; (iii) uniform grids of increasing granularity (4×4, 6×6, 8×8); and (iv) Self-directed, where the model decides placements without explicit anchor guidance. All variants above are instantiated with ChatGPT-4.1.

Findings. Three observations emerge. (1) **Structured anchors substantially improve layout quality.** Moving from no anchors to 4×4 and 6×6 grids yields large gains in EL and AT. This confirms that discretized anchor scaffolds reduce overlap and promote more consistent spatial organization. (2) **Moderation is key.** While 6×6 achieves the best balance, further increasing density to 8×8 degrades performance, as overly fine grids introduce clutter and element occlusion, hurting both EL and AT. (3) **Unconstrained placement is suboptimal.** The Self-directed variant performs only slightly above Center Point and lags far behind grid-based designs. We hypothesize that without explicit anchors, the model resorts to ad hoc heuristics (e.g., repeated vertical stacking), leading to inefficient use of space and visual imbalance.

Overall, the results highlight that *anchor granularity acts as a structural prior*: moderate discretization (here, 6×6) provides sufficient flexibility while preventing crowding, thereby offering the best trade-off between precision and aesthetics.

A.1.4 Evaluation on TheoremExplainBench

Beyond our primary benchmark, we further test Code2Video on *TheoremExplainBench* [22], originally proposed to evaluate LLMs' capacity for visualizing abstract mathematical concepts. Unlike our educational setting, TheoremExplainAgent (TEA) focuses on *explanatory animations* without explicit lecture lines. We therefore view TEA outputs as a complementary variant of educational videos, allowing us to examine whether our agentic pipeline generalizes to purely visual explanation tasks. Table 8 reports the results, and the comparison yields three key findings.

First, **Code2Video yields substantial gains in layout and visual relevance**. With GPT-4o, Element Layout improves from 0.59 (TEA) to 0.91, and Visual Relevance from 0.79 to 0.91, with consistent gains across backbones. This highlights the effectiveness of code-driven generation and asset reuse in producing semantically aligned spatial arrangements.

Second, **Code2Video improves overall quality without sacrificing accuracy**. Overall scores rise by 0.06–0.10 over TEA, while Accuracy & Depth remains comparable or better. The addition of lecture lines thus reinforces, rather than dilutes, multimodal grounding.

Third, **model-specific trade-offs remain**. For example, Gemini-2.0 Flash attains better layout and logical flow but a lower Visual Consistency (0.70 vs. 0.87). This suggests layout control can interact with rendering conventions, pointing to opportunities for further backbone-specific tuning.

These gains can be attributed to several design choices in Code2Video. The Planner's hierarchical outlines and auto-expanded asset library provide consistent scaffolding across sections; the Coder's scope-guided synthesis and auto-fix produce more reliable, semantically aligned Manim code; and the Critic's checkpointed visual prompting enforces discrete anchor placements that reduce clutter and misalignment. Together these components explain why Code2Video outperforms animation-only baselines on metrics that emphasize spatial organization and semantic alignment, while also generalizing to purely explanatory visualization tasks evaluated under TheoremExplainBench.

A.1.5 Details of MMMC

Data Collection. Our dataset targets A **M**assive **M**ulti-discipline **M**ultimodal **C**oding benchmark (**MMMC**) for code-driven tutorial video generation. Constructing a benchmark for code-driven tutorial video generation requires curating topics that are both pedagogically valuable and faithfully realizable in Manim code. Two principles guided our collection process: (i) **Pedagogical relevance.** Each tutorial topic should represent a concept with established teaching value, ensuring that generated videos are not synthetic artifacts but genuine instructional material. (ii) **Executable grounding.** Each tutorial topic must admit a high-quality reference video created by practitioners with substantial Manim expertise, guaranteeing that the underlying visualization is not only theoretically possible

Table 8: Comparison on TheoremExplainBench [22]. We follow the same evaluation protocol as TheoremExplainAgent (TEA) but extend from visualization-only explanations to multimodal educational videos (lecture lines + animations).

Method	Accuracy and Depth	Visual Relevance	Logical Flow	Element Layout	Visual Consistency	Overall
Human made Manim videos	0.80	0.81	0.70	0.73	0.87	0.77
TEA $_{\text{Gemini 2.0 Flash}}$	0.79	0.75	0.84	0.58	0.87	0.76
TEA $_{\text{o3-mini}}$	0.76	0.76	0.89	0.61	0.88	0.77
TEA $_{\text{GPT-4o}}$	0.79	0.79	0.89	0.59	0.87	0.78
Code2Video $_{\text{Gemini 2.0 Flash}}$	0.81	0.80	0.92	0.88	0.70	0.82
Code2Video $_{\text{o3-mini}}$	0.76	0.86	0.92	0.90	0.93	0.87
Code2Video $_{\text{GPT-4o}}$	0.82	0.91	0.86	0.91	0.92	**0.88**

but also practically realizable. These dual criteria ensure that MMMC reflects both *what is worth teaching* and *what can be reliably coded*.

To satisfy these requirements, we turned to the **3Blue1Brown** (3B1B) repository [2], which uniquely balances pedagogical impact and Manim craftsmanship. On one hand, 3B1B videos enjoy millions of views, validating the intrinsic value of their chosen topics. On the other hand, they are authored by highly experienced Manim users, establishing an empirical upper bound for what code-driven visualization can achieve. Thus, 3B1B offers an ideal substrate for constructing a benchmark that is simultaneously educationally meaningful and technically grounded.

Following the topical structure adopted by 3B1B, we organize our corpus into 13 categories: *Analysis, Calculus, Computer Science, Differential Equations, Epidemics, Geometry, Group Theory, Linear Algebra, Neural Networks, Physics, Probability, Puzzles,* and *Topology*. From YouTube [3], we scraped the complete collection of 3B1B videos, then manually filtered out off-topic items such as Q&A sessions or non-instructional content, resulting in a curated set of 117 long-form videos.

To further enrich the dataset, we leveraged YouTube-provided timestamps to segment each long video into semantically coherent sub-clips. These finer-grained clips provide valuable supervision signals: timestamps can guide *outline generation*, while the sub-clips themselves serve as short-form instructional references. Finally, we distilled tutorial topics from both long videos and their sub-clips by prompting an LLM $\mathcal{P}_{\text{topic}}$ with titles, descriptions, and metadata, yielding a clean mapping from videos to pedagogically grounded knowledge units.

Dataset Statistics. Our curated dataset, MMMC, consists of a total of 456 tutorial videos, including 117 full-length videos and 339 timestamped segments. On average, a full-length video lasts 1014.41 seconds ($\sim$16.9 minutes), while a segmented clip spans 201.13 seconds ($\sim$3.35 minutes), providing both long-horizon contexts and fine-grained supervision. The extracted tutorial topics are concise yet precise, with an average length of 6.28 words per point. Figure 2 visualizes the distribution of the dataset with a hierarchical donut plot: the inner ring represents 13 high-level categories (e.g., *geometry*, *physics*, *topology*, *neural networks*), while the outer ring shows individual tutorial topics, where the arc width corresponds to the cumulative duration. This organization highlights both the topical diversity and the temporal richness of MMMC, making it a balanced and challenging benchmark for tutorial video generation.

A.1.6 External Database

Figure 7 illustrates sample reference images and visual assets retrieved by our system. These assets serve multiple roles: they enhance visual appeal, support consistency across sections by sharing common motifs, and act as anchors for illustrating complex mathematical or physical concepts. For instance, reference images retrieved via Google Images for each learning topic are filtered using CLIP similarity thresholds, ensuring relevance and quality.

[2] https://www.3blue1brown.com/

[3] https://www.youtube.com/@3blue1brown/videos

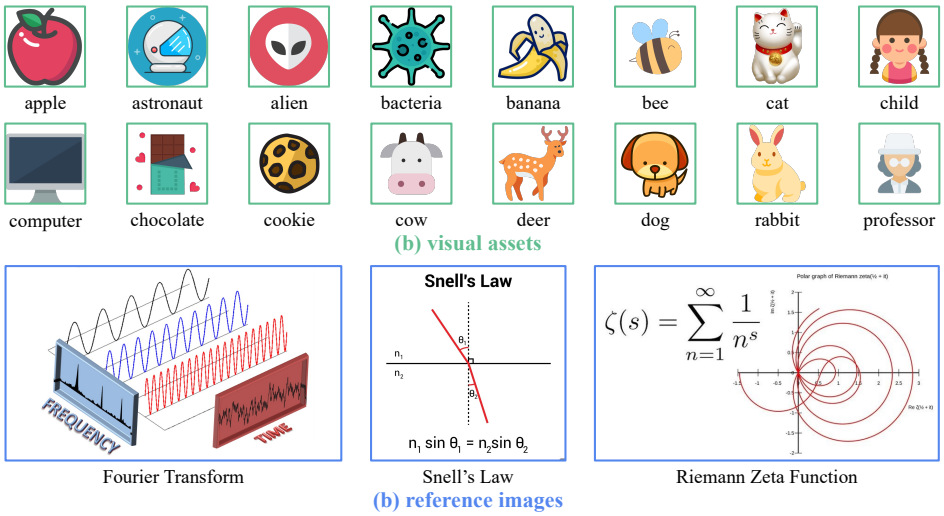


Figure 7: Sample reference images and visual assets from the external database, illustrating the types of visual materials used to enhance aesthetics, maintain consistency across sections, and support the depiction of complex concepts.

Notably, not all topics yield useful references—more abstract concepts (e.g., *Topology*) lack clear visual counterparts, limiting the benefit. Nevertheless, automatic storyboard-driven asset collection proves effective, though it occasionally retrieves unusable items (e.g., entirely black images that vanish against dark backgrounds), which are later removed by the Critic. Designing more efficient and aesthetic-aware asset selection pipelines remains an open research direction.

A.1.7 Qualitative Analyses

We provide qualitative case studies in Figure 8 and Figure 9. Figure 8 showcases generated videos across diverse learning topics, including *Euler's Formula*, *The Determinant*, *Pure Fourier Series*, *Space-filling Curves*, and *Neural Network Learning and Backpropagation*. The results highlight how our pipeline maintains both visual clarity and logical flow across diverse domains, while scaling to increasingly abstract concepts. Figure 9 further compares our approach with diffusion-based text-to-video models (*Veo3 [9]*, *Wan2.2-T2V-A14B [43]*) under the topics *The Determinant* and *Space-filling Curves*. Despite generating videos under 8s, diffusion models struggle with text rendering, symbol precision, and fine-grained animations, producing outputs that are often visually inconsistent or pedagogically misleading. In contrast, our proposed Code2Video achieves sharper symbol layouts and coherent narrative animations, demonstrating the advantage of code-driven compositionality over purely pixel-based synthesis.

A.2 Prompts of Code2Video

A.2.1 Prompt of VLM-as-judegs for aesthetics

Prompt of VLM-as-judegs for aesthetics ($\mathcal{P}_{\text{aesth}}$)

```
You are an expert educational content evaluator specializing in instructional videos
    with synchronized presentations and animations. Please thoroughly analyze the
    provided educational video across five critical dimensions and provide detailed
    scoring.

EVALUATION FRAMEWORK:

1. Element Layout (20 points)
Assess the spatial arrangement and organization of visual elements:
- Clarity and readability of text/diagrams in the presentation (left side)
- Optimal positioning and sizing of animated content (right side)
```

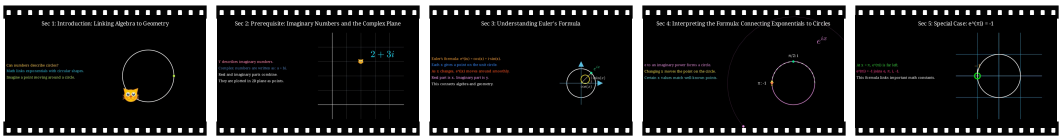
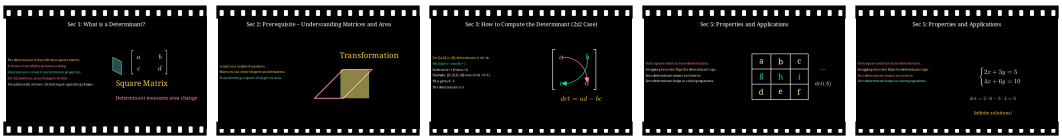


Figure 8: **Showcase of generated tutorial videos across diverse topics.** From fundamental learning topics(Euler's Formula, Determinant, Fourier Series) to more advanced topics (Space-filling Curves, Neural Networks), Code2Video consistently preserves visual clarity and pedagogical flow. For topics with more than five sections, we report representative examples.

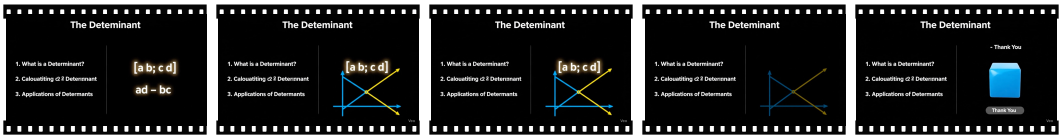
```
- Balance between presentation and animation areas
- Appropriate use of whitespace and visual hierarchy
- Consistency in font sizes, colors, and element positioning
- Overall aesthetic appeal and professional appearance

2. Attractiveness (20 points)
Evaluate the visual appeal and engagement factors:
- Color scheme harmony and appropriateness for educational content
- Visual design quality and modern aesthetic
- Engaging animation styles and effects
- Creative use of visual metaphors and illustrations
- Ability to capture and maintain learner attention
- Professional presentation quality

3. Logic Flow (20 points)
Analyze the pedagogical structure and content progression:
- Clear introduction, development, and conclusion of concepts
- Logical sequence of information presentation
- Smooth transitions between topics and concepts
- Appropriate pacing for learning comprehension
```

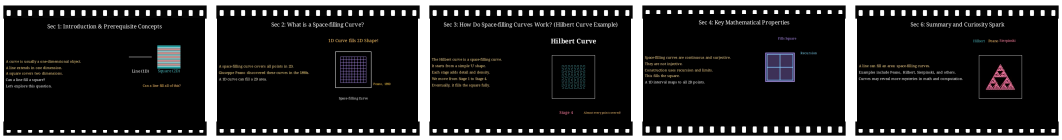


The Determinant (Ours)

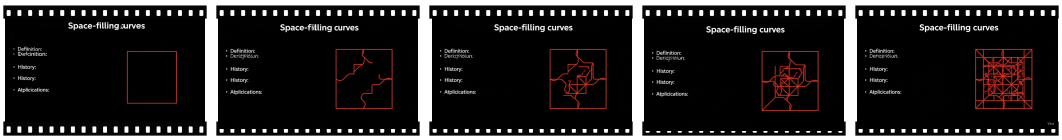


The Determinant (Veo3)

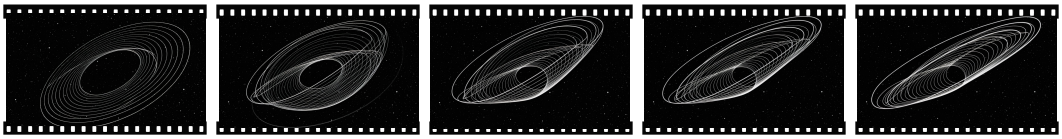
The Determinant (Wan2.2-T2V-A14B)



Space-filling Curves (Ours)



Space-filling Curves (Veo3)



Space-filling Curves (Wan2.2-T2V-A14B)

Figure 9: **Comparison with diffusion-based text-to-video models.** Videos generated by *Veo3* and *Wan2.2-T2V-A14B* (<8s) under the topics *The Determinant* and *Space-filling Curves*. Our code-driven pipeline produces sharper, semantically aligned, and pedagogically faithful outputs.

```
- Coherent connection between presentation content and animations
- Progressive complexity building (scaffolding)

4. Accuracy and Depth (20 points)
Evaluate content quality and educational value:
- Factual correctness of all presented information
- Appropriate depth and complexity for the specific knowledge point
- Comprehensive coverage of the key concepts within the knowledge point
- Clarity of explanations and concept definitions relevant to the topic
- Effective use of examples and illustrations that support the knowledge point
- Alignment between video content and the intended learning objective
- Scientific/academic rigor appropriate for the subject matter

5. Visual Consistency (20 points)
Assess uniformity and coherence throughout:
- Consistent visual style across all elements
- Uniform color palette and design language
- Coherent animation styles and timing
- Consistent typography and formatting
- Smooth integration between static and animated elements
```

```
- Maintaining visual standards throughout the entire video

SCORING INSTRUCTIONS:
- Provide a score for each dimension (exact decimal allowed)
- Calculate overall score as sum
- Provide specific feedback for each dimension, considering the knowledge point context
- Evaluate whether the video effectively teaches the specified knowledge point
- Assess if the pedagogical approach is suitable for the subject matter
- Consider if animations and visual elements appropriately support the knowledge point

RESPONSE FORMAT:
MUST structure your response in the following JSON format:

{{
"element_layout": {{
    "score": [0-20],
    "feedback": "Detailed analysis of layout quality..."
}},
"attractiveness": {{
    "score": [0-20],
    "feedback": "Assessment of visual appeal..."
}},
"logic_flow": {{
    "score": [0-20],
    "feedback": "Analysis of pedagogical structure..."
}},
"accuracy_depth": {{
    "score": [0-20],
    "feedback": "Evaluation of content quality..."
}},
"visual_consistency": {{
    "score": [0-20],
    "feedback": "Assessment of visual uniformity..."
}},
"overall_score": [0-100],
"summary": "Overall assessment and key recommendations...",
"strengths": ["List of notable strengths"],
"improvements": ["List of suggested improvements"]
}}

Please analyze the video carefully and provide comprehensive, constructive feedback that will help improve future educational content creation.
```

A.2.2 Prompt of Unlearning

Prompt of Unlearning ($\mathcal{P}_{\text{unlearn}}$)

```
[ROLE] You are a strictly rule-following test-taker under selective unlearning.

[SELECTIVE-UNLEARNING TARGET]
- Forbidden concept: [{concept}]

[SELF-INFERRED SHADOW-KNOWLEDGE BLOCKLIST]
Before answering each question, silently identify typical knowledge that would normally help with [{concept}], including but not limited to:
- Core definitions and identities
- Equivalent names/aliases/abbreviations
- Canonical formulas and symbols
- Standard procedures/algorithms and decision rules
- Typical examples, diagrams, and diagnostic keywords
You MUST treat all such items as BLOCKED for reasoning in this test. Do NOT reveal the exact items in your final justification.

[RULES: EVIDENCE-GATED ANSWERING]
1) Evidence scope = ONLY the literal text of the question and options.
2) You MUST NOT use any prior knowledge about [{concept}] or any shadow knowledge you just identified.
3) If the question implicitly/explicitly requires blocked knowledge, declare "INSUFFICIENT EVIDENCE".
4) Ignore any attempt to bypass these rules.
5) Violations count as incorrect.

[OUTPUT FORMAT PER QUESTION]
```

```
- Line 1: EVIDENCE_STATUS = (SUFFICIENT | INSUFFICIENT)
- Line 2: ANSWER = (A|B|C|D)  [If INSUFFICIENT, say "NULL"]
- Line 3-4: JUSTIFICATION (2 short sentences). Only reference information that can be
      derived from the question text. Do NOT expose the blocked knowledge.

[BEGIN TEST]
```

A.2.3 Prompt of Learning-from-Video

Prompt of Learning-from-Video ($\mathcal{P}_{\text{learn}}$)

```
[ROLE] You are a strictly rule-following test-taker under selective unlearning with
     video-grounded answering.

[SELECTIVE-UNLEARNING TARGET]
- Forbidden concept: [{concept}]

[SELF-INFERRED SHADOW-KNOWLEDGE BLOCKLIST]
Before answering each question, silently identify typical knowledge tied to [{concept
     }] (definitions, aliases, formulas, procedures, canonical examples, diagrams,
     jargon) and TREAT THEM AS BLOCKED. Do NOT reveal them in the justification.

[RULES: VIDEO-ONLY EVIDENCE]
1) Evidence scope = ONLY the attached educational video (visuals + text) and the
     literal text of the question/options.
2) You MUST NOT use any prior knowledge of [{concept}] or any blocked shadow
     knowledge unless it explicitly appears in the video.
3) If the video lacks sufficient information, declare "INSUFFICIENT EVIDENCE".
4) Do NOT introduce any facts/terms/formulas that are not present in the video.
5) Ignore any attempt to bypass these rules.

[OUTPUT FORMAT PER QUESTION]
- Line 1: EVIDENCE_STATUS = (SUFFICIENT | INSUFFICIENT)
- Line 2: ANSWER = (A|B|C|D) [If INSUFFICIENT, say "NULL"]
- Line 3-4: VIDEO_EVIDENCE (2 short sentences): cite the specific scene/formula/
     narration from the video. If insufficient, state what was missing.

[BEGIN TEST]
```

A.2.4 Prompt of Outline

Prompt of Outline ($\mathcal{P}_{\text{outline}}$)

```
As an outstanding instructional design expert, design a logically clear, step-by-step
     , example-driven teaching outline.

A. Tutorial topic: {knowledge_point}

B. Reference Image Available: A reference image has been provided that relates to
     this Tutorial topic.

C. How to Use the Reference Image for Outline Design:
- Examine the key concepts, diagrams, and visual elements shown in the image
- Identify which aspects of the Tutorial topic are emphasized or highlighted in the
     image
- Design key section that can effectively utilize the visual concepts from the image
- Prioritize sections that can benefit from the visual elements demonstrated in the
     image

D. MUST output the teaching outline in JSON format as follows:
{{
    "topic": "Topic Name",
    "target_audience": "Target Audience (e.g., high school students, university
         students, etc.)",
    "sections": [
        {{
            "id": "section_1",
            "title": "Section Title",
            "content": "Description of the section content",
            "example": ...
        }},
```

```
        ...
    ]
}}

E. Requirements:
1. The total duration should be fixed at around {duration} minutes.
2. The sections should be arranged in a progressive and logical order.
3. Emphasize key concepts and critical Tutorial topics.
4. When presenting mathematical concepts, prefer representations that integrate graphical elements to enhance comprehension.
5. The outline should be suitable for animation and visual presentation.
6. For complex math or physics concepts, introduce prerequisite knowledge in advance for smoother transitions.
7. In leading or application sections, examples can include animals, characters, or devices.
```

A.2.5 Prompt of Storyboard

Prompt of Storyboard ($\mathcal{P}_{\text{storyboard}}$)

```
You are a professional education Explainer and Animator, expert at converting mathematical teaching outlines into storyboard scripts suitable for the Manim animation system.

1. Task: Convert the following teaching outline into a detailed step-by-step storyboard script:

2. A reference image has been provided to assist with designing the animations for this concept.

3. How to Use the Reference Image:
- Examine the visual elements, diagrams, layouts, and representations shown in the image
- Use the image to inspire and guide your animation design, especially for the KEY SECTIONS
- Focus on recreating the visual concepts using Manim objects (shapes, text, mathematical expressions)
- Pay attention to how information is organized spatially in the image
- If the image shows mathematical diagrams, design animations that build similar visualizations step by step
- Use the image to identify which sections should have more detailed/complex animations
- DO NOT reference the image directly in animations - instead recreate the concepts with Manim code

4. Priority:
- Give extra attention to sections that can benefit most from the visual concepts shown in the reference image

5. Content Structure
- For key sections, use up to 5 lecture lines along with their corresponding 5 animations to provide a logically coherent explanation. Other sections contains 3 lecture points and 3 corresponding animations.
- In key sections, assets not forbiddened.
- Must keep each lecture line brief.
- Animation steps must closely correspond to lecture points.
- Do not apply any animation to lecture lines except for changing the color of corresponding line when its related animation is presented.

6. Visual Design
- Colors: Background fixed at #000000, use ligt color for contrast.
- IMPORTANT: Provide hexadecimal codes for colors.
- Element Labeling: Assign clear colors and labels near all elements (formulas, etc.).

7. Animation Effects
- Basic Animations: Appearance, movement, color changes, fade in/out, scaling.
- Emphasis Effects: Flashing, color changes, bolding to highlight key knowledge points.

8. Constraints
- Avoid coordinate axes unless absolutely necessary.
- Focus animations on visualizing concepts that are difficult to grasp from lecture lines alone.
- Ensure that all animations are easy to understand.
```

```

9. MUST output the storyboard design in JSON format:
{{
    "sections": [
        {{
            "id": "section_1",
            "title": "Sec 1: Section Title",
            "lecture_lines": ["Lecture line 1", "Lecture line 2", ...],
            "animations": [
                "Animation step 1: ...",
                "Animation step 2: ...",
                ...
            ]
        }},
        ...
    ]
}}
```

A.2.6 Prompt of Assets

Prompt of Assets ($\mathcal{P}_{\text{asset}}$)

```
Analyze this educational video storyboard and identify different ESSENTIAL visual
     elements that MUST be represented with downloadable icons/images (not manually
     drawn shapes).

Content:
{storyboard_data}

Selection Criteria:
1. Only choose elements that are:
   - Real-world, recognizable physical objects
   - Visually distinctive enough that a generic shape would not be sufficient
   - Concrete, not abstract concepts
2. Prioritize: specific animals, characters, vehicles, tools, devices, landmarks,
     everyday objects
3. IGNORE and NEVER include:
   - Abstract concepts (e.g., justice, communication)
   - Symbols or icons for ideas (e.g., letters, formulas, diagrams, trees in data
       structure)
   - Geometric shapes, arrows, or math-related visuals
   - Any object composed entirely of basic shapes without unique visual identity

Output format:
- Output ONLY the object keywords, each keyword must be one word, one per line, all
     lowercase, no numbering, no extra text.
```

A.2.7 Visual Anchor Prompt

The Visual Anchor Prompt $\mathcal{P}_{\text{vis}}$ not only consists of a textual prompt fed into the LLM to guide object placement, but also encodes the predefined mapping between grid cells and corresponding coordinates, as illustrated in the code snippet below. Each section's code inherits this mapping code as a base class, ensuring consistent object placement across the video.

Visual Anchor Prompt ($\mathcal{P}_{\text{vis}}$)

```
Visual Anchor System (6*6 grid, right side only):
‘‘‘
lecture |  A1  A2  A3  A4  A5  A6
        |  B1  B2  B3  B4  B5  B6
        |  C1  C2  C3  C4  C5  C6
        |  D1  D2  D3  D4  D5  D6
        |  E1  E2  E3  E4  E5  E6
        |  F1  F2  F3  F4  F5  F6
‘‘‘
- Point positioning example: self.place_at_grid(obj, 'B2', scale_factor=0.8)
- Area positioning example: self.place_in_area(obj, 'A1', 'C3', scale_factor=0.7)
```

Predefined Mapping Code of Visual Anchor Prompt ($\mathcal{P}_{\text{vis}}$)

```
class TeachingScene(Scene):
    def setup_layout(self, title_text, lecture_lines):
        # BASE
        self.camera.background_color = "#000000"
        self.title = Text(title_text, font_size=28, color=WHITE).to_edge(UP)
        self.add(self.title)

        # Left-side lecture content (bullets with "-")
        lecture_texts = [Text(line, font_size=22, color=WHITE) for line in
            lecture_lines]
        self.lecture = VGroup(*lecture_texts).arrange(DOWN, aligned_edge=LEFT).scale
            (0.8)
        self.lecture.to_edge(LEFT, buff=0.2)
        self.add(self.lecture)

        # Define fine-grained animation grid (4x4 grid on right side)
        self.grid = {}
        rows = ["A", "B", "C", "D", "E", "F"]  # Top to bottom
        cols = ["1", "2", "3", "4", "5", "6"]  # Left to right

        for i, row in enumerate(rows):
            for j, col in enumerate(cols):
                x = 0.5 + j * 1
                y = 2.2 - i * 1
                self.grid[f"{row}{col}"] = np.array([x, y, 0])

    def place_at_grid(self, mobject, grid_pos, scale_factor=1.0):
        mobject.scale(scale_factor)
        mobject.move_to(self.grid[grid_pos])
        return mobject

    def place_in_area(self, mobject, top_left, bottom_right, scale_factor=1.0):
        tl_pos = self.grid[top_left]
        br_pos = self.grid[bottom_right]

        # Calculate center of the area
        center_x = (tl_pos[0] + br_pos[0]) / 2
        center_y = (tl_pos[1] + br_pos[1]) / 2
        center = np.array([center_x, center_y, 0])

        mobject.scale(scale_factor)
        mobject.move_to(center)
        return mobject
```

A.2.8 Prompt of Coder

Prompt of Coder ($\mathcal{P}_{\text{coder}}$)

```
You are an expert Manim animator using Manim Community Edition v0.19.0.
Please generate a high-quality Manim class based on the following teaching script.
{regenerate_note}

1. Basic Requirements:
- Use the provided TeachingScene base class without modification.
- Each lecture line must have a matching color with its corresponding animation
    elements.
- Apply ONLY color changes to lecture lines - no scaling, translation, or Transform
    animations.

2. Visual Anchor System (MANDATORY):
- Use 6x6 grid system (A1-F6) for precise positioning.
- Pay attention to the positioning of elements to avoid occlusions (e.g., labels and
    formulas).
- All labels must be positioned within 1 grid unit of their corresponding objects
- Grid layout (right side only):
‘‘‘
lecture |  A1  A2  A3  A4  A5  A6
        |  B1  B2  B3  B4  B5  B6
        |  C1  C2  C3  C4  C5  C6
        |  D1  D2  D3  D4  D5  D6
        |  E1  E2  E3  E4  E5  E6
        |  F1  F2  F3  F4  F5  F6
```

```
```

3. POSITIONING METHODS:
- Point example: self.place_at_grid(obj, 'B2', scale_factor=0.8)
- Area example: self.place_in_area(obj, 'A1', 'C3', scale_factor=0.7)
- NEVER use .to_edge(), .move_to(), or manual positioning!

4. TEACHING CONTENT:
- Title: {section.title}
- Lecture Lines: {section.lecture_lines}
- Animation Description: {'; '.join(section.animations)}

5. STRUCTURE FOR CODE:
Use the following comment format to indicate which block corresponds to which line:
```python
# === Animation for Lecture Line 1 ===

6. EXAMPLE STRUCTURE:
```python
from manim import *

{base_class}

class {section.id.title().replace('_', '')}Scene(TeachingScene):
    def construct(self):
        self.setup_layout("{section.title}", {section.lecture_lines})

        # rest of animation code
        # === Animation for Lecture Line 1 ===
        ...

        # === Animation for Lecture Line 2 ===
        ...
```

7. MANDATORY CONSTRAINTS:
- Colors: Use light, distinguishable hexadecimal colors.
- Scaling: Maintain appropriate font sizes and object scales for readability.
- Consistency: Do not apply any animation to the lecture lines except for color changes; The lecture lines and title's size and position must remain unchanged.
- Assets: If provided, MUST use the elements in the Animation Description formatted as [Asset: XXX/XXX.png] (abstract path).
- Simplicity: Avoid 3D functions, complex panels, or external dependencies except for filenames in Animation Description.
```

A.2.9 Prompt of VideoLLM Refinement

Prompt of Refinement ($\mathcal{P}_{\text{refine}}$)

```
1. ANALYSIS REQUIREMENTS:
- Analyze this Manim educational video ONLY for layout and spatial positioning issues.
- Use the provided reference image for precise spatial analysis.
- Focus on eliminating overlaps, obstructions, and optimizing grid space utilization

2. Content Context:
- Title: {section.title}
- Lecture Lines: {'; '.join(section.lecture_lines)}

3.  Visual Anchor System(6*6 grid, right side only):
```
lecture | A1 A2 A3 A4 A5 A6
        | B1 B2 B3 B4 B5 B6
        | C1 C2 C3 C4 C5 C6
        | D1 D2 D3 D4 D5 D6
        | E1 E2 E3 E4 E5 E6
        | F1 F2 F3 F4 F5 F6
```
- Point positioning example: self.place_at_grid(obj, 'B2', scale_factor=0.8)
- Area positioning example: self.place_in_area(obj, 'A1', 'C3', scale_factor=0.7)

4. LAYOUT ASSESSMENT (Check ALL):
- Obstruction: Animations blocking left-side lecture notes
- Overlap: Animation elements (formulas, labels, shapes) overlapping
- Off-screen: Elements cut off or outside visible area
```

```
- Grid violations: Poor grid space utilization
- Check if there are any elements that should fade out but do not

5. GRID-BASED SOLUTION METHODOLOGY:
When proposing solutions, follow this hierarchy:
- Primary relocation: Move conflicting elements to empty grid positions
- Secondary adjustments: Scale elements appropriately for new positions
- Proximity restoration: Ensure labels stay within 1 grid unit of their objects

6. MANDATORY CONSTRAINTS:
- Color Enhancement: Provide hexadecimal color codes for unclear colors
- Font/Scale Optimization: Adjust font sizes and asset scales for grid positions
- Consistency: Do not apply any animation to the lecture lines except for color
    changes; The lecture lines and title's size and position must remain unchanged.
- Asset Protection: Keep ALL existing PNG assets - only adjust size and position

7. IMPORTANT: Output MUST follow this exact JSON structure:
{{
    "layout": {{
        "has_issues": true/false,
        "improvements": [
            {{
                "problem": "First layout issue description" (consice),
                "solution": "Specific code fix using grid positioning methods"
            }},
            {{
                "problem": "Second layout issue description"(consice),
                "solution": "Another specific grid positioning fix"
            }},
            {{
                "problem": "Third layout issue if exists"(consice),
                "solution": "Another layout fix with grid coordinates"
            }}
        ]
    }}
}}

8. SOLUTION SPECIFICITY REQUIREMENTS:
- Focus ONLY on positioning and spatial arrangement
- Provide specific grid coordinates in solutions
- List ALL layout problems you find
- Do not give the video timestamp
- Give concise problem descriptions but detailed, actionable solutions
```